
# *ODYSSEYPROCESSREUSE*: UMA PROPOSTA DE SISTEMÁTICA DE ENGENHARIA DE LINHA DE PROCESSO DE SOFTWARE BASEADA EM UM PROJETO ARQUITETURAL DE COMPONENTES DE PROCESSOS DE SOFTWARE

Eldanae Nogueira Teixeira

Exame de Qualificação apresentado ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadora: Cláudia Maria Lima Werner

Rio de Janeiro

Março de 2014

# Índice

# Índice de Figuras

# ÍNDICE DE TABELAS

# CAPÍTULO I

# INTRODUÇÃO

## 1.1. Motivação

Diante da ampla competitividade e dinamicidade de mercado, as organizações têm buscado técnicas que aumentem a produtividade, a eficiência e a efetividade das soluções de software para fornecer a flexibilidade necessária no provimento de serviços e produtos de software que atendam às exigências dos usuários. Sendo um processo de software definido 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), a qualidade desses produtos é influenciada pela qualidade dos processos de desenvolvimento adotados para construí-los (OSTERWEIL, 1987; CUGOLA e GHEZZI 1998; FUGGETTA, 2000). No entanto, a definição de processos de software não é trivial. Alguns modelos de processos de referência, tais como o CMMI (CHRISSIS *et al*., 2006), a ISO/IEC 12207 (ISO/IEC, 2008) e o MPS.Br (SOFTEX, 2012), foram criados como forma de auxiliar no gerenciamento orientado a processos e na agregação de boas práticas de desenvolvimento de software reconhecidas. Esses modelos tratam de descrições genéricas de processos de software, especificando seus propósitos e resultados esperados dentro de um determinado grau de maturidade para esses processos dentro de uma organização. No entanto, devido à diversidade de organizações e projetos, suas características e objetivos, estes modelos não são diretamente aplicáveis a qualquer projeto de desenvolvimento de software, pois não contam com a especificação de um conjunto de atividades e ações específicas para alcançar os propósitos e resultados esperados descritos.

Desta forma, novos processos específicos de projeto podem ser derivados através de adaptações de processos definidos organizacionalmente ou conduzidos em projetos anteriores. A adaptação de processos de software pode ser entendida como o ato de ajustar as definições ou particularizar os termos de uma descrição de processo geral com o intuito de derivar um novo processo aplicável a um ambiente alternativo (e, provavelmente, menos geral) (GINSBERG e QUINN, 1995). Essa adaptação deve definir, estabelecer o escopo e executar as mudanças de uma maneira sistemática e consistente.

Há uma grande quantidade de material indicando as melhores práticas a serem seguidas, mas cada organização deve definir seus processos levando em consideração não só essas informações, mas também suas próprias características (BERTOLLO *et al*., 2006). Processos de software podem ser classificados como sistemas dinâmicos complexos, uma categoria de sistemas com uma grande complexidade estrutural e escalar (STERMAN, 1992). Assim, definir um processo de software exige experiência, envolve o conhecimento de muitos

aspectos da engenharia de software e deve levar em conta muitos fatores, tais como: necessidades e características da organização ou projeto, técnicas e métodos que serão utilizados, conformidade com padrões ou modelos de referência, restrições de negócio (prazo, custo, etc.) (BARRETO, 2007). A mesma experiência é requerida nas demais atividades do ciclo de vida da engenharia de processos de software, com ênfase no gerenciamento, onde os gerentes utilizam suas habilidades e sua experiência para tomar decisões, enquanto a equipe executa o processo de desenvolvimento de software (BARROS, 2001). Os gerentes de projeto mais experientes, geralmente, realizam este processo de tomada de decisão mentalmente. Entretanto, como um modelo mental não pode ser diretamente reutilizado por gerentes menos experientes, faz-se necessária uma representação explícita para o conhecimento tácito dos gerentes experientes, assim como uma técnica que permita a avaliação do impacto destes problemas e oportunidades sobre um projeto de desenvolvimento de software (BARROS, 2001). O mesmo raciocínio pode ser aplicado à área de definição de processos específicos de projetos, onde é de se esperar que caso o conhecimento dos engenheiros de processo experientes seja explicitado, formalizado e disponibilizado para outros profissionais, seja possível reutilizar esse conhecimento de forma mais efetiva (BARRETO, 2007).

Diante disso, abordagens de apoio às diferentes tarefas envolvidas nas etapas de desenvolvimento, execução, gerenciamento e melhoria de processos de software tornam-se essenciais, principalmente através da aplicação de conhecimento adquirido na condução de projetos anteriores, de forma a minimizar a complexidade e o esforço envolvidos. A reutilização de processos de software vem se destacando como uma das principais práticas para prover a melhoria contínua de processos, por aproveitar as informações (tecnológicas e gerenciais) produzidas durante desenvolvimentos de software passados, com o objetivo de reduzir o esforço necessário para novos desenvolvimentos (BARRETO, 2007). OSTERWEIL (1987) já argumentava que processos de software são software também e, assim como software, podem ser modelados, desenvolvidos e testados. Transferindo esta analogia para o campo de reutilização, técnicas de reutilização de software, como a Engenharia de Domínio (ED) (ARANGO e PRIETO-DIAZ, 1991) e, uma de suas vertentes, a Linha de Produtos de Software (LPS) (NORTHROP, 2002), e ainda o Desenvolvimento Baseado em Componentes (DBC) (SAMETINGER, 1997) têm sido adaptadas ao contexto de definição de processos de software (KELLNER, 1996; ROMBACH, 2006).

Uma das técnicas de reutilização de processos, denominada Linha de Processos, foi proposta em abordagens (JAUFMAN e MÜNCH, 2005; ROMBACH, 2006; WASHIZAKI, 2006a; ARMBRUST *et al.,* 2009; BARRETO *et al*., 2009, ALEIXO *et al*., 2010, ALEGRÍA *et al*., 2012) que trabalham o conceito de LPS cujos produtos são processos. Uma Linha de Processos de Software (LPrS) pode ser definida como “um conjunto de processos de software que

compartilham um conjunto de características comuns e variáveis, e são desenvolvidas a partir de artefatos (*core assets*), que podem ser reutilizados e combinados entre si, segundo regras de composição e recorte, para compor e adaptar processos de software" (NUNES *et al*., 2010; MAGDALENO, 2010; TEIXEIRA, 2011).

Outro campo dentro da reutilização de processos trata da definição de processos através da composição de elementos menores (componentes de processos), que sintetizam e encapsulam a semântica de conhecimento existente. Existem diferentes abordagens que trabalham o conceito de componente de processos (GARY e LINDQUIST, 1999; RU-ZHI *et al.,* 2005; FUSARO *et al*., 1998; BHUTA *et al*., 2005). Desta forma, o processo é derivado a partir de blocos de construção menores que devem ser estruturados através de um arcabouço estrutural.

Resumindo, o propósito da área de reutilização de processos de software é facilitar a definição de processos, diminuindo o custo e o esforço associados à atividade, além de possivelmente aumentar a qualidade dos processos gerados, inclusive tornando a realização da atividade acessível a profissionais menos experientes (BARRETO, 2007). A dificuldade na reutilização de processos de software reside no fato de que a variabilidade dos requisitos dos projetos frequentemente implica em variabilidade dos processos de software (ROUILLÉ *et al*., 2013). Para alcançar uma reutilização de processo efetiva, é necessário desenvolver métodos para capturar elementos comuns e variáveis de processos e criar definições de processos que possam ser aplicadas em uma variedade de situações, ou seja, que são reutilizáveis (HOLLENBACH e FRAKES, 1996).

## 1.2. Caracterização do Problema

Seguindo a analogia com a reutilização de software e com base em trabalhos nesta área (BLOIS, 2006) que estudam a oportunidade de promover a reutilização de artefatos através da combinação das áreas de ED e DBC, pode-se analisar a mesma oportunidade no campo de processos de software.

A abordagem de LPrS é recente e não está ainda totalmente consolidada na literatura (BARRETO *et al*., 2009). Mesmo com todos os resultados na área, esse campo de pesquisa ainda apresenta desafios a serem perseguidos. As abordagens tratam da concepção, modelagem e implementação de LPrS, tornando clara a necessidade de mais estudos empíricos e avaliações das propostas (JUNIOR *et al*., 2013). Apesar dos esforços na área, não há uma abordagem única ou um consenso sobre como realizar a adaptação de processo de uma maneira controlada e consistente (MARTÍNEZ-RUIZ *et al*., 2012).

Um conjunto de abordagens foi analisado em TEIXEIRA (2011), onde foi observado que nenhuma das representações propostas consegue atender de forma completa e explícita os

requisitos de uma notação para modelagem de variabilidades em LPrSs. De forma complementar, uma das conclusões do estudo realizado por MARTÍNEZ-RUIZ *et al*. (2012), para identificação dos requisitos de uma notação para suporte à adaptação de processos, é que apenas alguns trabalhos apresentam uma notação especial para representação de variabilidade em processos, e o desenvolvimento de uma notação que atenda a requisitos para modelagem de variabilidades é uma necessidade real.

Abordagens de LPrS, em geral, não contemplam a construção dos artefatos que as compõem em diferentes níveis de abstração complementares com tratamento de rastreabilidade (análise, projeto e implementação). O mesmo ocorre em abordagens de Definição de Processo de Software Baseada em Componentes (DPBC), onde o conceito de variabilidade é considerado de forma superficial e não é tratado através de representações gráficas. A forma de representar toda a informação especificada é de extrema relevância para o entendimento e aplicação eficaz e consistente dos artefatos desenvolvidos. Conceitos de modularidade devem ser aplicados no contexto do nível de componentização dos elementos de processos especificados no nível de análise.

Além disso, a organização da LPrS através de uma Arquitetura de Componentes de processos ainda pode ser melhor apoiada através de sistemáticas para o agrupamento de artefatos de processo reutilizáveis por meio de critérios, métricas ou outras técnicas. Uma Arquitetura de Processo consiste em um arcabouço conceitual para incorporar, relacionar e adaptar elementos de processos em instâncias de processos (HUMPRHEY, 1989).

Assim, um desafio identificado é promover a reutilização de processos de software através de uma sistemática de reutilização que trabalhe a propriedade de variabilidade inerente a famílias de processos e estruture o conhecimento do domínio através de níveis de abstração semanticamente complementares, com a visão geral representada através de uma arquitetura baseada em componentes de processos de software e com uma notação que trate a variabilidade do domínio de processos de software e mecanismos para garantir a rastreabilidade entre os múltiplos artefatos.

## 1.3. Objetivos

Diante do exposto, *o objetivo central deste trabalho é definir, implementar e avaliar uma sistemática de Engenharia de Linha de Processo de Software para apoiar a reutilização de processos de software através de um Projeto Arquitetural de Componentes de Processos de Software.*

A proposta envolve a definição de uma Engenharia de LPrS, um método sistemático para desenvolver uma LPrS, especificando uma Arquitetura de Domínio Baseada em Componente de Processos de Software. Tal abordagem é composta por cinco elementos

principais: (1) um método, que especifica um processo de Engenharia de LPrS com foco nos conceitos de definição de processos baseada em componentes; (2) uma representação para modelagem de variabilidades de uma LPrS com diferentes níveis de abstração e artefatos; (3) mecanismos de rastreabilidade para estabelecer os mapeamentos entre os artefatos do domínio representados em diferentes modelos e níveis de abstração; (4) procedimentos de suporte a organização dos elementos que compõem a Arquitetura da LPrS baseada em componentes; e (5) uma infraestrutura de reutilização.

Esta proposta de abordagem tem como objetivos principais:

1. A definição de um Processo de Desenvolvimento *para* Reutilização com foco na definição dos artefatos que compõem uma LPrS.
2. Definição de representações de modelagem do domínio de processos de software através de diferentes níveis de abstração complementares de uma Linha de Processos de Software com suporte a representação de uma Arquitetura de Componentes de Processos de Software.
3. A definição de mecanismos de rastreabilidade para apoiar o mapeamento entre os artefatos do domínio nos diferentes níveis de abstração, estabelecendo a rastreabilidade necessária entre os elementos, permitindo que um dado componente de processo de software seja descrito através de diferentes visões, desde representações abstratas de suas unidades de trabalho e conceitos, passando por um detalhamento maior, através de diagramas adicionais representativos de sua estrutura e seu comportamento.
4. A definição de um conjunto de procedimentos como o uso de critérios, métricas ou técnicas para apoiar a criação dos componentes arquiteturais.
5. A construção de apoio ferramental como infraestrutura de utilização da abordagem proposta, como extensão do ambiente Odyssey (ODYSSEY, 2014), infraestrutura de reutilização baseada em modelos de domínio, que utiliza técnicas de LPS e DBC.

## 1.4. Metodologia de Pesquisa

A metodologia para desenvolvimento da abordagem foi especificada segundo um conjunto de passos, descritos a seguir:

1. Realização de uma revisão da literatura nas principais áreas envolvidas no desenvolvimento deste trabalho de pesquisa;
2. Especificação do método de Engenharia de LPrS, através da descrição dos fluxos de atividades do desenvolvimentos dos artefatos da LPrS, com foco em conceitos de DPBC;

3. Especificação da representação da LPrS, através da definição dos modelos em diferentes níveis de abstração com gerenciamento dos conceitos de opcionalidade e variabilidade;
4. Definição dos procedimentos de mapeamento entre os diferentes artefatos do domínio e dos diferentes níveis de abstração;
5. Definição de procedimentos para estruturar os componentes de processo em elementos da Arquitetura da LPrS;
6. Estabelecimento das estratégias de avaliação da abordagem e dos diferentes artefatos que a compõem, segundo técnicas consolidadas na literatura, ao longo do processo, com e sem ferramental de apoio;
7. Desenvolvimento da infraestrutura de reutilização, através da adaptação do ambiente Odyssey; e
8. Publicação dos resultados do trabalho, ao longo e ao término dos passos anteriores.

## 1.5. Projetos Relacionados

Esta proposta de tese será desenvolvida com apoio dos trabalhos realizados dentro do grupo de reutilização da COPPE/UFRJ. Um conjunto de trabalhos foi realizado dentro do projeto Reuse (WERNER, 2005), que tinha como objetivo principal explorar técnicas e ferramentas que dêem apoio à Reutilização de Software, particularmente nos contextos da Engenharia de Domínio e do Desenvolvimento Baseado em Componentes. Dentre as técnicas propostas, podemos citar: 1) Busca e Recuperação de Componentes em Ambientes de Reutilização de Software (BRAGA, 2000); 2) Máquina de Processos (MURTA, 2002); 3) Representação e Consistência de variabilidades (OLIVEIRA, 2006); 4) Projeto Arquitetural Baseado em Componentes no Contexto de Engenharia de Domínio (BLOIS, 2006) e Abordagem de Apoio à Criação de Arquiteturas de Referência de Domínio (VASCONCELOS, 2007). Tal projeto teve como origem o Projeto Odyssey (WERNER *et al.*, 1999). Partindo da analogia entre software e processos de software (OSTERWEIL, 1987), tais trabalhos serão utilizados como possíveis fontes de informação para auxiliar a definição das técnicas de reutilização de processos aqui propostas.

O ambiente Odyssey (ODYSSEY, 2014), desenvolvido pela equipe de reutilização de software da COPPE/UFRJ desde 1998, tem como objetivo principal prover mecanismos para reutilização, visando o desenvolvimento de software baseado em componentes. Este ambiente serve como um arcabouço em que modelos conceituais e arquiteturas de software são especificados para domínios de aplicações e será estendido ao longo do desenvolvimento deste trabalho como infraestrutura de reutilização de processos de software.

Atualmente, um novo projeto, Projeto CDSoft[1] (WERNER *et al.*, 2011), está sendo desenvolvido com o objetivo de definir uma abordagem e desenvolver um ferramental de apoio à definição de processos de software suportada por sistemáticas de reutilização. Tal projeto é uma parceria entre o Grupo de Reutilização de Software da COPPE/UFRJ e o Programa de Pós-Graduação em Informática da UNIRIO. A intenção do projeto é prover soluções para apoiar o gerente de projeto na tomada de decisão sobre a melhor forma de compor um processo para o projeto. Com o propósito de facilitar esta atividade, está sendo desenvolvido um suporte computacional ao gerente de projeto, possivelmente diminuindo o esforço necessário para sua execução e melhorando os resultados obtidos. O projeto envolve diferentes linhas de pesquisa, na qual uma delas trata-se da sistemática proposta neste trabalho. A visão geral do projeto é apresentada na Figura 1, onde podem ser observadas as suas principais linhas de pesquisa: (a) **Engenharia de LPrS** (*sistemática desenvolvida neste trabalho*); (b) Gestão de Contexto (GC) (NUNES *et al.*, 2012); (c) LPrS baseada em Contexto (NUNES *et al.*, 2010); e (d) COMPOOTIM (MAGDALENO, 2013).

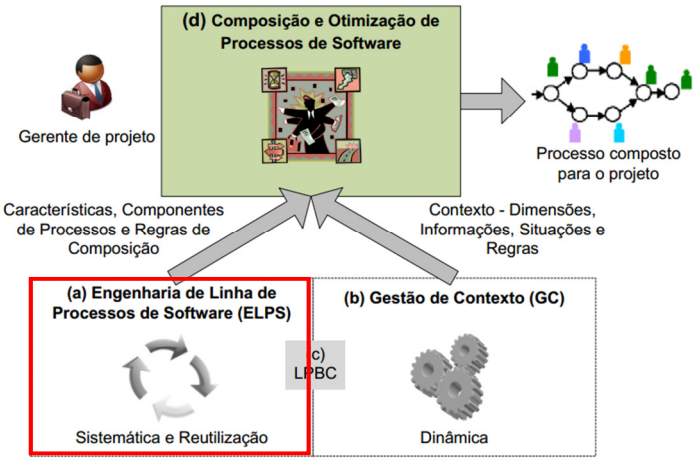



**Figura 1 - Visão Geral Projeto CDSoft (MAGDALENO, 2013)**

## 1.6. Organização da Monografia

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

O **Capítulo 2** apresenta a Fundamentação Teórica, com a descrição dos conceitos fundamentais sobre processos de software, incluindo uma análise das diferentes etapas do ciclo de vida de processos de software e suas principais características, além de questões de modelagem de processos. O conceito de Variabilidade em processos de software é também discutido, assim como uma breve apresentação da área de reutilização de processos de software.

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

O **Capítulo 3** apresenta uma revisão da área de reutilização de processos de software, com foco para as abordagens de LPrS. O objetivo é analisar tais abordagens segundo um conjunto de critérios estabelecidos e identificar oportunidades de melhoria e limitações que podem ser trabalhadas. As diferentes possibilidades de representação de variabilidades existentes nas abordagens de LPrS identificadas também são analisadas.

O **Capítulo 4** trata da abordagem proposta. Apresenta em linhas gerais a abordagem de Engenharia de Linha de Processos de Software com projeto arquitetural baseado em componentes. O Capítulo busca especificar os elementos que compõem a abordagem, descrevendo, inicialmente, como os objetivos serão alcançados e os resultados preliminares já desenvolvidos.

O **Capítulo 5** apresenta as principais contribuições esperadas e possíveis limitações identificadas, além de descrever as atividades de pesquisa, e uma proposta inicial de cronograma para a realização do trabalho. O planejamento de estudo para avaliar a viabilidade de aplicação da abordagem proposta na atividade de desenvolvimento de LPrS é brevemente discutido neste capítulo.

# CAPÍTULO II

# FUNDAMENTAÇÃO TEÓRICA

## 2.1. Introdução

O objetivo deste capítulo consiste em apresentar os fundamentos e conceitos básicos requeridos para entendimento da abordagem aqui proposta. Desta forma, os conceitos de processos e as diferentes etapas do ciclo de vida que compõem sua engenharia são descritas na Seção 2.2. A Seção 2.3 apresenta o conceito de variabilidade, fortemente presente nas variações de definições de processos propostas para atender os múltiplos aspectos dos diferentes projetos que apoiam. A área de reutilização de processos de software é brevemente apresentada na Seção 2.4 com o objetivo de destacar as analogias existentes entre produto de software e processo de software (OSTERWEIL, 1987) aplicadas no campo da reutilização, como forma de apoiar a definição de novos processos a partir de conhecimento previamente adquirido em desenvolvimentos de projetos anteriores, através da adaptação de algumas técnicas de reutilização de software para a reutilização de processos.

## 2.2. Processo de Software

Um processo de software pode ser definido 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). O ciclo de vida de um processo de software abrange as atividades de engenharia de um processo. As atividades deste ciclo são chamadas de meta-atividades, e o ciclo de vida é chamado de meta-processo de software (DERNIAME *et al.*, 1999). Existem diversas propostas de ciclo de vida para processos de software, como, por exemplo, NGUYEN e CONRADI (1994) mostram uma classificação de meta-processos, propõem fases para o meta-processo e comparam alguns ambientes europeus quanto à adoção das fases propostas. Por outro lado, DERNIAME *et al*. (1999) propõem um ciclo de vida de referência para processos de software chamado *PROMOTER Reference Model*. Em REIS (2003), um ciclo de vida é descrito contendo as seguintes atividades: *provisão de tecnologia; análise de requisitos; projeto do processo; instanciação do processo; simulação do processo; execução do modelo de processo; e avaliação do processo.* A **Provisão de Tecnologia** inclui o fornecimento de tecnologia de suporte à produção de software e de modelos de processo (como as linguagens de modelagem de processo, modelos de processo prontos para reutilização e ferramentas para aquisição, modelagem, análise, projeto, simulação, evolução, execução e monitoração de modelos de processo). A **Análise de Requisitos do Processo** identifica requisitos para o projeto de um novo processo, ou novos requisitos para um processo existente. Os requisitos resultantes especificam os recursos e propriedades que o processo deve oferecer. A fase de **Projeto do Processo** provê a arquitetura geral e detalhada do processo. Nesta etapa,

as linguagens de modelagem do processo são utilizadas. Essa fase pode ser também descrita como uma etapa de modelagem do processo, em que descrições de processos informais são elicitadas e capturadas, sendo convertidas em modelos de processos formais ou instâncias de modelos de processos (SCACCHI e MI, 1997). Apesar de não citado no ciclo de vida de REIS (2003), uma etapa de ***Análise do Processo Modelado*** pode ser conduzida visando avaliar propriedades estáticas e dinâmicas do modelo de processo, incluindo consistência, completude, corretude interna, rastreabilidade e outras verificações semânticas. Também aborda a avaliação de viabilidade e otimização de modelos de processos alternativos (SCACCHI e MI, 1997). A **Instanciação do Processo** modifica a especificação do processo produzida na fase de Projeto do Processo, acrescentando informações detalhadas sobre os prazos, agentes e recursos utilizados por cada atividade definida no processo. A **Simulação do Processo** permite a verificação e validação dos processos definidos antes da execução. A **Execução do Modelo de Processo** utiliza o processo instanciado e o executa através da invocação de ferramentas para guiar e assistir a realização do processo no mundo real. Informações sobre o andamento do processo (*feedback*) são coletadas e analisadas durante a execução. A **Avaliação do Processo** provê informação quantitativa e qualitativa que descreve o desempenho de todo o processo em execução. A avaliação pode ocorrer em paralelo com a execução do modelo de processo e as informações adquiridas podem ser utilizadas nas futuras ocorrências da atividade de análise de requisitos.

A definição de um processo de software de acordo com um conjunto de requisitos se encontra nas fases iniciais do ciclo de vida de um processo. Como resultado, o processo é especificado através de um modelo, uma estrutura complexa, que inter-relaciona diferentes aspectos tecnológicos, organizacionais e sociais para descrever o processo de desenvolvimento de software (REIS, 2002). Esses modelos de processos podem ser entendidos como representações de processos de software em abstrações definidas de acordo com objetivos e interesses dos envolvidos nos processos (STOROLLI *et al.*, 2009). Um modelo especifica como as atividades de engenharia de software serão executadas, quais papéis estão envolvidos, os recursos consumidos, as ferramentas utilizadas, os artefatos consumidos e produzidos pelas tarefas, o produto desenvolvido, assim como os mecanismos de comunicação entre papéis e tarefas (ZAMLI e LEE, 2001).

Alguns objetivos específicos e benefícios da modelagem de processos de software foram apresentados por CURTIS *et al*. (1992): facilidade de entendimento e comunicação; controle e suporte ao gerenciamento de processos; fornecimento de orientações automatizadas para desempenho de processos; fornecimento de suporte para execução automatizada de processos; suporte a melhoria de processos.

Para representar um modelo de processo de software é frequentemente adotada uma Linguagem de Modelagem de Processos (*Process Modeling Language* – PML), a qual deve

oferecer recursos para descrever e manipular os elementos do processo. PMLs devem suportar uma notação visual como forma de suprir a dificuldade de entendimento de representações textuais. Devem possibilitar o entendimento do trabalho de outros papéis envolvidos no processo e o conhecimento do próprio e de outros processos, permitindo um maior apoio a trabalhos colaborativos e um maior suporte ao processo (ZAMLI e LEE, 2001).

Existem várias classificações sobre os principais elementos de um modelo de processo (FEILER e HUMPHREY, 1993; LONCHAMP, 1993). Vários tipos de dados são integrados em um modelo de processo para indicar quem, quando, onde, como e por que os passos são realizados (LONCHAMP, 1993). Apesar de diferentes representações, podemos observar que um conjunto de elementos de processo está sempre presente e pode ser considerado como um conjunto dos elementos essenciais. A Tabela 1 apresenta um resumo dos elementos encontrados das diferentes abordagens e o que representam. O conceito de unidades de trabalho está sendo representado por diferentes granularidades, variando desde um processo como um todo, até o nível de unidade elementar. Esta unidade elementar pode ser uma atividade ou tarefa que não pode ser mais decomposta e é especificada através de um conjunto de passos que descrevem sua forma de execução. Um conjunto de artefatos é requerido para sua execução e um conjunto de artefatos é gerado como resultado. Uma unidade de trabalho elementar exige um conjunto de habilidades, competências e responsabilidades participantes de sua execução que são especificados através de papéis. Um conjunto de recursos é requerido para apoiar sua execução, podendo ser recursos humanos, recursos de hardware, ferramentas que automatizem parte desta execução ou sistemas de apoio. Procedimentos designam como uma unidade de trabalho deve ser executada, através de diferentes tipos de intervenção. Procedimentos podem especificar diretrizes, que apenas visam apoiar e auxiliar uma tarefa, identificando o que deve ser realizado. Diretrizes podem ser consideradas como roteiros ou normas. Métodos são procedimentos mais sistemáticos e especificam os passos que devem ser executados para alcançar resultados, determinando que conjunto de atividades deve compor uma atividade maior. Técnicas descrevem aspectos gerais que devem ser considerados para condução de tarefas ou unidades elementares. Regras e políticas especificam restrições que podem impactar a execução de uma tarefa. *Templates* descrevem estruturas de documentos e artefatos gerados pelas tarefas.

Um conjunto de variações para combinação dos elementos presentes na definição de processos de software é decorrente da necessidade de atender a uma série de características específicas de projetos, produtos e organizações. Características como paradigma de desenvolvimento; o tipo de software a ser desenvolvido; a tecnologia de desenvolvimento a ser adotada; o nível de maturidade da organização; as restrições de prazo, custo e pessoal; as políticas organizacionais, dentre outros, podem impactar na aplicação de procedimentos e

especificações de elementos de processo que acabam por influenciar a atividade de definição de processos de software. Esta condição pode ser denominada de *variabilidade*.

| Elementos de Processo Abordagem | Unidade de Trabalho | Papel | Produto de Trabalho | Recursos | Procedimento |
|---|---|---|---|---|---|
| (ACUÑA e FERRÉ, 2001) | Atividade | Papel | Artefato (entrada/ saída) | Ator (humano/ ferramentas) | Procedimentos, Regras, Políticas, Objetivos |
| (STOROLLI *et al*., 2009) | Processo é composto por Atividades (sequência) compostas por Tarefas | Habilidades | Artefatos (produzidos, modificados e utilizados) | Atores (pessoas, sistemas e equipamentos) utilizam Recursos | Políticas (aspectos de qualidade, regras e critérios); Métricas e Controle (verificação de aplicação das políticas e métricas) |
| (FALBO e BERTOLLO, 2005) | Atividade (Composição: Super/Sub atividades) (Dependências e ordem de execução - sequência: Pré/Pós-atividades) | Agente Humano | Artefatos (Produto/ Insumo) | Agentes: Hardware; Software (Ferramentas – automatiza um procedimento e Sistema de Apoio); Humano | Procedimento: Técnica/Método/Diretriz Adequação a Paradigma e Tecnologia de Desenvolvimento |
| APSEE-Reuse (REIS, 2002) | Atividades | Papéis e Habilidades | Artefatos | Ferramentas | Políticas |
| (LONCHAMP, 1993) | Atividades e Tarefas | Papéis | Artefatos e Entregas de Produtos | Recursos Humanos e Computadorizados | Restrições: regras, procedimentos, políticas ou disciplinas. |
| (LANNA, 2009) | Processos, Atividades e Tarefas | Papéis, habilidades, conhecimento dos membros | Artefatos | Ferramentas e Linguagens | Restrições (custo, tempo, pessoal); políticas e metas |
| SPEM (OMG, 2008) | Processos, Atividades, Tarefa | Papel | Artefatos (produzidos, consumidos ou modificados) | Ferramentas | Procedimentos: *Checklist*, Métricas, Guidelines: técnicas, regras, diretrizes; Relatório (*template* para documentos); Roadmap (método); *Templates* (formatos para artefatos gerados); Especificação de Uso de ferramenta |

**Tabela 1 - Resumo dos elementos de processos presentes em diferentes propostas de representação**

## 2.3. Variabilidade em Processos de Software

Variabilidade é a propriedade de um objeto ser modificado (VAN EIJNDHOVEN *et al*., 2008). Para a área de produtos de software, o termo variabilidade pode ser entendido como a possibilidade de configuração, ou ainda, como a habilidade que um sistema ou artefato de software possui de ser alterado, customizado, ou configurado para um contexto em particular (BOSCH, 2004). Um paralelo pode ser feito para processo de software que possui a capacidade de derivar variações de acordo com as condições inerentes a diferentes requisitos de projetos. Portfólios de processos evoluem devido a fatores internos relacionados à evolução de processos e/ou a fatores externos ou combinações e aquisições em que diferentes processos, podendo ter partes comuns, precisam ser integrados (ROLLAND e NURCAN, 2010). Como todos os componentes de processo (atividades, produtos, agentes, ferramentas) e suas interações (fluxo de informação, fluxo de artefatos, controle, comunicação, tempo, dependências, concorrência) podem variar, processos irão variar, mesmo que estejam no mesmo nível, escopo e objetivo (LINDVALL e RUS, 2000).

Mudanças em processos de negócio envolvem a reutilização de partes de um processo a ser descartado, a inclusão de partes de outros processos, a coexistência de diferentes versões do mesmo processo, etc. (ROLLAND e NURCAN, 2010). Situação similar pode ser observada na

engenharia e manufatura de produtos, onde a noção de linhas de produtos e famílias de produtos foi introduzida. Por analogia, linhas e famílias de processos de negócios podem ser reconhecidas nas organizações atualmente. Observando a dualidade existente entre produtos e processos, famílias de processos podem se beneficiar pela introdução do conceito de variabilidade (ROLLAND e NURCAN, 2010). A variabilidade de processo pode ser entendida como a capacidade de definir, estabelecer o escopo, e executar alterações de uma maneira sistemática e consistente, com o intuito de adaptar modelos de processos a características únicas de um projeto em um dado contexto (MARTÍNEZ-RUIZ *et al.*, 2012).

Alterações em objetivos, estratégias, restrições ou necessidades dos envolvidos podem impactar em mudanças em diferentes abstrações (REGEV *et al.*, 2006). A Figura 2 descreve as mudanças que podem ocorrer no nível da definição do processo ou em uma instância do processo.

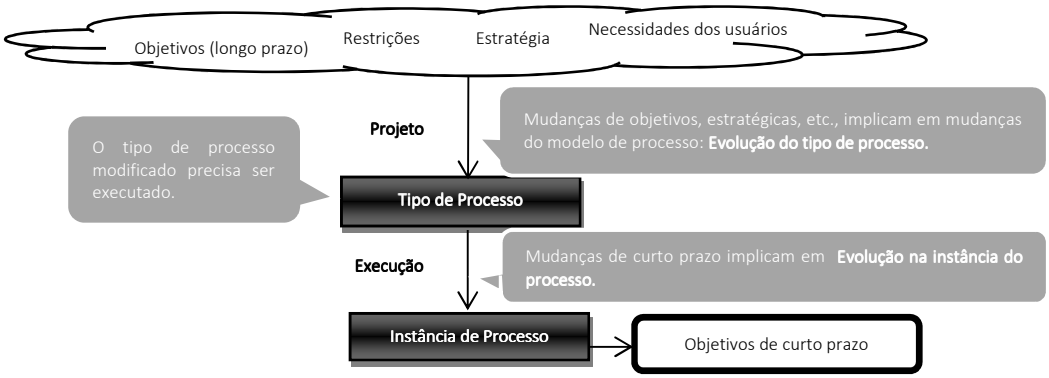



Figura 2 – Nível abstrato de mudanças (Adaptado de: REGEV *et al.*, 2006)

No nível de definição do processo, a mudança representa uma evolução do tipo do processo e reflete em um redesenho de processos. Na mudança de instâncias, há necessidade de tratamento de situações excepcionais de execução do processo. Tais alterações podem ocorrer com modificações sobre um processo existente ou com a criação de um novo processo; podem ser consideradas temporárias ou permanentes; podem ser implantadas em todos os processos imediatamente, incluindo os já instanciados, ou apenas implantadas em situações futuras, em novos projetos (REGEV *et al.*, 2006).

A variabilidade de processos pode ocorrer em duas dimensões: (1) tempo: variabilidade como resposta a mudanças ambientais, e (2) espaço do domínio: variabilidade dentro do espaço de domínio da aplicação em qualquer ponto do tempo, variabilidade como resposta a variação de produto (VERVUURT, 2007).

Um estudo recente (MARTÍNEZ-RUIZ *et al.*, 2012) analisou diferentes trabalhos para identificar o estado da arte das iniciativas de adaptação de processos de software. A análise busca requisitos de variabilidade que são indicados como importantes pelos mecanismos de adaptação. Um conjunto de categorias de elementos de processos foi adotado para analisar quais desses elementos de processos são considerados durante a atividade de adaptação: elementos de trabalho (atividades, tarefas, subprocessos, passos); papéis; artefatos; recursos;

técnicas; e práticas. Os elementos mais apontados como variações são unidades de trabalho,, artefatos e papéis (Figura 3).

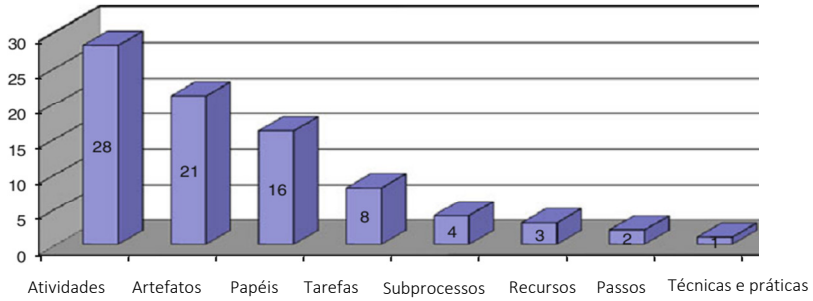



**Figura 3 - Frequência de elementos em que variações são aplicadas (MARTÍNEZ-RUIZ *et al.*, 2012)**

Variações em recursos também devem ser consideradas já que processos são executados por papéis que usam diferentes ferramentas e tais elementos variam de um projeto para outro. Algumas relações foram identificadas caracterizando o impacto de uma variação de um tipo de elemento em outro, cuja consistência é mantida através de relações de variabilidades entre tais elementos, identificando que variações secundárias são geradas a partir de uma variação primária a ser executada:

- Papéis variam no contexto de variação de atividades;
- Tarefas variam no contexto de variação de atividades;
- Passos variam no contexto de variação de tarefas;
- Artefatos variam no contexto de variação de atividades e tarefas;
- Recursos, técnicas e práticas variam no contexto de variação de atividades, tarefas, artefatos e papéis;
- Subprocessos variam no contexto de variação de atividades.

Além de variações em elementos de processos, alguns tipos de variação ocorrem em elementos estruturais, que são relações entre elementos de processos. RAZAVIAN E KHOSRAVI (2008) identificam três tipos de variabilidade:

(1) variabilidade em fluxos de controle: a variação pode ocorrer na forma de diferenciação entre as sequências que são estabelecidas entre as atividades, ou seja, diferentes caminhos de execução de atividades;

(2) variabilidade em fluxos de dados: a variação pode ocorrer como variação nos dados ou variação no objeto de armazenamento dos dados. A variação nos dados significa que os tipos de dados usados podem variar. A variação de objetos de armazenamento de dados indica a variação dos elementos que definem permanentemente o dado armazenado; e

(3) variabilidade em ações: ações podem ser descritas como pedaços de comportamento atômicos que não podem ser decompostos em ações menores. Sua variação pode ser designada pela opcionalidade de participação em um processo ou participação exclusiva em um dado processo.

Algumas operações de adaptação foram identificadas (MARTÍNEZ-RUIZ *et al.*, 2012):

- Operações diretas: (1) *inserção*: adição de elementos de processos; (2) *remoção*: exclusão de elementos de processos; (3) *modificação*: implica na alteração de propriedades dos elementos ao invés de substituí-los; (4) *substituição de elementos*; (5) *alteração de relações entre elementos.*
- Operações indiretas: (1) *padrões*: estruturas de alterações compostas por variações bem definidas aplicáveis na presença de um conjunto de requisitos a ser satisfeito; (2) *parametrização*: atribuição de determinado valor a certos parâmetros previamente definidos de um processo no momento de adaptação; (3) *herança:* adaptação do processo superior através da definição de processos inferiores que estendem as propriedades do processo superior, de acordo com contextos particulares; (4) *encapsulamento de atividades*: definição de grupos de atividades que são tratados em conjunto para realizar a adaptação do processo, ou nós de decisão compostos por condições que são usados para modificar fluxos entre atividades (fluxo de controle) ou modificar o uso de artefatos (fluxo de artefatos).
- Operações baseadas na aplicação de conceitos de Linha de processos: a variabilidade sendo definida como pontos de variação e variantes. A variabilidade pode ser definida em três tipos: (1) *opcional*: elementos que podem ou não estar presentes em processos; (2) *mandatório*: elemento que deve estar presente em um processo; (3) *alternativo*: referente a pontos de variação que possuem diferentes alternativas para ser configurados.

Um paralelo foi realizado para estabelecer relações entre as operações diretas e os tipos de variabilidades descritos acima (Figura 4).

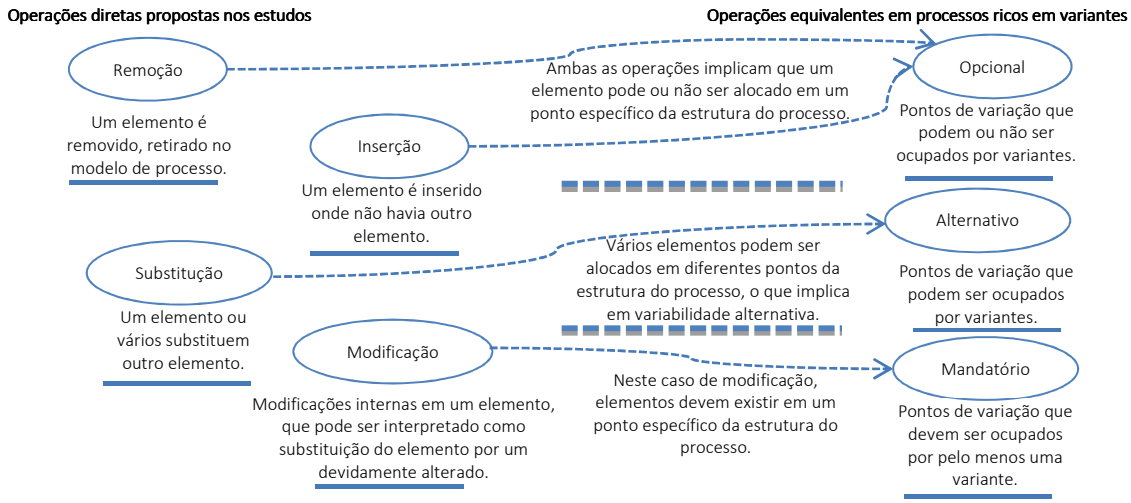



**Figura 4 - Correspondência entre operações de processos e mecanismos de processos ricos em variantes (Adaptado de: (MARTÍNEZ-RUIZ *et al.*, 2012))**

Em Aleixo (2013), uma proposta de categorização de variabilidades de processos é apresentada. Tal categorização divide as variabilidades de processos em duas dimensões: (1)

variabilidades de processos de software no espaço de solução e (2) variabilidades de processos de software no espaço de problema (Figura 5).

| Nível de Abstração | Critério | Classificação |
|---|---|---|
| **Espaço de Solução** | Elementos de processo | (1) papel; (2) atividade, (3) tarefa, (4) artefato e (5) orientação |
| | Granularidade | (6) fina e (7) grossa |
| | Natureza de especificação | (8) pontual e (9) transversal |
| | Tempo de *binding* | (15) modelagem, (16) implantação e (17) execução |
| **Espaço de Problema** | Técnicas e tecnologias associadas à disciplina de processo | (10) requisitos, (11) projeto (*design*), (12) implementação, (13) testes, (14) gerenciamento de projetos |
| | Nível de modelo de maturidade de processos | (20) MPS.Br G, (21) MPS.Br F, (22) MPS.BR E, (23) MPS.Br D (...) |

Figura 5 - Classificação proposta para variabilidades de processos de software (ALEIXO, 2013).

As variabilidades de processos de software no espaço de solução são classificadas de acordo com os seguintes critérios (Figura 5):

(1) *relativas aos elementos de processo* – que especificam e definem de forma explícita quem é de fato o processo, tais como, atividades, tarefas, papéis, práticas, roteiros, etc.. Este tipo de variabilidade identifica se os elementos de processos podem ou não ser incluídos em uma dada instância, através da definição de elementos opcionais ou alternativos;

(2) *relativas à granularidade* – que dizem respeito ao tipo de refinamento de granularidade grossa ou fina que é realizado sobre elementos de processo. Uma variabilidade de granularidade grossa se caracteriza por estar associada a pacotes (conjuntos) de elementos de processos, determinando, por exemplo, a inclusão de elementos de processo associada a possível alteração de elementos já existentes. Uma variabilidade de granularidade fina se caracteriza pela especialização do conteúdo de elementos de processos já existentes no núcleo da referida linha de processos;

(3) *relativas à natureza de especificação* – que indica se a variabilidade ocorre de forma pontual (localizada) ou dispersa (transversal) a um dado elemento de processo; e

(4) *relativas ao tempo de binding* – que indica a se a variabilidade pode ser aplicada ao processo, no momento de sua *modelagem/definição* – durante o início da execução de um projeto, *implantação* – preparação do mesmo para ser implantado em alguma ferramenta que promova o acompanhamento do processo, tais como, as ferramentas de gerenciamento de projeto, gerência de mudanças ou que agregue ambas as funcionalidades; e *execução* – que diz respeito a variabilidades realizadas no processo durante a execução de um projeto.

As variabilidades de processos de software no espaço de problema (Figura 5) indicam situações que podem impactar a seleção dos elementos que farão parte de um processo específico de projeto (modelagem contextual) e são classificadas de acordo com os seguintes critérios: (i) relativas às *técnicas e tecnologias associadas às disciplinas de processo*, tais como, requisitos, projeto, implementação, testes e gerência de projeto; e (ii) relativas aos *níveis de*

*modelos de maturidade de processos* – indica se um dado processo derivado precisa atender a um determinado nível de um modelo de maturidade, como por exemplo, os níveis G ou F do MPS.Br.

A Tabela 2 organiza os diferentes conceitos sobre variabilidade de processos de software discutidos nesta seção.

| Autor | Variabilidade em Processos | |
|---|---|---|
| REGEV *et al*. (2006) | Mudanças em processos no nível de definição. | Evolução do tipo do processo e reflete em um redesenho de processos. |
| | Mudanças em processos no nível de instância. | Necessidade de tratamento de situações excepcionais de execução do processo, podendo ser alterações temporárias ou permanentes. |
| VERVUURT (2007) | Variabilidade no tempo. | Variabilidade como resposta a mudanças ambientais. |
| | Variabilidade no espaço do domínio. | Variabilidade dentro do espaço de domínio da aplicação em qualquer ponto do tempo, variabilidade como resposta a variação de produto. |
| RAZAVIAN e KHOSRAVI (2008) | Variação ocorre em elementos estruturais, que são relações entre elementos de processos. | (1) Variabilidade em fluxos de controle; (2) variabilidade em fluxos de dados; e (3) variabilidade em ações. |
| MARTÍNEZ-RUIZ *et al*. (2012) | Variabilidade em elementos de processos: unidades de trabalho (atividades, tarefas, subprocessos, passos); papéis; artefatos; recursos; técnicas; e práticas. | Operações de variações identificadas: Operações diretas: (1) *inserção*; (2) *remoção*; (3) *modificação*; (4) *substituição de elementos*; e (5) *alterações de relações entre elementos.* Operações indiretas: (1) *padrões*; (2) *parametrização*; (3) *herança*; e (4) *encapsulamento de atividades*. Operações baseadas na aplicação de conceitos de Linha de processos: (1) *opcional*; (2) *mandatório*; e (3) *alternativo*. |
| ALEIXO (2013) | Variabilidades de processos de software no espaço de solução. | Variação em elementos de processos: papel, atividade, tarefa, artefato e orientação. |
| | | Variação relativa à granularidade: fina e grossa. |
| | | Natureza de especificação da variação: pontual ou transversal. |
| | | Tempo de aplicação da variação: modelagem, implantação ou execução. |
| | Variabilidades de processos de software no espaço de problema. | Variação de acordo com técnicas e tecnologias associadas às disciplinas de processo. |
| | | Nível de modelo de maturidade de processo. |

**Tabela 2 - Resumo dos tipos de variabilidades de processos identificadas**

A diversidade de organizações e projetos, suas características e objetivos a tarefa de definição de processos de software torna-se não trivial. Em decorrência do fato de que organizações são diferentes, e ainda que dois projetos dentro da mesma organização também podem ser diferentes (BERGER, 2003), não existe um processo de software que possa ser genericamente aplicado a todos os projetos (MACHADO, 2000). Dependendo das características do projeto, um processo aplicado com sucesso em um projeto pode ser um fracasso em outro (PEDREIRA *et al*., 2007). Essa complexidade da tarefa de definição de processos levou à necessidade de caracterizar, usar e gerenciar as similaridades e diferenças entre os processos (SUTTON e OSTERWEIL, 1996). Neste sentido, diversos autores apontam que a sistematização da reutilização pode ser também aplicada a processos (MONTERO *et al*., 2007).

## 2.4. Reutilização de Processos de Software

Atualmente, muitos processos similares são conduzidos por organizações de desenvolvimento de software, apontando a oportunidade de reutilização da experiência adquirida durante o planejamento e orientação de projetos anteriores, com o objetivo de determinar as melhores políticas a serem adotadas em novos projetos (REIS *et al.*, 2001).

O termo reutilização de processos de software (*Software Process Reuse*) define uma ampla área de estudo e utilização prática relacionada aos diferentes aspectos envolvidos com a reutilização do conhecimento adquirido na condução de projetos de software anteriores (REIS, 2002). Esse tópico recebeu bastante atenção de pesquisadores e da indústria na segunda metade da década de 1990 e, atualmente, mecanismos aperfeiçoados vêm sendo pesquisados e experimentados para o armazenamento e a definição de elementos reutilizáveis de processos (BARRETO, 2007). Essa evolução vem sendo motivada principalmente pela necessidade de novas abordagens no sentido de entender os processos correntes em uma organização, assim como no sentido de promover novas estratégias gerenciais (ELLMER *et al.*, 1996). Desta forma, a reutilização de processos de software vem se destacando como uma das principais práticas para prover a melhoria contínua de processos, por aproveitar as informações (tecnológicas e gerenciais) produzidas durante desenvolvimentos de software passados, com o objetivo de reduzir o esforço necessário para novos desenvolvimentos (BARRETO, 2007). Podemos concluir que as organizações de desenvolvimento de software podem obter expressivas economias, além de permitir um efetivo aumento na qualidade do software produzido (COSTA *et al.*, 2007). De fato, a reutilização de processos de software permite reutilizar o conhecimento que organizações de Engenharia de Software adquirem durante a realização de projetos, prevenindo a ocorrência de erros anteriores e fornecendo melhores práticas (ROUILLÉ *et al.*, 2013).

Além da reutilização de processo ser vista como a reutilização da descrição de processo na criação de outra descrição de processo, é importante considerar a reutilização de artefatos, procedimentos e documentações dentro desses processos (HENNINGER, 1998). Assim, a reutilização de processos de software inclui não apenas a reutilização de definições de processos, mas também a reutilização de informações relacionadas à utilização de processos (conhecimento e experiência adquiridos) (RU-ZHI *et al.*, 2005). No entanto, a criação de um processo capaz de ser utilizado em diversos projetos é uma tarefa difícil. O que é necessário é um método efetivo para capturar os elementos comuns e variantes de processos específicos de projetos e criar definições de processos que possam ser aplicadas em uma variedade de situações, ou seja, que são reutilizáveis (HOLLENBACH e FRAKES, 1996).

Seguindo um breve histórico, a reutilização de processos era, inicialmente, tratada através de um mecanismo de cópia e modificação, sem tratar a relevância do contexto de aplicação. Embora “copiar e modificar” seja importante, pois minimiza o esforço na construção

de um novo modelo, essa prática é menos útil para o aperfeiçoamento da organização e dos seus processos (JØRGENSEN, 2001) *apud* (REIS, 2002), já que, por meio dela, não são registradas informações que permitiriam acompanhar a evolução dos processos e seus componentes.

Aplicando a analogia existente entre produto de software e processo de software (OSTERWEIL, 1987) no campo de reutilização, podemos notar o uso do conhecimento de técnicas de reutilização de produtos de software na área de reutilização de processos (KELLNER, 1996), tais como: arquiteturas povoadas com componentes reusáveis; gerenciamento de repositórios para armazenar, catalogar, procurar, acessar, etc. ativos reusáveis; e gerência de configuração de ativos reutilizáveis. Assim, destacam-se algumas abordagens como técnicas de reutilização que seguem a linha de padrões, arquiteturas e *templates* de processo, técnicas que trabalham o conceito de componentização de processos.

Arquitetura de processos pode ser entendida como um conjunto padrão de unidades ou passos de processo principais com regras que os descrevem e relacionam (HUMPHREY, 1989). Já um *template* de processo consiste em um modelo de processo genérico e reutilizável que estabelece um ponto de início para a construção de um novo modelo de processo (REIS, 2002). Um padrão de processo descreve uma abordagem ou série de ações bem sucedidas para o desenvolvimento de software (AMBLER, 1998).

Um componente de processo pode ser visto como um encapsulamento de informações e comportamento de processo em um dado nível de granularidade (GARY e LINDQUIST, 1999). De forma similar, mas com uma nomenclatura diferente, componentes de processo podem ser definidos como elementos de processos. Um elemento de processo é um grupo de atividades de projeto e/ou outros elementos de processos relacionados por dependências lógicas, que quando executados fornecem valor a um projeto (BHUTA *et al.*, 2005). Existem diferentes abordagens que trabalham com o conceito de componentes de processos (RU-ZHI *et al.,* 2005, FUSARO *et al.,* 1998, BHUTA *et al.*, 2005). No entanto, para que possam ser definidos processos de software baseado nesses elementos, deve existir um mecanismo adequado para avaliação dos componentes quanto à sua aplicação no contexto específico (BASILI e ROMBACH, 1987), de forma a atender aos requisitos estabelecidos a um determinado processo (RU-ZHI *et al.,* 2005). FUSARO *et al*. (1998) levantam os seguintes problemas relacionados a definição de processos a partir de componentes de processo pré-existentes: (i) É necessário que existam componentes de processo descritos com o mesmo formalismo utilizado na descrição dos processos; (ii) Deve ser possível integrar os componentes de processo aos processos; e (iii) Deve existir um mecanismo para seleção e escolha do componente mais adequado, quando existir mais de um componente com o mesmo propósito.

Outra abordagem, denominada Linha de Processos de Software (LPrS), surgiu como uma técnica sistemática de reutilização de processos e foi apresentada em alguns trabalhos (BARRETO, 2007, JAUFMAN e MÜNCH, 2005, ROMBACH, 2006, WASHIZAKI, 2006a) que aplicam a ideia de LPS em processos. Uma LPS pode ser definida como um conjunto de sistemas de software que compartilham um conjunto de características comuns e controladas, que satisfazem necessidades de um segmento de mercado em particular, e são desenvolvidos a partir de artefatos (*core assets*), de forma predefinida (NORTHROP, 2002). Tal abordagem incorpora uma fase de captura das informações do domínio e sua representação através de um modelo de domínio. O objetivo de tal modelo é explicitar as similaridades e diferenças presentes no domínio, i.e., as suas variabilidades e opcionalidades. A modelagem de características é uma das maneiras mais utilizadas na área para representar tais conhecimentos. Tal modelo é considerado o modelo de mais alto nível de abstração, bem como o ponto de partida para a reutilização de artefatos em um processo de ED ou LPS (OLIVEIRA, 2006). Uma característica pode ser entendida como um aspecto, uma qualidade, uma característica visível ao usuário, proeminente ou distinta, de um sistema (ou sistemas) de software (KANG *et al*., 2002).

WASHIZAKI (2006a) define uma Linha de Processo como "um conjunto de processos de um determinado domínio de problema, ou com um determinado propósito, que têm características em comum e é construído baseado em ativos reutilizáveis de processos". Alguns objetivos podem ser apontados através do uso de Linha de Processos: aumento da produtividade da atividade de definição de processos, diminuindo o esforço necessário para realizá-la; aumento da qualidade e da adequação dos processos gerados, através da reutilização do conhecimento de especialistas e de dados sobre utilização; aumento do potencial de reutilização através da representação de variabilidades; e redução dos riscos de uma definição inadequada de processo (BARRETO *et al.,* 2009, JAUFMAN e MÜNCH, 2005, ROMBACH, 2006).

Com base em trabalhos na área de reutilização de software (BLOIS, 2006) que estudam a oportunidade de promover a reutilização de artefatos através da combinação das áreas de Engenharia de Domínio (ED) (ARANGO e PRIETO-DIAZ, 1991) e Desenvolvimento Baseado em Componentes (DBC) (SAMETINGER, 1997), e aplicando a analogia entre software e processos de software (OSTERWEIL, 1987), pode-se identificar a possibilidade da mesma oportunidade no campo de processos de software. Se por um lado a ED visa identificar aspectos variáveis e invariáveis de um domínio para a sua reutilização, por outro, o DBC propõe soluções reutilizáveis para o processo de desenvolvimento de software, disponibilizando aos usuários tais variabilidades através de componentes e interfaces (BROWN, 2000). Neste sentido, um desafio da área de ED é promover o uso de abordagens de DBC, visando à construção de Arquiteturas de Referência (ARs) baseadas em componentes, e como desafio da área de DBC, podemos citar

a modelagem de variabilidades através de componentes, representando de forma consistente tais aspectos de domínios (BLOIS, 2006). Transferindo para a área de processos de software, identifica-se os conceitos de Engenharia de Domínio aplicados em abordagens de LPrS e os conceitos de DBC em abordagens que visam realizar a DPBC. Desta forma, uma análise dos trabalhos relacionados a tais áreas foi realizada e é apresentada no próximo capítulo.

# CAPÍTULO III

# TRABALHOS RELACIONADOS

## 3.1. Introdução

Diante da complexidade envolvida na tarefa de definição de processos de software, técnicas de reutilização de processos de software têm sido propostas como forma de reduzir o esforço e tempo despendido, além de prover melhorias ao aproveitar o conhecimento adquirido em projetos anteriores através de informações (tecnológicas e gerenciais) produzidas durante desenvolvimentos de software passados.

Assim, o objetivo deste capítulo é apresentar uma análise dos trabalhos propostos na área de reutilização de processos de software com foco em abordagens de Linha de Processos de Software (LPrS). Um conjunto de critérios foi definido para análise dos trabalhos relacionados, buscando identificar as contribuições e pontos de melhorias das abordagens propostas. A Seção 3.2 enumera os critérios utilizados na análise dos trabalhos relacionados e o procedimento de identificação dos mesmos. A Seção 3.3 apresenta os resultados gerais obtidos através da revisão da literatura das abordagens de famílias de processos de software identificadas. A Seção 3.4 foca em uma análise de conceitos e técnicas para a modelagem de variabilidade de processos de software. A Seção 3.5 apresenta uma análise inicial dos trabalhos relacionados à área de Componentes de Processos de Software e Arquitetura de Família de Processos de Software. Por fim, na Seção 3.6, são feitas algumas considerações a partir dos conceitos e análises observados.

## 3.2. Procedimento de Identificação de Trabalhos Relacionados e Critérios de Análise de Abordagens Selecionadas.

As áreas de Linha de Processos de Software (LPrS) e de Definição de Processo de Software Baseada em Componentes (DPBC) servem de base para a construção da sistemática de reutilização de processos de software proposta nesta pesquisa. Desta forma, o objetivo da revisão é analisar publicações científicas sobre a definição de processos de software, com o propósito de caracterizar abordagens (ex.: técnicas, métodos, processos, ferramentas), com relação à utilização de técnicas de LPrS combinadas com conceitos de DPBC, do ponto de vista de pesquisadores, no contexto industrial e acadêmico. Para uma análise controlada, um conjunto de critérios foi definido como forma de auxiliar na identificação dos principais tópicos envolvidos nessas áreas.

Dentro da área de reutilização de software, o arcabouço de comparação utilizado no estudo de viabilidade do método FODA (Kang *et al.* 1990) apresenta três principais aspectos que, segundo os autores, devem ser considerados na avaliação de um método: a) *a existência de um processo*; b) *os produtos gerados pelo método*, como são representados e

aplicáveis ao desenvolvimento de um domínio de aplicações e; c) *as ferramentas para automatizar o processo*. Assim, um aspecto a ser observado nos trabalhos consistiu na identificação da existência de processos que especifiquem como conduzir a técnica proposta e a especificação dos produtos a serem gerados pela técnica. Tal sistemática pode ser alcançada mediante a definição de estratégias de definição de **processos *para* e *com* reutilização**. Essas estratégias devem definir o conjunto de atividades e resultados esperados que viabilizem uma reutilização efetiva. Assim, um dos critérios de análise consistiu em analisar a existência de uma especificação de um processo de desenvolvimento *para* reutilização de processos de software, definindo as etapas envolvidas na construção de artefatos de processos de software reutilizáveis, e de um processo de desenvolvimento *com* reutilização definindo as etapas envolvidas na descrição de processos específicos de projetos, que utilizam os artefatos de processo reutilizáveis gerados.

Um segundo critério definido consistiu na identificação da existência de uma proposta de **representação para modelagem de variabilidades e opcionalidades de uma LPrS**. Os elementos de processos classificados como variáveis e os tipos de variações permitidas são aspectos importantes a serem observados. Além disso, a definição de uma notação para representação da LPrS deve ser analisada. A possibilidade do uso de diferentes níveis de abstração para representar a LPrS, como ocorre em abordagens de reutilização de software, deve ser observada em conjunto com a identificação de procedimentos de mapeamento entre os diferentes artefatos utilizados para manutenção da consistência dos modelos e dos processos a serem derivados a partir da linha definida. Essa análise foi complementada com critérios específicos de modelagem de variabilidades em LPrS apresentados na Seção 3.4.

Outra área de reutilização de processos de software relevante consiste na DPBC. Desta forma, a **análise do uso de princípios de DPBC** nas abordagens analisadas visa identificar o quanto de componentização tem sido trabalhado de forma conjunta a técnica de LPrS, como forma de promover um aumento dos benefícios para a reutilização. Além disso, métodos de desenvolvimento de LPrS e DPBC devem fornecer sistemáticas para a organização de seus artefatos em elementos arquiteturais, os quais compõem uma arquitetura de componentes de LPrS. Para isso, é esperado o uso de critérios, métricas ou outras técnicas para apoio a criação destes elementos arquiteturais.

Por último, uma análise de **ferramentas de apoio as diferentes etapas da abordagem** deve ser realizada. Além disso, a análise da existência de uma infraestrutura de reutilização de apoio integrado as diferentes etapas de construção e uso de uma LPrS é importante, uma vez que, do ponto de vista de reutilização, os artefatos gerados em todas as fases do processo estão relacionados. Assim, caso não exista integração entre os diferentes apoios ferramentais, a

reutilização dos artefatos poderá ser prejudicada, limitando-se a alguma fase ou atividade específica de um processo.

Tal revisão foi iniciada através da união dos resultados de duas revisões sistemáticas recentemente conduzidas em teses de doutorado (ALEIXO, 2013; BARRETO, 2011) que tratam do tema de reutilização de processos de software. A revisão de ALEIXO (2013) teve como objetivo identificar as abordagens existentes de gerência de variabilidades de processos de software, indicando que técnicas elas utilizam, de que forma vem sendo avaliadas e com que tipos de exemplos de variabilidades tais estudos são realizados. A revisão de BARRETO (2011) teve como objetivo **analisar** relatos de experiência e publicações científicas sobre definição de processos de software, com o **propósito** de caracterizar abordagens (ex.: técnicas, métodos, processos, ferramentas), **com relação** à utilização de técnicas de reutilização, do **ponto de vista** de pesquisadores, no **contexto** industrial e acadêmico. A segunda revisão é mais abrangente do que o escopo estipulado neste trabalho, incluindo outras técnicas de reutilização de software, como padrões de processos de software. Desta forma, um filtro foi aplicado aos artigos desta segunda revisão focando nos trabalhos identificados como técnicas de Linhas de Processos e Famílias de Processos, trabalhos relacionados ao escopo desta pesquisa.

Ambas as revisões de base possuíam critérios focados no contexto da tese em que estavam sendo desenvolvidas e, desta forma, precisou-se estender esses critérios de acordo com os objetivos desta pesquisa. Em uma etapa inicial, um conjunto de artigos foi incorporado à revisão, partindo dos artigos identificados nas revisões anteriores. O resultado desta análise preliminar é apresentado na Seção 3.3.

Através de um trabalho de parceria com uma equipe de pesquisados do Rio Grande do Norte, essa análise está sendo estendida e seus resultados serão publicados ao longo do desenvolvimento da tese.

## 3.3. Revisão da Literatura - Abordagens de LPrS

SUTTON e OSTERWEIL (1996) discutem a noção de uma família de processos implícita na noção de uma família de produtos. Os autores buscam correlacionar variações de produtos de software com variações nos processos de desenvolvimento utilizados para construí-los. Assim, um grupo de produtos relacionados, mas diferenciáveis, pode ser visto como o resultado de um conjunto de processos relacionados, mas diferenciáveis, isto é, uma *família de processos*. O conceito de família aplicável a produtos poderia ser útil para organizar e gerenciar processos, desde a especificação de requisitos, projetos e especificações a execuções, manutenção, evolução, melhoria e reutilização. A variação entre processos de uma família pode refletir nos passos que compõem os processos, no sequenciamento ou concorrência na execução dos passos, nos componentes de produtos a serem produzidos, nas restrições de consistência

estabelecidas, no tamanho e estrutura da equipe, na natureza dos recursos usados e em outros fatores. As diferenças entre os membros de uma família de processo podem ser causadas por uma variedade de considerações, tanto relacionadas ao produto como a outros critérios, tais como tamanho e complexidade do produto, expectativas com manutenção, evolução e reutilização, e a natureza do projeto de desenvolvimento ou da organização. Os autores ainda mencionam que se um processo é visto como uma composição de subprocessos, então existe a possibilidade de reutilização de componentes de processos dentro e entre famílias de processos. Conclui-se que o conceito de família de processos apresenta considerável complexidade no tratamento da flexibilidade e adaptabilidade de processos, principalmente relacionada à caracterização, utilização e gerenciamento das similaridades e diferenças entre as famílias de processos e os membros dessas famílias.

DURÁN *et al*. (2004) propõem a aplicação da abordagem de família de processos como um meio para definir um processo central que pode ser adaptado a necessidades específicas das companhias envolvidas de uma maneira controlada e pré-definida através do uso de identificação de variabilidades. As variações são especificadas quanto às técnicas possíveis de execução de certas unidades de trabalho; aos papéis envolvidos; aos *templates* e documentações e a descrição de alternativas de passos das atividades. Utiliza descrições adicionais em cenários para representar as variações de execução dos passos e diagramas de classes UML (herança) para expressar as variabilidades de características das atividades, papéis e documentações. O trabalho não se preocupa em especificar o método para utilização da abordagem, apenas apresenta de forma descritiva as atividades que foram conduzidas para aplicação da técnica em um exemplo de um processo da área de Engenharia de Requisitos. Não trata o conceito de Arquitetura de LPrS, nem organiza os elementos em unidades de composição (componentes de processo de software). Não apresenta nenhum ferramental de apoio à abordagem.

JAUFMAN E MÜNCH (2005) propõem um método que usa uma Linha de Processo específica de domínio para adaptar e refinar um processo, inicialmente derivado baseado nas atividades executadas em uma primeira iteração. O método consiste em dois passos principais: (1) uma linha de processos específica de domínio é usada para a adaptação *top-down* realizada no início do projeto, com o propósito de fornecer conhecimento sobre o domínio necessário para definir um processo de software adequado; (2) após o primeiro ciclo de desenvolvimento, o processo definido é revisado com base nos dados coletados de sua execução. A ideia por trás de Linha de Processos é capturar as similaridades em blocos de construção de processos reutilizáveis e construir variantes de processos explícitas, baseadas nos desvios de processo, e reutilizar os blocos de construção quando aplicáveis. A técnica apresentada foca na fase de derivação de um processo específico de projeto com apoio a adaptabilidade e alterações

sucessivas para customização do processo de acordo com objetivos e características de contexto. Um ferramental de apoio é mencionado mas não é de fato apresentado pelos autores.

ROMBACH (2006) (2013) aponta que processos de software ainda que variáveis entre projetos não são gerenciados de maneira sistemática, como artefatos de software em LPSs. O autor defende a ideia de estabelecer Linhas de Produtos e Processos de Software integradas, com o intuito de selecionar sistematicamente tanto os artefatos quanto os processos necessários para um dado projeto. Quando um processo padrão é adotado, sua descrição altamente genérica só permite sua aplicação em projetos através de adaptações. No entanto, não é possível especificar os rastros entre os pontos comuns e as variabilidades controladas entre as diferentes instâncias de processos. Desta forma, um processo de engenharia do domínio deve suportar a criação de um (conjunto de) processo(s) genérico(s) capaz(es) de capturar as similaridades e as variabilidade controladas do domínio. Define como características principais de uma Engenharia de Linha de Processos de Software (ELPrS): (i) Dois processos de desenvolvimento separados: o **processo de Engenharia de Domínio**, pelo qual um (conjunto de) processo(s) genérico(s) para reutilização é criado para capturar as semelhanças e variabilidades controladas em um domínio, e o **processo de Engenharia de Aplicação**, pelo qual processos específicos de um projeto estão sendo desenvolvidos; (ii) Um **repositório** para disponibilização de processos reutilizáveis em todos os níveis de abstração; (iii) Um **processo sistemático de reutilização**, onde para cada escolha pré-definida de variabilidades, a escolha dos componentes de processo é pré-definida; e (iv) Um **processo sistemático de gerenciamento de processos**, onde para cada exceção (e.g., um comportamento inesperado do processo ocorre) será decidido se a exceção será levada em conta no processo genérico ou não. Os objetivos para o uso desta engenharia consistem no aumento de previsibilidade e redução de custo, tempo e risco. O processo de ED descrito pode ser apoiado por abordagens *top-down*, que refletem a típica padronização de processos, e abordagens *bottom-up*, que analisam diferentes projetos, em busca de pontos similares e variáveis a serem modelados. Essas variabilidades de processos são definidas como objetivos e requisitos de produtos ou processos, além de características de projeto. O autor ainda aponta algumas tarefas de pesquisa importantes: o projeto de linguagens de modelagem de processos com recursos para a especificação de variabilidades e o desenvolvimento de fundamentos teóricos e de engenharia para as linhas de processo.

WASHIZAKI (2006a) (2006b) define Linha de Processos como um conjunto de processos similares dentro de um domínio em particular ou para um propósito em particular, que possui características comuns e é construída baseada em artefatos de processos comuns e reutilizáveis. De forma similar a ELP, a ELPr é dividida em duas fases: (i) ED, que corresponde à coleta de similaridades e variabilidades de requisitos da linha de processos, utilizados no projeto

e na implementação da Arquitetura da linha e suas variantes; e (ii) EA, em que um processo específico de projeto é construído através da Arquitetura da linha e suas variantes. A Arquitetura de LPr é definida como uma estrutura de processo que reflete a similaridade e variabilidade de uma coleção de processos. A abordagem descreve os passos da etapa de ED. A descrição do método de EA se restringe a especificar a necessidade de recorte do modelo de características associado à Arquitetura de LPr. A técnica consiste de quatro etapas: (1) obtenção de uma linha de processos através da coleta de processos similares; (2) análise de similaridades e variabilidades; (3) estabelecimento de rastros entre características de projeto (na forma de requisitos da linha de processos) e elementos de uma arquitetura de linha de processos obtida; e (4) definição de processos específicos de projetos consistentes. O procedimento para construção da Arquitetura da LPr consiste em analisar elemento por elemento dos processos que compõem o domínio frente ao menor processo definido como principal. A análise de similaridades e variabilidades é realizada manualmente comparando detalhes das atividades, pré/pós-condições, entradas e saídas, papéis e ambientes, incluindo ferramentas. A classificação de variabilidade é realizada sob a análise de relacionamentos de generalização/especialização entre os elementos. A classificação de opcionalidade não é ortogonal a classificação de variabilidade, onde pontos de variação acabam sendo considerados mandatórios e fazem parte do processo principal. Relações de dependência podem ser expressas através do estereótipo <<*requires*>> entre dois elementos. E condições de escolha entre caminhos alternativos devem ser representadas pelo estereótipo {*xor*}. A técnica apresentada inclui algumas extensões ao SPEM v1.1 para expressar similaridades e variabilidades em fluxos de processos, expressos através de diagramas de atividades. O apoio à construção do diagrama de características é limitado e a manutenção da consistência entre os modelos não é garantida. O método para criação de relações de impacto entre os modelos não é definido.

SIMIDCHIEVA *et al*. (2007) tratam variações de processos através da especificação de mudanças de comportamentos dos agentes associados a tarefas; descrição detalhada de tarefas descritas em níveis mais abstratos e alterações na estrutura dos artefatos. Usam a linguagem de modelagem Little-JIL e uma base de ativos para descrever as variantes. A técnica de seleção de variantes mais apropriadas é usada de forma manual pelo usuário da abordagem. O trabalho aponta a necessidade de automação para apoio a essa atividade. Outro ponto importante constitui a definição do que seria de fato uma variante de um processo base. Aponta o desenvolvimento de métricas como um meio para definir o grau de similaridade entre os processos e definir o que seria uma variante viável de um processo base.

HALLERBACH *et al*. (2008) apresentam a abordagem *Provop* para gerenciar variantes de processos. A representação é realizada através da definição de operações para adaptação de

um modelo de processo base e a especificação de situações de contexto para aplicação de tais operações. Um dos desafios apontado como trabalho futuro é o tratamento da combinação das operações que podem ser redundantes ou provocar conflitos, com o propósito de evitar erros ou outros problemas de consistência. O método de utilização da abordagem descreve as atividade principais para modelagem das variações de um processo base e das etapas envolvidas na configuração de um processo utilizando a abordagem.

ARMBRUST *et al*. (2008) (2009) trabalham o conceito de LPrS como sendo a transferência dos conceitos de LPS para processos de software. Neste trabalho, uma LPrS pode ser definida como um conjunto de processos de software com um conjunto de características que satisfazem necessidades específicas de uma organização em particular e que são desenvolvidas a partir de um conjunto comum de processos, de acordo com uma forma prescrita. A abordagem descreve um conjunto de atividade para construção e uso da LPrS. A atividade inicial consiste na Definição do Escopo de uma LPrS, foco do trabalho. Tal atividade consiste em determinar os membros da LPrS. Durante este passo, os produtos e projetos correntes e futuros da organização são analisados para elicitar as necessidades de processos da organização (escopo da LPrS). A abordagem SCOPE (ARMBRUST, 2010) é apresentada com maiores detalhes visando identificar as demandas de processo de produtos e projetos correntes e futuros, análise dos processos existentes para seleção do grupo de processo com melhor benefício para apoio a essas demandas da organização. A **Engenharia de Domínio de Processo** constrói um repositório de processo contendo todas as partes de processo estáveis e variáveis, assim como o modelo de decisão que governa quando usar cada variante. Esta atividade pode ser dividida em: (1) Modelagem da LPrS: Análise dos processos que formam o escopo da organização em termos de suas similaridades e variabilidades. Baseada nesta análise, uma coleção de artefatos genéricos ("blocos de construção") capturando as similaridades e variabilidades identificadas é construída e um modelo de decisão, incluindo todos os pontos de variação, é especificado, o que permite a derivação de um modelo de processo específico a partir dos artefatos construídos. As variações são modeladas usando a ferramenta gráfica de modelagem de processos de software SPEARMINT™ (ARMBRUST *et al*., 2009). Pontos de variação representam uma decisão que precisa ser realizada ao escolher entre diversas alternativas. Enquanto alguns atributos analisados não provocam alterações nos processos outros provocam. As partes de processos opcionais são marcadas como uma descrição detalhada de quando devem ser consideradas através de regras, e (2) Arquitetura de LPrS: criação de uma arquitetura de referência de processo, ou seja, um modelo de processo contendo todos os artefatos genéricos e o modelo de decisão governando quais artefatos devem ser organizados conjuntamente sob determinadas circunstâncias, formando uma instância de processo específico. A **Instanciação de LPrS** extrai do repositório de processo uma

instância de processo específica sem variabilidade para cada projeto que pode ser posteriormente adaptada durante a customização. Não apresenta ferramental de apoio às etapas da abordagem. A abordagem não discute o conceito de componentes de processos e o agrupamento dos elementos identificados.

A abordagem proposta por TERNITÉ (2009) apresenta conceitos da técnica de reutilização baseada em LPrS. A definição de variabilidades é feita através de operações de adaptação: adição/remoção de elementos opcionais, e operações de mudanças pré-estabelecidas em um metamodelo que define os possíveis elementos e as operações passíveis de realização. Neste trabalho, uma LPrS é definida como um conjunto de processos que captura as similaridades e variabilidades controladas. Cada um desses processos é desenvolvido a partir de um conjunto comum de artefatos principais (características) de uma maneira prescritiva. Cada característica pode ser entendida como um elemento do modelo (elemento de processo ou relação). Os tipos de características são: mandatória (característica deve sempre ser incluída); opcional (características adicionais – um complemento independente que pode ou não ser incluído) e alternativa (características para substituição de outras). As características podem ser: (1) independentes: a seleção de uma característica não afeta a seleção de outra; (2) inclusivas: a seleção de uma característica implica na seleção de outra e (3) mutualmente exclusivas: características não podem ser selecionadas em conjunto. Os tipos de variabilidades definidos como positiva (adição de elemento de processo ou relações), negativa (elementos ou relações são removidos), extensão (elementos ou relações são estendidos) e substituição (elementos ou relações são substituídos), indicam as operações e adaptações possíveis que devem ser configuradas para os casos particulares. O conceito de arquitetura é definido como arcabouço arquitetural que prove mecanismos para variações controladas. A arquitetura deve permitir a configuração de características mandatórias, opcionais e alternativas que implicam os efeitos definidos por tipos de variabilidades positiva, negativa, extensão e substituição. A abordagem não discute o conceito de componentes de processos e o agrupamento dos elementos identificados.

SIMMONDS *et al.* (2011) e SIMMONDS e BASTARRICA (2011) apresentam uma análise da aplicabilidade de formalismos de modelagem para representar a variabilidade de processos de software identificada em LPrS. A análise observa aspectos do meta-modelo SPEM 2.0 e de modelagens de variabilidades em processos de software e para LPS (modelagem de características e modelagem ortogonal de variabilidades). De acordo com a experiência "limitada" dos autores, a total expressividade dos formalismos de modelagem de variabilidades não é necessária para aplicação em processos de software que parecem variar menos do que produtos de software (SIMMONDS *et al.*, 2011). Assim, a aplicabilidade de determinado formalismo deve ser analisada pela disponibilidade de ferramenta. SIMMONDS e BASTARRICA

(2011) e SIMMONDS *et al*. (2012) descrevem a criação de uma família de processos através da identificação dos aspectos comuns desses processos, assim como as possibilidades de variações desses processos de acordo com o tipo de projeto sendo desenvolvido. Portanto, uma LPrS (ou família) consiste de dois itens: (1) um modelo geral de processo e (2) especificação de quais elementos de processos variam e como podem variar. Os autores colocam que para uso de tal abordagem, a relação custo-benefício só é válida para casos em que a abordagem seja automatizada e isso só é possível com o uso de modelos que são formalmente especificados. A abordagem é dividida em um processo geral e sua variabilidade modelada separadamente. Tais modelos são ligados através de restrições de integridade que permitem a análise de consistência e são especificados pelo engenheiro de processo. O fato de tais regras serem especificadas, manualmente, através da intervenção do engenheiro de processo, pode levar a inclusão de erros ou não abrangência na definição das regras necessárias para manter a consistência entre os modelos. Os autores apontam como trabalho futuro a definição automática das regras. O resultado final consiste na base para especificação formal de uma LPrS que poderá ser adaptada para diferentes contextos. A variação de processos identificada pela abordagem trata de elementos classificados como opcionais ou alternativos. Os tipos das variantes associadas a um ponto de variação devem ser do mesmo tipo do ponto de variação. Como forma ilustrativa da abordagem, a variabilidade do processo geral foi demarcada manualmente. O modelo de processo geral utiliza a notação SPEM 2.0 através da ferramenta EPFC (*Eclipse Process Framework Composer*). A variabilidade é representada através de um modelo de características através da ferramenta SPLOT. Para lidar com questões de inconsistências entre os dois modelos, um terceiro modelo estabelece restrições de consistência com a ferramenta Modisco/AMW. Para a representação de variabilidade nenhum suporte gráfico foi demonstrado.

BARRETO *et al*. (2007)(2009)(2010)(2011) propõem uma abordagem para reutilizar processo de software através de componentes de processos. Neste contexto, um componente de trabalho é definido como uma unidade básica de composição de processos, considerada relevante para: (i) ser reutilizada em outras definições de processos; (ii) ter sua estabilidade e desempenho analisados; (iii) ser versionada; e (iv) ter vários tipos de rastreabilidade estabelecidos; entre outros. Uma Linha de Processos de Software é definida como uma arquitetura de processos que possui variabilidades ou opcionalidades (BARRETO *et al.*, 2009). Uma arquitetura de processos é similar a fluxos de trabalho, e pode ser composta por componentes de processo, atividades ou qualquer combinação entre eles. Define um "esqueleto" que o processo deve possuir, determinando os principais elementos e como estes se relacionam, sem necessariamente definir como será o detalhamento desses elementos principais. Este trabalho usa o conceito de características de processos como um aspecto,

qualidade ou característica que um processo precisa ser compatível (BARRETO *et al.*, 2010). Características de processo são usadas apenas como um mecanismo de alto nível para seleção de componentes. O autor menciona que poucos mecanismos de relacionamento entre características foram utilizados pela abordagem e que expressões lógicas podem ser usadas para aumentar a expressividade de representação. Desta forma, a modelagem de variabilidades de uma LPrS é realizada através de um modelo de componentes que descreve componentes de processo mandatórios e opcionais, além de representar pontos de variação e variantes. A variabilidade também é trabalhada com a definição de dois tipos de componentes: componentes concretos, que não permitem mais configurações, e componentes abstratos, com definições incompletas que permitem configurações através da implementação por componentes concretos. A visualização da LPrS e dos componentes deveria permitir identificar mais facilmente a estrutura interna de um componente e as variantes de um ponto de variação. A abordagem ainda inclui estratégias de definição de processos *para* e *com* reutilização. No entanto, não descreve procedimentos para organizar elementos de processos em componentes de processo. O apoio ferramental é provido para algumas etapas da abordagem.

Uma abordagem anotativa foi utilizada para gerência de variabilidades em processos de software e derivação automática de processos, oriundos da customização automática de especificações de famílias de processos de software em ALEIXO *et al*. (2010) (2012). A abordagem parte da modelagem e definição de uma LPrS, através de ferramentas existentes de especificação de processos. Em seguida, concentra-se na gerência automatizada de variabilidades dos elementos da linha de processos sendo modelada, através de especificação dos modelos de variabilidades (modelo de características, modelo de configuração e modelo de processo). Na etapa seguinte, com o auxílio de uma ferramenta de derivação, são selecionadas as características relevantes para um processo, possibilitando a geração automática de processos de software customizados que lidem com cenários e necessidades específicas de um dado projeto. A abordagem também visa especificar o processo customizado como *workflow* para execução em máquinas de *workflow*. A abordagem não discute o conceito de componentes de processos e o agrupamento dos elementos identificados. A etapa de modelagem é realizada usando o EPF (*Eclipse Process Framework*). A etapa de gerenciamento de variabilidade é realizada usando uma extensão da ferramenta GenArch (derivação de produtos).

Em ALEGRÍA *et al.* (2013), uma abordagem baseada em modelo para adaptação de processos de software é apresentada para suportar a geração automática de processos específicos de projetos com base em processos organizacionais e contextos de projetos. A abordagem CASPER (ALEGRÍA *et al*., 2011; ALEGRÍA E BASTARRICA, 2012; ALEGRÍA *et al.,* 2013) visa definir e usar LPrS como apoio à atividade de definição de modelos de processos de

software adaptáveis. A abordagem propõe desenhar cenários através de contextos e criar regras de transformação baseadas nas relações entre os atributos das situações de contexto definidas e os elementos de processos e suas variações modeladas através de um modelo de características e um modelo de processo usando o meta-modelo SPEM 2.0. O foco encontra-se na fase de ELPrS, através de um conjunto de atividades que se inicia com a descrição do contexto por meio de uma linguagem própria definida pelos autores, seguida da definição das características de processos e a definição das relações entre os dois modelos. Descreve o conceito de Arquitetura seguindo a estrutura provida pela SPEM 2.0. A proposta não menciona o tratamento da variabilidade no nível comportamental, tratando das variações em termos estruturais (elementos) e utilizando fluxos de processos convencionais para descrever os diagramas de atividades do modelo de projeto da LPrS. A abordagem não discute o conceito de componentes de processos e o agrupamento dos elementos identificados. Falta um ferramental de apoio dedicado a todas as etapas como uma infraestrutura de reutilização, apenas ferramentas isoladas já existentes são destinadas a algumas atividades de modelagem.

MAGDALENO *et al*. (2012) apresentam a abordagem COMPOOTIM para apoiar decisões de gerentes de projetos na seleção e combinação de componentes de processos para derivação de um processo específico de projeto. Para uma reutilização e composição de processo sistemática, uma LPrS baseada em contexto (NUNES *et al*., 2010) foi definida como um conjunto de componentes de processo, organizados para representar partes variáveis e comuns dentro de um domínio específico que podem ser reutilizados e combinados, de acordo com regras de composição, para compor e adaptar dinamicamente abordagens de processos que apoiam decisões de gerentes de projetos sobre a seleção e composição de componentes de processo e otimiza a sugestão de processos para um contexto de projeto particular. O foco do trabalho está na fase de derivação de um processo para um projeto específico (MAGDALENO, 2013). Usando a abordagem contextual, a abordagem é capaz de oferecer informações adicionais a LPrS para apoiar a decisão do gerente de projeto. Para isso, informações de contexto de projetos de software (por exemplo, informações sobre a estrutura organizacional; informações de projeto: tamanho, complexidade, duração, estabilidade; informações sobre equipe: experiência, proximidade geográfica, etc.) devem ser identificadas e registradas. O uso de mecanismos de otimização visa identificar soluções para a derivação de um processo específico de projeto (MAGDALENO, 2010). O trabalho apresenta a representação de LPrS através da notação *OdysseyProcess-FEX* (TEIXEIRA, 2011) e componentes de processos de software. No entanto, como o foco do trabalho está na fase de derivação e desenvolvimento de mecanismos de otimização para seleção de variantes o trabalho não discute procedimentos de como organizar os elementos de processo em componentes e não fornece uma infraestrutura de reutilização para apoio a todas as etapas de desenvolvimento e uso de uma LPrS.

MARTÍNEZ-RUIZ *et al*. (2013) aplicam o paradigma de processos rico em variantes no cenário de desenvolvimento de software global. Um ciclo de quatro etapas (adaptação, execução, análise e padronização) para transformação e inclusão de processos como novos e efetivos ativos é proposto. O ciclo visa inserir variantes no processo base já definido, coletar dados de desvios durante a execução para análise de melhorias que serão incorporadas no processo padrão e disponibilizadas para uso. O paradigma oferece suporte à variabilidade nas duas primeiras etapas do ciclo. A modelagem de variabilidade no caso de estudo apresentado que utiliza o paradigma proposto inclui um procedimento de quatro etapas: (1) Análise do Contexto de Variabilidades (Escopo do Contexto; Análise Sintática e Semântica); (2) Projeto das Variações; (3) Implementação de Variantes seguindo a abordagem de aspectos apresentada em MARTÍNEZ-RUIZ *et al.* (2011) e (4) Apresentação de resultados. Uma ferramenta denominada vEPF fornece suporte às atividades de definição e adaptação de processos ricos em variantes.

A Tabela 3 apresenta um resumo da análise das abordagens de LPrS realizada segundo os critérios definidos na Seção 3.2.

| Abordagem | Descrição do Processo de desenvolvimento *para* reutilização | Descrição do Processo de desenvolvimento *com* reutilização | Definição do conceito de Arquitetura da LPrS | Definição do conceito de Componente de Processo | Ferramental de Apoio |
|---|---|---|---|---|---|
| **DURÁN *et al.* (2004)** | Parcialmente | Não | Não | Não | Não |
| **WASHIZAKI (2006)** | Sim | Parcialmente | Sim | Não | Não |
| **SIMIDCHIEVA *et al*. (2007)** | Não | Não | Não | Não | Não |
| **HALLERBACH *et al.* (2008)** | Parcialmente | Parcialmente | Não | Não | Não |
| **ARMBRUST *et al.* (2008) (2009)** | Parcialmente | Parcialmente | Sim | Não | Não |
| **TERNITÉ (2009)** | Não | Não | Sim | Não | Não |
| **SIMMONDS *et al.* (2001)<br>SIMMONDS e BASTARRICA (2011)<br>SIMMONDS *et al.* (2012)** | Sim | Não | Não | Não | Parcialmente |
| **BARRETO *et al*. (2007) (2009) (2010) (2011)** | Sim | Sim | Sim | Sim | Parcialmente |
| **ALEIXO *et al*. (2010) (2012)** | Parcialmente | Parcialmente | Parcialmente | Não | Parcialmente |
| **ALEGRÍA *et al*. (2011)<br>ALEGRÍA e BASTARRICA (2012)<br>ALEGRÍA *et al*. (2013)** | Sim | Sim | Parcialmente | Não | Parcialmente |
| **MAGDALENO *et al*. (2012)** | Sim | Sim | Não | Parcialmente | Parcialmente |
| **MARTÍNEZ-RUIZ *et al*. (2013)** | Parcialmente | Parcialmente | Não | Não | Parcialmente |

**Tabela 3 - Resumo da Análise das Abordagens de LPrS**

A análise identificou a existência de alguns métodos de apoio ao desenvolvimento e uso de LPrS, através da descrição de atividades. Pode-se observar que as abordagens não apresentam a definição de técnicas e procedimentos de apoio à execução de todas as atividades especificadas. A organização dos diferentes mecanismos encontrados nas diferentes abordagens pode auxiliar na formação do corpo de conhecimento necessário para guiar processos de desenvolvimento *para* e *com* reutilização de processos de software. O uso de conceitos de DPBC foi analisado através da identificação da presença do conceito de Arquitetura de LPrS ou do conceito de componente de processo. Esses conceitos não são amplamente utilizados pelas abordagens. Na maioria dos casos, apenas uma definição é provida

sem especificar procedimentos para organização de elementos de processos em componentes ou sem definição de como estabelecer os relacionamentos existentes em uma estrutura arquitetural. Por último, a análise do ferramental de apoio disponibilizado visou identificar quais etapas da abordagem possuem apoio automatizado. Poucas abordagens possuem apoio ferramental a todas as etapas. Nenhuma infraestrutura de reutilização de processos de software, como um mecanismo integrado das etapas, foi encontrada disponível na literatura. A análise da representação de variabilidade é discutida na Seção 3.3. 1.

### 3.3.1. Variabilidade de Processos de Software nas Abordagens de LPrS

Assim como em Linha de Produtos de Software (LPS), o conceito de variabilidade está fortemente presente na técnica de Linha de Processos de Software (LPrS). A variabilidade em uma família de processos visa explicitar as similaridades presentes e os pontos em que tais processos se diferenciam entre si. O conceito de opcionalidade, que indica a necessidade ou não da presença de um elemento em processo específico, também é de extrema relevância na identificação e representação das variabilidades de uma família de processos.

As abordagens de LPrS identificadas foram analisadas segundo cinco critérios focados no apoio à representação de variabilidades em famílias de processos: (1) identificação dos elementos de processos que foram considerados pela abordagem como passíveis de sofrer variação; (2) identificação do tipo de variação utilizado: opcionalidade; variabilidade; ações de adaptação; evolução das variações, etc.; (3) identificação do tipo de restrições e regras de consistência foi definido; (4) identificação do uso de alguma notação existente ou de alguma proposta de notação; e (5) identificação de algum ferramental de apoio à modelagem.

A Tabela 4 busca enumerar quais elementos de processos foram considerados como passíveis de sofrerem alterações nas abordagens analisadas. Em sua maioria, os elementos unidades de trabalho, papéis e produtos de trabalho foram os mais abordados. A forma como a variabilidade é especificada é descrita e possui diversas formas de realização, usando mecanismos de parametrização; uso do relacionamento de herança; uso da representação de alternativas via ponto de variação e variantes; uso do conceito de opcionalidade; uso do conceito de blocos de construção (componentes de processos); uso de operações de adaptação (inserção, exclusão, modificação, etc.). Os aspectos que podem provocar variações são descritos na tabela junto a possibilidade de descrição de restrições e regras de consistência entre elementos ou modelos, no caso de mais de um artefato ser utilizado para representar a LPrS. Em geral, as abordagens não especificam mais de um nível de abstração para representar a LPrS, e em casos de uso de mais de um modelo, os procedimentos de mapeamentos entre os artefatos muitas vezes não são definidos.

| Abordagem | Elementos de Processos que podem sofrer variações | Formas de realização da variaçãopara reutilização | Tipos de Restrições e Regras de Consistência |
|---|---|---|---|
| **SUTTON e OSTERWEIL (1996)** | Elementos citados: Passos; componentes de produtos produzidos; sequenciamento ou concorrência de execuções dos passos. | Formas de Realização: Parametrização; Controle em tempo de execução; Componentização. | As variações podem ser provocadas por considerações relacionadas ao produto ou a outros critérios, como tamanho e complexidade do produto, expectativas com manutenção, evolução e reutilização, e a natureza do projeto ou da organização, restrições de consistência impostas, tamanho e estrutura da equipe, natureza dos recursos usados, entre outros fatores. |
| **DURÁN *et al.* (2004)** | Atividades; papéis e *templates* de documentação. | Descrição de alternativas possíveis para a implantação do processo e subpassos em blocos condicionais e o uso de herança para representação de variantes. | Critérios de seleção das características apropriadas foram definidos através de heurísticas. |
| **ROMBACH (2006)** | Não especificado. | Componentes de Processos, artefatos de processos reutilizáveis em todos os níveis de abstração. | Os discriminadores que impactam em variações são objetivos de produtos e processos, assim como características de projeto (contexto). |
| **WASHIZAKI (2006a)(2006b)** | Atividades, artefatos de entrada e saída, e papéis. | Definição de similaridades (processo núcleo) e variabilidades (variações): pontos de variação e variantes. Elementos também podem ser classificados como opcionais: Modelo de Arquitetura de Linha de Processo. | Cita relações de dependência e de seleção exclusiva para consistência dos processos. As variações são ocasionadas por características específicas de projetos (requisitos pré-estabelecidos). Relações devem ser estabelecidas entre os modelos da Arquitetura de Linha de Processo e o Modelo de Características. |
| **SIMIDCHIEVA *et al*. (2007)** | Agentes; tarefas/passos; artefatos. | Multiplicidade (zero/uma ou mais instâncias de tarefas/passos); descrição de alterações do comportamento do agente (variações na forma de execução de passos pelo papel responsável pela execução); variações nos passos (refinamento da descrição de passos que se encontram no nível mais baixo da hierarquia que representa o processo); variações no conteúdo dos artefatos. | Não especificado. |
| **HALLERBACH *et al*. (2008)** | O trabalho trata como elementos de processo, mas demonstra o uso com atividades. | As variações são trabalhadas através da definição de opções para um processo básico com o uso de ações: inserção; exclusão; alterações na ordem de execução e modificação de atributos. | Restrições entre opções de variantes de processos definidas através de relações de dependência (AND); exclusão mútua (XOR); restrições de ordem de execução (sequencia) e definição de hierarquia. Definição de situações de contexto para auxiliar as configurações de processos. Os valores das variáveis de contexto podem ser fixos ou dinâmicos (variando durante o tempo de execução). Situações de contexto mais complexas são descritas usando regras IF THEN ELSE. |
| **ARMBRUST *et al.* (2008) (2009)** | A modelagem de variabilidades não é o foco do trabalho. | Tratamento de similaridades e variabilidades representadas por blocos de construção. As partes de processos opcionais são marcadas como uma descrição detalhada de quando devem ser consideradas através de regras. No entanto, o trabalho aborda essa modelagem superficialmente, não explora a técnica, nem descreve quais são os elementos envolvidos nessa modelagem. No exemplo, as variabilidades são representadas como opcionalidades e trabalhadas de acordo com os artefatos usados e gerados na execução das atividades | Modelo de decisão, incluindo todos os pontos de variação, é especificado, o que permite a derivação de um modelo de processo específico a partir dos artefatos construídos. |
| **TERNITÉ (2009)** | Atividades, artefatos e papéis. Relações entre elementos de processos. | Tipos de características*: mandatória (característica deve sempre ser incluída); opcional (características adicionais – um complemento independente que pode ou não ser incluído) e alternativa (características para substituição de outras). Tipos de variabilidades: positiva (adição de elemento de processo ou | As características podem ser: (1) independentes: a seleção de uma característica não afeta a seleção de outra; (2) inclusivas: a seleção de uma característica implica na seleção de outra e (3) mutuamente exclusivas: características não podem ser selecionadas em conjunto. |

| Abordagem | Elementos de Processos que podem sofrer variações | Formas de realização da variaçãopara reutilização | Tipos de Restrições e Regras de Consistência |
|---|---|---|---|
| | | relações), negativa (elementos ou relações são removidos), extensão (elementos ou relações são estendidos) e substituição (elementos ou relações são substituídos).<br>* Característica = elemento de processo. | |
| **SIMMONDS *et al.* (2011)<br>SIMMONDS e BASTARRICA (2011)<br>SIMMONDS *et al.* (2012)** | Tarefas; produtos de trabalho e papéis (produtos e papéis variam com relação a tarefas). | A abordagem utiliza dois modelos para tratar a variabilidade. Um modelo de processo geral e a representação da variabilidade através de um modelo de características.<br>A variação possível identificada pela abordagem trata de elementos classificados como opcionais ou alternativos. | Os tipos das variantes associadas a um ponto de variação devem ser do mesmo tipo do ponto de variação.<br>Para lidar com questões de inconsistências entre os dois modelos previamente especificados, um terceiro modelo estabelece restrições de consistência. |
| **BARRETO *et al*. (2010)<br>BARRETO *et al.* (2011)** | Componente de processo (subprocesso e atividade) e tarefa | Componentes de Processos mandatórios e opcionais, além de representar pontos de variação. A variabilidade é trabalhada com a definição de componentes concretos, não configuráveis, e componentes abstratos, que permitem configurações através de implementação por componentes concretos. Uma *conexão de elementos de processo* também pode ser opcional, indicando que, no momento da definição de um processo baseado na *arquitetura de processos*, deverá ser tomada uma decisão sobre incluir ou não a *conexão* no processo resultante. | Uma *característica* pode ser vista como um aspecto, qualidade ou caracterização com a qual o *processo* precisa ser compatível. As *características* podem ser consideradas um conjunto de regras aplicadas a *componentes de processo* que guiam a definição de *processos* com base nas necessidades do processo. Modelo proposto prevê dois tipos de relações: *relação* de *dependência* e de *conflito*. Essas *relações* são um mecanismo auxiliar adicional para permitir a modelagem de restrições sobre a seleção de *componentes*. |
| **ALEGRÍA *et al*. (2011)<br>ALEGRÍA E BASTARRICA (2012)<br>ALEGRÍA *et al*. (2013)** | Atividade, tarefa, papel e produto de trabalho. | Tratamento de opcionalidade e variabilidade. | O processo organizacional deve se adaptar as características de um projeto em particular. Assim, as características da organização ou de projeto devem ser capturadas através de um modelo de contexto, que representa as informações do ambiente onde o processo será aplicado. |
| **MARTÍNEZ-RUÍZ *et al*. (2008)<br>(2011)** | Atividade, tarefa, produto de trabalho e papel. | Tratamento de variabilidade: classificação de elementos de processos como pontos de variação (pontos no modelo de processo em que as variações ocorrem) e variantes. Relações de ocupação de um ponto de variação por suas variantes são representadas por relações de generalização indicando classificação de opcional, mandatório ou alternativo. | Relações de dependência de inclusão ou exclusão podem ser especificadas entre pontos de variações, entre variantes, e entre pontos de variações e variações.<br>Definição de relações para manutenção das relações de dependência entre elementos e descrição de restrições para manter consistência. |

Tabela 4 - Análise do apoio à representação de variabilidades nas abordagens de LPrS identificadas

A Tabela 5 apresenta a lista de notações para modelagem de variabilidades usadas pelas abordagens e descreve se algum apoio ferramental é provido para suporte a esta atividade. Pode-se notar que apenas algumas abordagens tratam da definição de notações específicas para o contexto de reutilização de processos de software. A maioria aplica notações já existentes para modelagem de processos ou para modelagem de variabilidade em produtos de software. A representação gráfica muitas vezes não é especificada. O ferramental de apoio utiliza, de modo geral, ferramentas já existentes da área de produtos de software ou modelagem de processos. Poucos casos se preocupam em unificar em um mesmo ambiente o desenho de diferentes modelos usados na abordagem e não há especificação de uma verificação sintática ou semântica dos modelos construídos.

| Abordagem | Notação | Ferramental de Apoio |
|---|---|---|
| **SUTTON e OSTERWEIL (1996)** | Não especificado. | Não especificado. |
| **DURÁN *et al.* (2004)** | Através de descrições adicionais em formulários de cenários (Seção variações). Uso de diagramas de classes UML para expressar as variantes possíveis (Uso de herança). | Não especificado. |
| **ROMBACH (2006)** | Indica a necessidade de construir uma linguagem de modelagem de características para especificação de variabilidade como uma tarefa importante. | Não especificado. |
| **WASHIZAKI (2006a)(2006b)** | Extensões do SPEM 1.1.<br><br>Modelagem de características baseada no FODA.<br>No entanto, não descreve as adaptações das notações adotadas. | Não especificado. |
| **SIMIDCHIEVA *et al.* (2007)** | Little-JIL | Não especificado. |
| **HALLERBACH *et al.* (2008)** | Notação própria da abordagem PROVOP. | Não especificado. |
| **ARMBRUST *et al.* (2008) (2009)** | Ferramenta gráfica de modelagem de processos de software SPEARMINT™. | Ferramenta gráfica de modelagem de processos de software SPEARMINT™. |
| **TERNITÉ (2009)** | Metamodelo V-Modell XT 1.3 | Não especificado. |
| **SIMMONDS *et al.* (2011)<br>SIMMONDS e BASTARRICA (2011)<br>SIMMONDS *et al.* (2012)** | Modelagem de variabilidades usando modelo de características. | O modelo de processo geral utiliza a notação SPEM 2.0 através da ferramenta EPFC (*Eclipse Process Framework Composer*). A variabilidade é representada através de um modelo de características através da ferramenta SPLOT. Como forma ilustrativa da abordagem, a variabilidade do processo geral foi demarcada manualmente.<br>O terceiro modelo que trata da consistência entre os modelos anteriores é tratado através da ferramenta Modisco/AMW. |
| **BARRETO *et al.* (2010)<br>BARRETO *et al.* (2011)** | Notação própria para modelagem de LPrS através de modelo de componente. | A2M – Ambiente de Alta Maturidade.<br>Cadastro de Componentes e Características e Tela de edição do modelo de componentes de processos que representa a LPrS. |
| **ALEGRÍA *et al.* (2011)<br>ALEGRÍA e BASTARRICA (2012)<br>ALEGRÍA *et al.* (2013)** | SPEM 2.0 para representar o processo organizacional (eSPEM) e modelagem de características sem notação específica. | Eclipse Modeling Framework EMF 3.4<br>TCS e ATL rules.<br>ATL plug-in 2.0<br>Exeed (Extended EMF Editor). |
| **MARTÍNEZ-RUÍZ *et al.* (2008) (2011)** | vSPEM<br>Extensão do SPEM com elementos para tratamento de variabilidade. | Não especificado. |

Tabela 5 - Análise do uso de notação específica para modelagem de variabilidade e do apoio ferramental fornecido nas abordagens de LPrS identificadas

Algumas notações foram propostas especificamente para tratar da representação de variabilidades em processos de software. O metamodelo SPEM (*Software & Systems Process Engineering Meta-Model 2.0*) (OMG, 2008) foi proposto pela OMG (*Object Management Group*) com os meios necessários para modelar, documentar, apresentar, gerenciar e instanciar métodos e processos de desenvolvimento. A especificação separa conteúdo de métodos reutilizáveis da sua aplicação em processos. O conteúdo de métodos de desenvolvimento fornece explicações passo a passo, descrevendo como objetivos de desenvolvimento específicos são alcançados independentes da colocação dessas etapas dentro de um ciclo de desenvolvimento. Já os processos usam os elementos de conteúdo de métodos e os relacionam dentro de sequências parcialmente ordenadas, customizadas para tipos específicos de projetos. A representação de variabilidade é tratada através de mecanismos de extensão com quatro tipos de relacionamentos: (1) Contribuição (*contributes*): indica que o elemento base é logicamente substituído por uma variante aumentada do elemento que adiciona valores de atributos e relacionamentos de associações; (2) Substituição (*replaces*): indica que o elemento base é substituído por uma variante do elemento com a definição de outros valores de atributos

e associações; (3) Extensão (*extends*): indica uma relação de herança, provendo a capacidade de reutilização das propriedades do elemento base. Proporciona a capacidade do elemento variante de herdar as propriedades do elemento base, mas com a possibilidade de definição do seu próprio conteúdo; e (4) Extensão-substituição (*extends-replaces*): que permite uma substituição seletiva de parte dos valores de atributos e instâncias de associações. Apenas substitui os valores que foram redefinidos, deixando da mesma maneira todos os outros valores do elemento base. Essa abordagem apresenta um conjunto de limitações apontado por (MARTÍNEZ-RUIZ *et al*., 2008) como a não possibilidade de representar os pontos comuns a todos os processos de uma linha, ou seja, as características do núcleo da linha não podem ser especificadas. Desta forma, não é possível garantir a manutenção dos objetivos dos processos.

PILLAT *et al*. (2012) apresentam uma proposta de representar mecanismos de adaptação de processos de software no meta-modelo BPMN 2.0. Os mecanismos propostos utilizaram os conceitos de adaptação de processos apresentados pelo SPEM 2.0. O objetivo é acrescentar a área de processos de software uma notação, que apesar não focada na modelagem desse tipo de processo, com a vantagem de possuir ferramentas e técnicas para otimização, execução e simulação de processos. No entanto, a proposta está em um estágio inicial e ainda apresenta alguns desafios de adaptação do BPMN para o contexto de processos de software, como inserir as modificações propostas para o meta-modelo em ferramentas de modelagem.

Outra modelagem proposta na abordagem de BARRETO (2011) representa uma LPrS através de uma arquitetura de processos, um "esqueleto" de processo com seus principais elementos e suas relações. A modelagem é realizada através de um modelo de componentes que descreve componentes de processo mandatórios e opcionais, além de representar pontos de variação. A variabilidade é trabalhada com a definição de componentes concretos, não configuráveis, e componentes abstratos, que permitem configurações através da implementação por componentes concretos. O conjunto de elementos usados na modelagem e a notação são mostrados na Figura 6 (CARDOSO, 2012).

Este trabalho usa o conceito de características de processos como um aspecto, qualidade ou característica que um processo precisa ser compatível (BARRETO, 2011). Características de processo são usadas como um mecanismo para seleção de componentes, podendo ser entendidas como um conjunto de regras (necessidades de processos) que guiam a definição de processos.

Essa abordagem apresenta uma representação explícita dos conceitos de variabilidades e opcionalidades, mas apenas para um grupo de elementos de processos (alto nível de granularidade – componentes e tarefas). A análise de opcionalidades e variabilidades em

elementos como papéis, artefatos e ferramentas não é tratada nem expressa graficamente. Relações de dependência e mútua exclusividade não são amplamente exploradas.

| Notação Utilizada | Nome do Elemento | Descrição |
|---|---|---|
|  | Componente de processo concreto obrigatório | Representação gráfica de elementos não configuráveis (variante). |
|  | Componente de processo concreto opcional | Representação gráfica da opcionalidade de elementos não configuráveis (variante). |
|  | Componente de processo abstrato obrigatório | Representação gráfica de pontos de variação. |
|  | Componente de processo abstrato opcional | Representação gráfica da opcionalidade de pontos de variação. |
|  | Atividade obrigatória | Representação gráfica de uma atividade obrigatória em uma arquitetura de processos. |
|  | Atividade opcional | Representação gráfica de uma atividade opcional em uma arquitetura de processos. |
|  | Tipo de conexão opcional entre elementos de processo | Representação gráfica do relacionamento opcional entre elementos de processos. Nesse caso, uma relação do tipo "Fim-Início". |
|  | Tipo de conexão obrigatória entre elementos de processo | Representação gráfica do relacionamento obrigatório entre elementos de processos. Nesse caso, uma relação do tipo "Início- Início". |
|  | Item início da arquitetura | Representação gráfica do ponto de início dentro de uma arquitetura. |
|  | Item fim da arquitetura | Representação gráfica do ponto de encerramento dentro de uma arquitetura. |

**Figura 6 - Descrição dos elementos gráficos notação BARRETO (2011) - Fonte (CARDOSO, 2012)**

A abordagem CASPER (ALEGRÍA *et al.*, 2013) propõe a representação das informações da LPrS em três modelos: modelo contextual, modelo de características e modelo de processos com variabilidades.

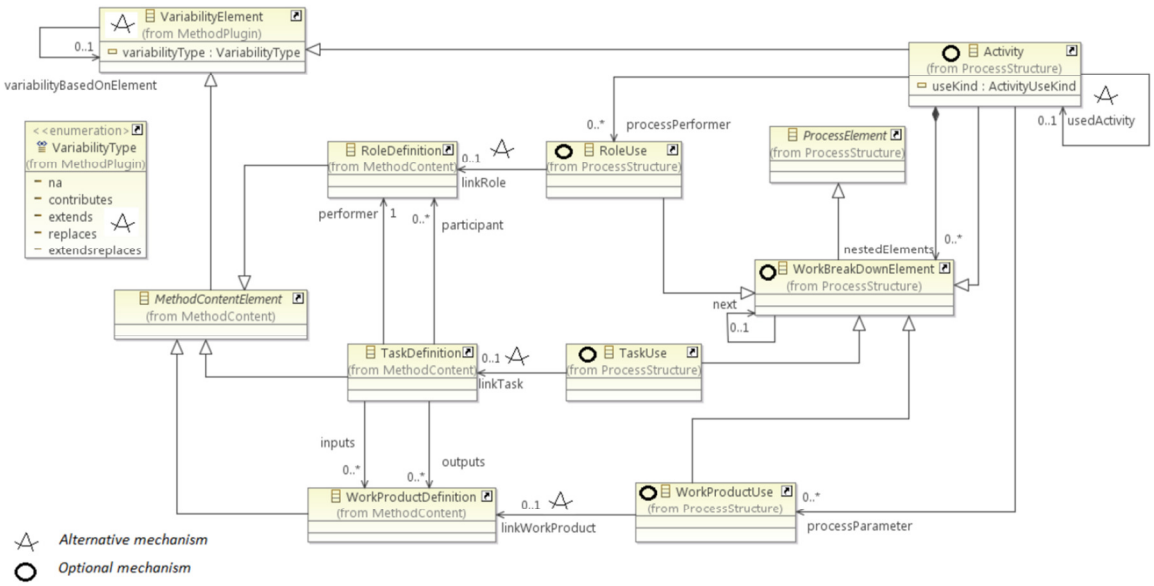



**Figura 7 - Metamodelo eSPEM (ALEGRÍA e BASTARRICA, 2012)**

O modelo contextual representa informações do contexto de diferentes projetos relacionados à linha. O modelo define atributos de elemento de contexto com valores associados. O modelo de característica é definido através de um metamodelo que contempla os conceitos da notação de CZARNECKI e ANTKIEWICZ (2005) com os estereótipos do SPEM 2.0. O modelo de variabilidade é uma extensão do SPEM 2.0, denominada eSPEM, com mecanismo

alternativo e de opcionalidade introduzido aos elementos de processo atividade, tarefa, papel e produto de trabalho (Figura 7). Nenhuma representação gráfica é descrita, nem tratamento de variações nos comportamentos dos processos em termos de fluxos de controle e de dados.

A abordagem vSPEM (MARTÍNEZ-RUIZ *et al.*, 2008; MARTÍNEZ-RUIZ *et al.*, 2011) apresenta uma extensão do SPEM v2.0 para permitir a especificação de variabilidades em modelos de processos de software. Para isso, a abordagem acrescenta ao metamodelo SPEM um pacote denominado *ProcessLineComponents,* que trata da criação de elementos para trabalhar variabilidade nos modelos. As relações entre elementos classificados como pontos de variação e variantes são realizadas através de uma relação denominada Ocupação. Essa relação pode ser opcional, mandatória ou alternativa e possui um conjunto de restrições associadas para determinar o tipo de elemento variante que pode vir a ocupar um ponto de variação. Dependências podem ser expressas através de relações entre os elementos participantes de uma LPrS (pontos de variação e variantes). Essa relação pode ser inclusiva ou exclusiva. Um conjunto de novos ícones é proposto à notação SPEM para viabilizar a representação explícita de tais conceitos (Figura 8).

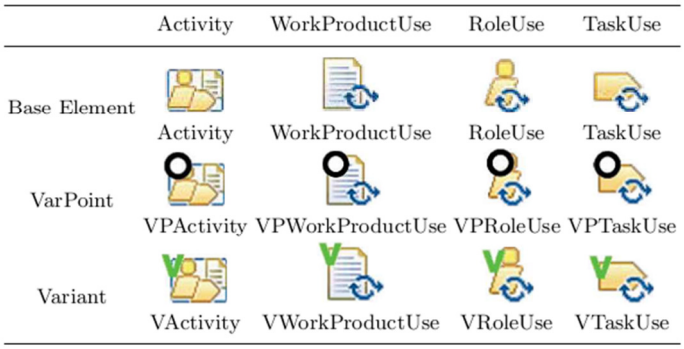



**Figura 8 - Ícones gráficos vSPEM (MARTÍNEZ-RUIZ *et al.*, 2008)**

Desta forma, as variações entre os diferentes processos que compõem a linha são gerenciadas segundo a criação de pontos de variação e tratamento da ocupação por variantes. As classificações de opcionalidades apesar de atribuídas a diversos elementos não possui representação gráfica explícita. Definições de relações de dependência e mútua exclusividade ficam restritas a elementos pertencentes a relacionamentos de variabilidade. O tratamento do impacto de variações no comportamento do processo através dos fluxos de controle e de dados não é mencionado na proposta.

Outras limitações a essa abordagem foram apontadas em PAZIN (2012), que propõe a abordagem SMartySPEM para dar suporte ao gerenciamento de variabilidades em LPrS. Baseada nas abordagens SMarty (OLIVEIRA JUNIOR *et al.*, 2010; FIORI *et al.*, 2012) e SPEM 2.0, propõe extensões que permitam representar variabilidades em processos de software. Tal abordagem é composta por um perfil UML, denominado *SMartySPEMProfile,* e um conjunto de

diretrizes que guiam o usuário para identificar e representar variabilidade em elementos de processos de software (atividade, tarefa, passo, papel e artefato). O SMartySPEMProfile define um conjunto de estereótipos e meta-atributos para representar variabilidades em modelos de LPrS (Figura 9).

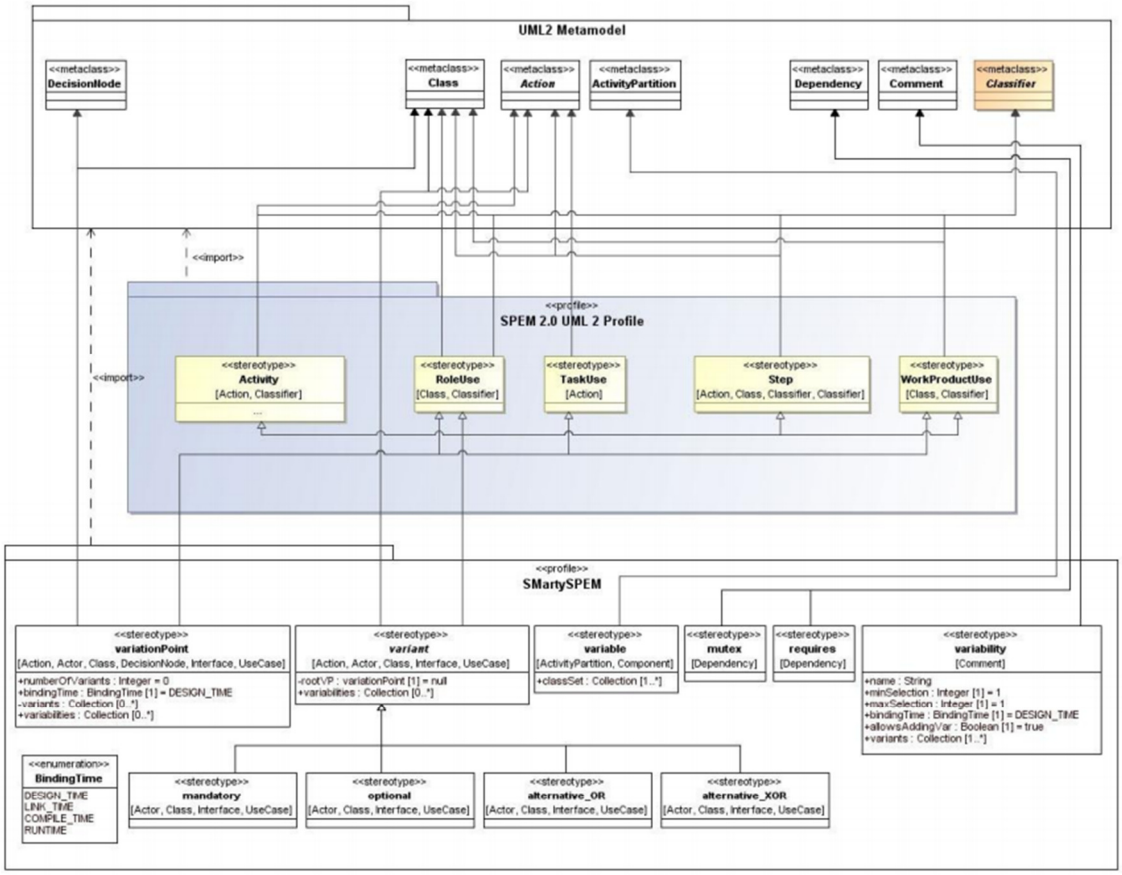



Figura 9 - Estereótipos e meta-atributos do SMartySPEMProfile (PAZIN, 2012)

O conceito de variabilidade é tratado como um estereótipo (<<*variability*>>) através de uma extensão da metaclasse *Comment*. Possui um conjunto de atributos que permite relacionar o número mínimo e máximo de variantes que podem solucionar um ponto de variação; o momento em que a variabilidade deve ser resolvida; e o conjunto de variantes associadas. O conceito de ponto de variação, que estabelece locais específicos de elementos de processos onde ocorre variação, pode ser atribuído a nós de decisão; papéis; tarefas e artefatos. O conceito de variante pode ser atribuído a atividades; papéis; tarefas; passos e artefatos. Um pouco diferente do conceito tradicional em que variante é designada apenas como alternativa para ponto de variação, a abordagem classifica variante como elementos de processos que promovam algum tipo de variação na derivação de processos específicos a partir de uma LPrS. Variantes podem ter como classificação adicional o tratamento de opcionalidade através dos estereótipos <<*mandatory*>>: para designar elementos de processos obrigatórios que devem ser parte de todos os processos específicos derivados de uma LPrS; e <<*optional*>>: para designar elementos de processos opcionais que podem ou não ser parte de processos

específicos derivados de uma LPrS. Para variantes associadas à resolução de pontos de variação, os esterótipos <<*alternative_OR*>> e <<*alternative_XOR*>> podem ser utilizados para especificar que diferentes combinações de elementos de processos podem ser aplicadas ou que apenas um único elemento de um conjunto associado pode ser aplicado, respectivamente. O relacionamento de dependência entre dois elementos de processos é representado pelo estereótipo <<*requires*>> e o relacionamento de mútua exclusividade entre dois elementos de processos é representado pelo estereótipo <<*mutex*>>.

Um conjunto de diretrizes é definido com o objetivo de auxiliar na identificação e representação de variabilidades em LPrS. Um primeiro conjunto de diretrizes foca no tratamento de variabilidades através de diagramas de atividades UML. Trabalha a variação em termos de atividades que podem ser consideradas variantes dentro do domínio de aplicação de um processo de software. O segundo conjunto trata das variabilidades dentro da modelagem com SPEM 2.0. Um exemplo do uso da abordagem na representação de uma LPrS hipotética pode ser encontrada na Figura 10.

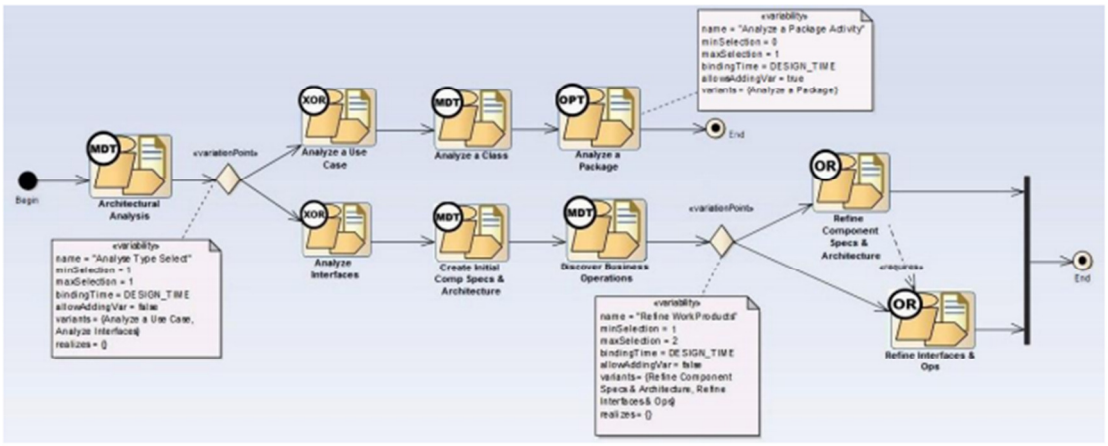



**Figura 10 - LPrS hipotética extraída de PAZIN (2012)**

Esta notação não trabalha os conceitos de opcionalidade e variabilidade de forma ortogonal. A classificação de opcionalidade não é atribuída a elementos classificados como alternativas a pontos de variação na LPrS. Algumas representações gráficas tiveram seu conceito inicial modificado, como é o caso do ponto de decisão representado um elemento de processo como uma atividade, o que pode causar alguma dificuldade no entendimento e aplicabilidade. O elemento passo, que indica uma ação, também pode ser classificado como ponto de variação, atribuindo variabilidade a um baixo nível de granularidade. No entanto, não fica claro como os múltiplos comportamentos do processo serão afetados por tantos níveis de variação. O elemento ferramenta não foi contemplado na abordagem.

## 3.4. Componentes de Processos de Software

Segundo HUMPHREY (1989), uma organização geralmente necessita de diferentes tipos de processos. Essa família de processos deve incluir uma arquitetura de processos, uma biblioteca de definições de processos reutilizáveis e vários *templates*, padrões e formas. A arquitetura de processo deve relacionar todos os membros dessa família de processos e deve conter as seguintes seções: definições de elementos de processos maiores e suas funções; formatos de padrões de processos; especificações de interface e regras de composição e adaptação. A biblioteca de elementos de processos reutilizáveis deve ser usada como fonte de blocos de construção comuns para construir múltiplas definições de processos. Esses processos devem ser construídos de forma consistente com a arquitetura.

A construção de modelos de processos como conjuntos de componentes que interagem permite interoperabilidade através do encapsulamento da semântica de representações existentes. Uma abordagem baseada em componentes evita uma profunda integração de modelos semânticos; controla a complexidade natural de processos de software; responde a dinamicidade de processos de software e facilita a reutilização (GARY *et al.,* 1998). Assim como em reutilização de produto, não é possível reutilizar planos de projetos inteiros através de múltiplos projetos, dado um conjunto de parâmetros de projetos. No entanto, se elementos discretos, como componentes de software, são construídos com estratégias de reutilização, passa a ser possível integrá-los para criar planos de desenvolvimento de software para múltiplos projetos (BHUTA *et al*., 2005).

Componentes de processo devem ser compreensíveis, ou seja, devem possuir um conjunto de informações que descrevam suas funcionalidades, o valor ou resultados providos, sua forma de execução e os recursos necessários para sua execução. Componentes também devem ser independentes, podendo ser executados sem outros elementos. Além disso, devem ser robustos, sendo tolerantes a falhas (BHUTA *et al*., 2005). Uma análise da definição do conceito de componente de processo encontrada em uma revisão inicial da literatura de abordagens de definição de processos de software baseada em componentes é apresentada na Tabela 6.

A análise dos diferentes trabalhos permitiu observar que ainda não há um consenso da definição do termo componente de processo. Por muitas vezes, componente de processo é citado como um fragmento de processo e visto como o encapsulamento de um conjunto de informações e comportamentos, mas sem especificar a granularidade e como os diversos elementos de processo essenciais são trabalhados. De forma geral, o conteúdo de um componente de processo é formado por uma unidade de trabalho, variando de tarefas a processos.

| Abordagem | Definição de Componente de Processo |
|---|---|
| BARRETO *et al.* (2010) (2011) | Unidade básica de composição de processos. É considerado algo relevante para: ser reutilizado; ter sua estabilidade e desempenho analisado; ser versionado. |
| GARY *et al.* (1998) GARY e LINDQUIST (1999a) (1999b) | Um componente pode ser definido como uma representação de entidades de processos bem encapsuladas (atividades, papéis e agentes, e produtos).<br>Um encapsulamento de informações e comportamentos de processos em um dado nível de granularidade. |
| RU-ZHI *et al.* (2003) (2005) (2010) | Uma ou uma série de atividades encapsuladas para atingir um objetivo especificado com produtos reconhecidos. |
| BHUTA *et al.* (2005) | Um elemento de processo é um grupo de atividade de projetos, e /ou outros elementos relacionados por dependências lógicas, que quando executados provêem valor ao projeto. |
| DAI e LI (2007a) (2007b) DAI *et al.*(2008) | Um componente de processo de evolução é um fragmento de processo consistente e coeso. Esse componente é considerado uma caixa-preta sem comunicação direta com outro componente, realizada através de conectores. Encontra-se estruturado através de atividades, condições e relações de fluxos. |
| SEGRINI (2009) | Um componente pode ser entendido como uma macroatividade ou processos simples e complexos, com interfaces definidas. |
| LANNA (2009) LANNA e PIETROBON (2010) | Um componente de processo foi definido como sendo uma unidade de composição com interfaces e dependências de contexto bem definidas; capaz de representar um processo de software em seus três aspectos seja qual for seu nível de detalhe (processo, atividade ou tarefa); e capaz de armazenar os itens de conhecimento relacionado ao próprio componente. |
| SPEM (OMG, 2008) | O meta-modelo SPEM define um componente de processo como um pacote de processo especial que aplica os princípios de encapsulamento. Um componente de processo contém exatamente um processo representado por uma atividade e define um conjunto de produtos de trabalho como entradas e saídas do componente. |
| AVRILIONIS *et al.* (1996) | Cada ***componente*** é visto como uma descrição com o objetivo de criação de um (ou uma parte de) artefato de software. Nesta abordagem, um componente é visto como uma caixa-preta. O componente é um (ou uma parte de) modelo de processo elementar correspondente a um módulo. São elementos estáveis e partes mais ou menos genéricas de modelos de processos. É constituído de uma interface e um fragmento de modelos de processo, que descreve o comportamento do componente usando uma linguagem de modelagem. Esse fragmento implementa funcionalidades, propriedades especificadas no nível de interface. |
| COULETTE *et al.* (2000) (2001) | Componentes elementares correspondem constituintes atômicos de processo como atividades, produtos, papéis e estratégias. |
| IIDA e TANAKA (2002) | *Componente* de processo auto-configurável é uma técnica chave dentro de um framework que deve prover um modelo de processo que suporte modularidade e adaptabilidade. Cada *componente* de processo encapsula uma série de atividades de processos autônomas e possui as seguintes características: (a) possui interfaces definidas explicitamente; (b) utiliza objetos (artefatos/produtos) como entrada e saída; (c) possui objetivos e responsabilidades especificados. |
| TORTORELLA e VISAGGIO (1997) FUSARO *et al.* (1998) | Um *componente* de processo pode indicar uma técnica (algoritmo ou série de passos que requer conhecimento e habilidades para produzir certo resultado), um método (procedimento gerencial para aplicação de uma técnica, organizado como um conjunto e regras que avalia, seleciona e estabelece como e quando usar uma técnica) ou um processo (conjunto de métodos e os relacionamentos necessários para alcançar um objetivo). |

Tabela 6 – Definições de Componentes de Processos

A comunicação por meio de interfaces é precariamente especificada (Tabela 7), sem definir exatamente o que são os pontos de entrada e saída do componente e como são estabelecidas as relações de fluxos de trabalho na definição de componentes mais complexos formados pela composição e adaptação de estruturas menores. A representação dos componentes, tanto gráfica como descritiva, é pouco discutida e só é efetivamente abordada em alguns trabalhos que apresentam alguma proposta de representação e classificação de componentes.

| Abordagem | Interfaces e conectores |
|---|---|
| BARRETO *et al.* (2010) (2011) | Conector: compreende um elemento fonte, um elemento destino e regras de associação (início-fim, fim-fim, etc.) |
| GARY *et al.* (1998) GARY e LINDQUIST (1999a) (1999b) | Interações entre componentes realizadas através de métodos (definem comportamentos de modificação do esquema do processo e o grafo de transição de estados), eventos (comunicação entre componentes sobre troca de estados e mudanças em suas informações) e mensagens (promover troca de estados). |
| RU-ZHI *et al.* (2003) (2005) (2010) | Não especificado. |
| BHUTA *et al.* (2005) | Interfaces de entrada (pré-condições: informações necessárias para execução como dependências e estimativas de esforço) e saída (pós-condições: resultados). |
| DAI e LI (2007a) (2007b) DAI *et al.*(2008) | Conectores: unidades de comunicação. São condições nas redes de Petri que representam os componentes. |
| SEGRINI (2009) | As interfaces de um componente devem definir as características pertinentes ao componente, tais como domínio de aplicação e paradigma de desenvolvimento, que especifiquem as condições de aplicação do componente. Interfaces definem a estrutura do componente e seu contexto de aplicação. |
| LANNA (2009) LANNA e PIETROBON (2010) | Interfaces: define os serviços daquele componente, bem como os artefatos que ele utiliza ou produz em sua execução. |
| SPEM (OMG, 2008) | A conexão entre componentes é realizada através de suas portas. No entanto, diversos problemas podem surgir: (1) o número e tipos de produtos de trabalho de entrada e saída podem não corresponder dentro do conjunto de componentes de processo; (2) a invariante requerida para entrada e saída conectadas pode ser contraditaria; (3) o nome e a constituição dos produtos de trabalho podem variar de um componente para outro; e (4) o nome e a definição de papéis podem variar de um componente para outro. |
| AVRILIONIS *et al.* (1996) | A interface é composta de duas partes: (1) interface de conexão para conectar componentes e (2) interfaces para troca de informações (entrada e saída). |
| COULETTE *et al.* (2000) (2001) | A especificação de atividades declara produtos de entrada e saída, declarações (pré e pós-condições e invariantes), e o papel dos desenvolvedores que a executam. A implementação da atividade descreve esboços de realização e guias metodológicos (tratamentos de inconsistências). |
| IIDA e TANAKA (2002) | Componentes de processo podem ser conectados com outros baseados na especificação de interfaces, que estabelecem restrições míninas. |
| TORTORELLA e VISAGGIO (1997) FUSARO *et al.* (1998) | Não especificado. |

Tabela 7 - Definição das relações entre componentes via interfaces e conectores

A maioria dos trabalhos não aborda nem trata a questão de variabilidades e opcionalidades dos diferentes elementos de processo, nem como isso se reflete no componente como um todo. Ou seja, as abordagens tratam apenas a visão do componente como unidades de trabalho, sem considerar que os outros elementos associados precisam ser identificados e analisados dentro de contextos de aplicação.

Apesar de o uso de componentes para composição de processos aparentar fornecer muitas vantagens, partir de unidades tão pequenas para compor grandes processos parece ainda ser insuficiente. Caso se analise a reutilização no contexto de produtos de software, é possível observar que uma das lições aprendidas com os esforços para se alcançar a reutilização nas últimas décadas foi a de que a reutilização *bottom-up*, ou seja, a composição de componentes arbitrários para construir sistemas, não funciona bem na prática (BARRETO, 2007). Programas de reutilização bem sucedidos devem empregar, também, uma abordagem *top-down*, ou seja, componentes são desenvolvidos de forma a se encaixarem em uma estrutura de alto nível definida por uma arquitetura de software (BOSCH, 2000). Assim, a definição de processos através de componentes pode ser potencializada se combinada com uma estrutura maior que define as diretrizes de combinação e organização das unidades fundamentais.

A Arquitetura de Processo consiste em um arcabouço conceitual para incorporar, relacionar e adaptar elementos de processos em instâncias de processos (HUMPRHEY, 1989). Essa estrutura geralmente é requerida quando há a necessidade de relacionar processos com outros já existentes ou processos futuros, como nos casos de reutilização de elementos e adaptação de processos. Segundo o CMMI-DEV (CHRISSIS *et al.*, 2006), arquitetura de processos são definidas como a ordenação, interfaces, interdependências, e outros relacionamentos entre os elementos de processo em um processo padrão ou processos externos. Podemos citar como propósitos de uma arquitetura de processo: (1) a descrição de componentes relevantes, estrutura, relacionamentos, interfaces internas e externas; e (2) guiar a seleção, composição e adaptação de componentes (DAI *et al*., 2008).

Desta forma, uma abordagem para definição de processo de software a partir de componentes deve possuir uma estrutura básica para direcionar a definição de processos específicos. A definição de uma arquitetura possui esse propósito. No entanto, uma arquitetura deve ser representada por elementos arquiteturais e os relacionamentos entre esses elementos. A definição de elementos arquiteturais deve ser suportada por sistemáticas para a organização de artefatos de processo reutilizáveis em elementos arquiteturais, os quais compõem uma arquitetura de componentes. Para isso, é esperado o uso de critérios, métricas ou outras técnicas para apoio a criação destes elementos arquiteturais. Esse suporte não foi encontrado nas diferentes abordagens estudadas (Tabela 8).

| Abordagem | Abordagem para Composição de Processos de Software |
|---|---|
| **BARRETO *et al.* (2010) (2011)** | Abordagem composta por 4 atividades: (1) Definir componentes de processos: analisar processos específicos de projeto e o processo padrão da organização e a seguir, buscar agrupamentos de atividades que estejam em um dado nível de detalhamento suficiente para serem reutilizadas, analisadas e que possuem dados de utilização associados; (2) Definir características de processo; (3) Definir uma LPr ou arquitetura; (4) Aprovar inclusão de elementos reutilizáveis na biblioteca |
| **GARY *et al.* (1998) GARY e LINDQUIST (1999a) (1999b)** | *Open Process Components (OPC) framework*<br>Divide a informação da abordagem de composição de processos em três maneiras: Esquema de processo (possíveis entidades e relações); estados de processo (transições do comportamento dinâmico do componente: executando, suspenso e abortado) e implementação de processo (representação executável de um componente). |
| **RU-ZHI *et al*. (2003) (2005) (2010)** | Definição de um arcabouço para melhoria de processo baseado em reutilização.<br>Os componentes devem ser recuperados e, posteriormente, adaptados para o contexto atual. |
| **BHUTA *et al*. (2005)** | Descreve atividades para construção e uso de elementos de processos, incluindo a definição de atributos de qualidade dos elementos e meios para recuperação dos elementos apropriados a um contexto. Cada elemento de processo construído é descrito usando o modelo 3C (Conceito, Conteúdo e Contexto) e a abordagem de Similaridade e Variabilidade com a classificação de variante e invariante das atividades. |
| **DAI e LI (2007a) (2007b) DAI *et al.*(2008)** | Uma abordagem *top-down* para composição de processos baseada em Arquitetura foi definida em cinco passos: (1) Elicitação e Análise de Requisitos de Processos; (2) Construção da Arquitetura de Processos; (3) Construção dos Componentes de Processos que compõem a Arquitetura; (4) Composição de um Processo Específico e (5) Simulação e Execução. |
| **SEGRINI (2009)** | A reutilização pode ser vertical, utilização de componentes do nível padrão e especializados na definição de processos no nível projeto. Ou ainda, a reutilização pode ocorrer em um mesmo nível de abstração, através da definição de componentes de maior nível de granularidade a partir de componentes de granularidade mais fina (reutilização horizontal). |
| **LANNA (2009) LANNA e PIETROBON (2010)** | Não especificado. |
| **SPEM (OMG, 2008)** | Não especificado. |
| **Abordagem Pynode<br><br>AVRILIONIS *et al*. (1996)** | A modularização de processos é realizada através da definição de visões, que permitem a decomposição de processos de acordo com características bem definidas (papel, atividade ou produto). Neste trabalho, foi introduzido o conceito de visão de execução (*execution view*). Uma visão de execução contém um conjunto de componentes reusáveis. A abordagem de modularização consiste em uma decomposição *top-down* de modelos de processos, combinada a uma abordagem *bottom-up*, que considera experiências anteriores, políticas pré-definidas, estratégias de gerenciamento de processo específicas, ou seja, um conjunto de informações reusáveis empacotadas em componentes. |
| **COULETTE *et al*. (2000) (2001)** | Um conjunto de fatores de qualidades de componentes foram definidos: (a) separação entre especificação e implementação; (b) auto-descrição: informações descritivas do componente; (c) autonomia: reutilização de um componente independente de outros, ou pelo menos deve descrever claramente de quais outros componentes depende; (d) evolutivo: gerenciamento de evolução estática e dinâmica; (e) consistência; (f) visões de descrição múltiplas. |
| **IIDA e TANAKA (2002)** | Este trabalho apresenta um framework para modelagem de processos de software através de componentes capazes de serem conectados com outros através de classes de conexões definidas como padrões composicionais. Um Padrão composicional é definido como um conjunto de elementos de processos compostos e fluxos de produtos entre eles (relações de conexões). Esses padrões (*templates*) são usados para busca de descrições de processos já desenvolvidos e estabelecidos como elementos de processos compostos. Cada processo composto possui um gerenciador de composição de elementos de processo baseado no padrão composicional. |
| **TORTORELLA e VISAGGIO (1997) FUSARO *et al*. (1998)** | O método é dividido em quatro passos: (1) Construção de um Modelo de Processo; (2) Validação do Modelo de Processo; (3) Modificação do Modelo de Processo; e (4) Caracterização do Componente de Processo. Para analisar e avaliar componentes é necessário formalizar o conhecimento obtido. Nesta abordagem, o formalismo de modelagem é o Modelador Conceitual de Processo (*Process Conceptual Modeler* – PCM), uma ferramenta no PROMETHEUS (*PROcess Model Evolution Through Experience Unfolded Systematically*) (Cimitile e Visaggio, 1994). |

Tabela 8 - Descrição das abordagens de DPBC

## 3.5. Conclusão

Existe uma diversidade de fatores que podem afetar o contexto no qual processos precisam ser definidos e, desta forma, um conjunto de variantes de processos podem surgir em uma organização visando atender os múltiplos cenários e métodos de desenvolvimento. Apesar da existência de diferentes abordagens visando auxiliar a definição de processos via reutilização, mais especificamente, linhas de processos, alguns pontos ainda podem ser considerados oportunidade de trabalho na área. A especificação de uma sistemática de reutilização que contemple processos de desenvolvimento *para* e *com* reutilização combinando a técnica de LPrS com conceitos de DPBC, apoio à modelagem de variabilidades da LPrS em diferentes níveis de abstração, integrados em uma infraestrutura com suporte ferramental a diferentes etapas do processo de reutilização para a definição de processos de software específicos de projetos é um diferencial ainda a ser trabalhado.

A combinação das áreas de LPrS e DPBC ainda não foi amplamente explorada nas abordagens analisadas. O conceito de componente de processo de software ainda precisa ser melhor especificado, incluindo o conjunto de informações que devem estar presentes em sua especificação e sua forma de representação descritiva e gráfica. Os relacionamentos entre componentes e seus pontos de entrada e saída devem ser definidos através de interfaces de comunicação. Técnicas, procedimentos e critérios definidos para organizar os elementos de processos reutilizáveis em componentes a serem armazenados para posterior uso e aplicação em contextos específicos precisam ser definidas. O conceito de Arquitetura de Componentes de Processos de Software precisa ser especificado, de forma a estabelecer o arcabouço estrutural de organização da LPrS através de seus componentes e suas relações de comunicação.

O uso de diferentes níveis de abstração foi utilizado em alguns trabalhos. No entanto, não fica clara a fronteira entre tais níveis. Falta um método para a definição dos rastros e ligações semânticas e sintáticas, de forma a garantir o tratamento da consistência entre os elementos e a complementariedade das informações a serem definidas, evitando redundâncias e retrabalho para o engenheiro do processo de software.

A variabilidade é mais amplamente tratada e modelada através de linguagens não elaboradas para o contexto de reutilização de processos. O foco de análise dos trabalhos encontra-se em variações de elementos estruturais de processos de software. Ainda há campo de estudo para definir os tipos de variações possíveis em comportamentos de processos designados pela variabilidade nos fluxos de execução e de dados. Assim, um tópico ainda a ser melhor explorado consiste na modelagem de variabilidade de LPrS, combinando o aspecto estrutural (elementos como unidades de trabalho, papéis e artefatos) e o aspecto comportamental (fluxos de execução e de dados) de processos de software.

O apoio ferramental é um ponto importante a ser trabalhado, pensando em uma infraestrutura de reutilização para prover o apoio necessário ao desenvolvimento e uso de uma abordagem de linha de processos de software. Tal infraestrutura deve fornecer meios capazes de guiar o engenheiro de processo e prover ferramentas para modelagem e projeto da linha com apoio através da aplicação de técnicas nas diferentes etapas envolvidas, visando minimizar o esforço envolvido e prover mecanismos de verificação da consistência dos ativos a serem reutilizados.

## 4.1. INTRODUÇÃO

A abordagem proposta ***OdysseyProcessReuse*** consiste em uma sistemática de reutilização de processos de software, combinando técnicas e conceitos de LPrS e DPBC. A abordagem envolve a definição de um método de uma Engenharia de LPrS para o desenvolvimento de núcleos de artefatos de processos de software para domínios específicos, estabelecendo uma Arquitetura de Componentes de Processos de Software como resultado final. A LPrS desenvolvida é composta por diferentes níveis de abstração complementares, assim como ocorre com LPSs, com tipos de artefatos relacionados e com representações específicas para lidar com a variabilidade do domínio de processos nos diferentes níveis. Tal abordagem é composta por cinco elementos principais (Figura 11): (1) definição do método (processo de desenvolvimento *para* reutilização de processos de software); (2) representação para modelagem de variabilidade da LPrS nos diferentes níveis de abstração; (3) estabelecimento de mecanismos de mapeamento entre os artefatos dos diferentes níveis de abstração; (4) definição de procedimentos para suportar a organização dos elementos de processos em componentes de processos da Arquitetura da LPrS; e (5) desenvolvimento de uma infraestrutura de reutilização.


**Figura 11 - Elementos Principais do desenvolvimento da *OdysseySPrLE-CBArch***

O método consiste no estabelecimento do procedimento de desenvolvimento de uma LPrS com princípios de DPBC da Engenharia de LPrS (ELPrS). A ELPrS descreve a sequência de atividades para construir uma LPrS, com o desenvolvimento dos artefatos de processos de software a serem reutilizados (processo de desenvolvimento *para* reutilização), e as atividades para a instanciação de um processo específico de projeto a partir da linha (processo de

desenvolvimento *com* reutilização). Desta forma, a Engenharia da LPrS é dividida em duas fases: Engenharia de Domínio de Processos de Software – EDPS e Engenharia de Projeto de Processos de Software – EPPS. Cada uma das etapas é dividida em três atividades principais: (1) Análise, (2) Projeto e (3) Implementação (Figura 12).

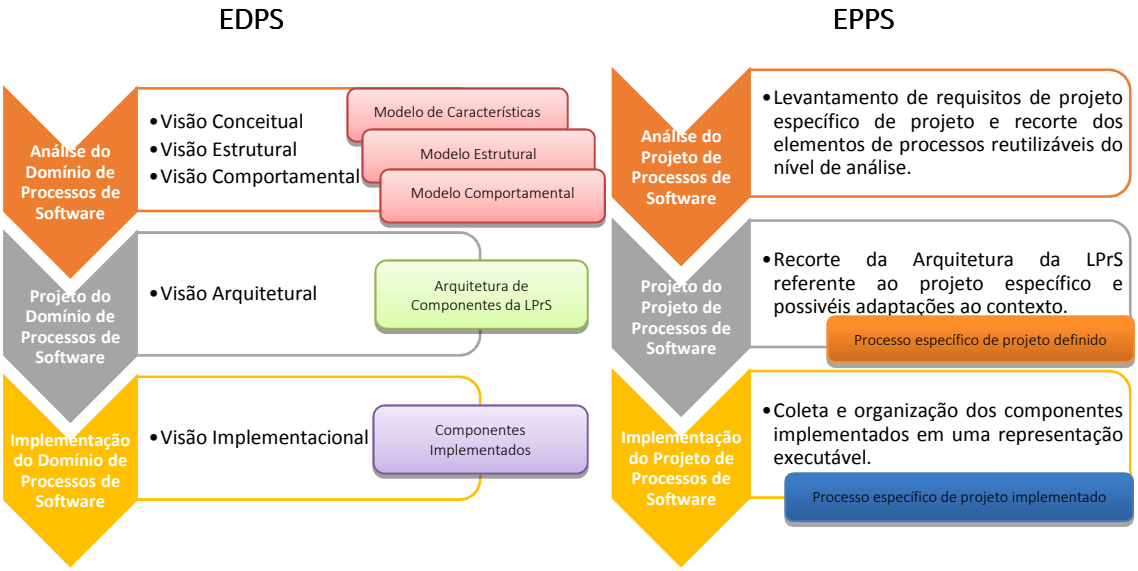



**Figura 12 - Visão Geral da Engenharia da LPrS**

Os artefatos do domínio de processos de software devem ser representados através de diferentes níveis de abstração complementares. Desta forma, cada nível permite uma visão do conjunto de elementos de processos que o compõem e como estes se relacionam. O primeiro nível de abstração representa uma visão conceitual da LPrS, um alto nível de abstração de fácil entendimento representado por um modelo de domínio que agrega os elementos básicos de processos de software reutilizáveis (unidades de trabalho, papéis e produtos de trabalho) e analisa suas propriedades de opcionalidade e variabilidade no domínio. Uma das formas de representação deste nível consiste em um modelo de características, um modelo amplamente utilizado em LPS e que visa representar o conhecimento do domínio sem a necessidade de explicitar os detalhes de mais baixo nível de abstração.

O segundo nível complementa o modelo do domínio representando a LPrS através de uma visão estrutural e uma visão comportamental. A visão estrutural especifica detalhes de como as unidades de trabalho de processos de software estão descritas em termos de passos, ações que definem como podem ser executadas mantendo e refinando as relações entre os produtos de trabalho requeridos e produzidos e os papéis envolvidos. Além disso, os possíveis comportamentos esperados para processos a serem derivados são especificados através de fluxos e sequenciamentos de execução na visão comportamental. A representação final da LPrS consiste em uma Arquitetura de Componentes da LPrS. Tal arquitetura representa um arcabouço da LPrS através do estabelecimentos das relações entre os componentes de processos de software identificados que direciona a definição de processos específicos de

projetos na EPPS, garantindo a consistência de composição dos elementos de processos reutilizáveis selecionados. A Visão de implementação funciona de forma complementar à representação dos componentes lógicos da arquitetura definida com a especificação através de uma linguagem de modelagem de processos com suporte à execução do processo derivado. Para contemplar cada nível, notações devem ser definidas para a representação de cada artefato, envolvendo o tratamento dos pontos configuráveis e opcionais do domínio, os diferentes relacionamentos entre os elementos, as regras de relações de dependência e exclusividade entre elementos e a definição das propriedades dos elementos de processos de software especificados. A Visão de Implementação encontra-se fora do escopo desta proposta.

A construção progressiva de uma LPrS através dos seus diferentes níveis de abstração necessita de um conjunto de procedimentos para realizar mapeamentos entre os artefatos, mantendo os rastros das propriedades de variabilidade, opcionalidade, relacionamentos, regras e restrições existentes de um artefato do domínio para o outro.

A organização dos elementos de processos de software especificados nos níveis anteriores em componentes de processos de software precisa ser direcionada através de procedimentos para a especificação das relações que definem o arcabouço arquitetural.

O último elemento da abordagem proposta é um ambiente de reutilização que proporciona o suporte necessário para a construção, integração e utilização dos artefatos envolvidos. Este projeto de pesquisa é baseado na adaptação do ambiente Odyssey (ODYSSEY, 2014), uma infraestrutura de reutilização de software com base em modelos de domínio desenvolvida pelo Grupo de Reutilização de Software da COPPE/UFRJ. O apoio ferramental deve suportar o desenvolvimento dos diferentes níveis de uma LPrS e gerenciar os rastros entre eles. Além disso, tal ambiente poderia prover funcionalidades adicionais que auxiliassem a equipe de Engenharia do Domínio, como apoio à verificação dos modelos que estão sendo desenvolvidos através da implantação de regras de boa formação das representações utilizadas e acompanhamento por meio de *checklists* para realização de inspeção sintática e semântica dos modelos.

Uma visão geral de uma abordagem de ELPrS pode ser vista na Figura 13. Conforme dito anteriormente, as duas grandes fases consistem na EDPS e EPPS. No entanto, o foco desta trabalho está na fase de EDPS e na construção da infraestrutura para guiar o desenvolvimento de LPrS com conceitos de DPBC. Desta forma, apenas os elementos contemplados na fase de EDPS, diferentes modelos descritos serão trabalhos nesta proposta.

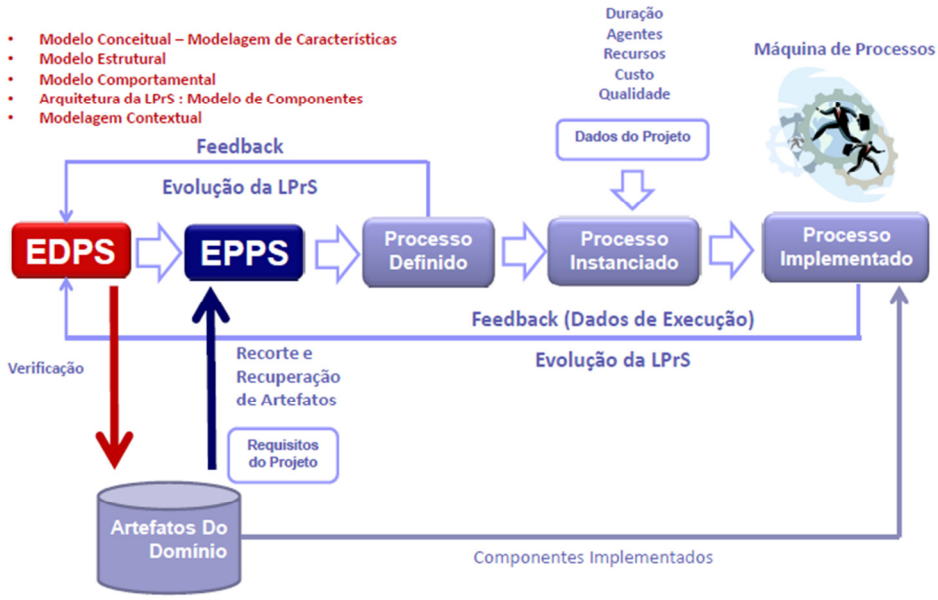



Figura 13 - Visão Geral de uma ELPrS

## 4.2. RESULTADOS PRELIMINARES

De acordo com a metodologia anteriormente descrita na Seção 1.4 do Capítulo I, alguns passos da pesquisa já foram elaborados, incluindo alguns resultados preliminares na parte de descrição do método, da especificação de representações de variabilidades em diferentes níveis de abstração (visão conceitual e estrutural), de mecanismos de mapeamentos entre os níveis de representação especificados e discussões iniciais sobre a complementação da representação através da visão comportamental e arquitetural. Tais resultados são apresentados nas seções a seguir.

### 4.2.1. Engenharia de Linha de Processos de Software (ELPS) com apoio ao Desenvolvimento de Processos Baseado em Componentes (DPBC)

A Engenharia de LPrS consiste no método que descreve as etapas que constituem o ciclo de desenvolvimento e aplicação de uma LPrS. É constituída de duas grandes fases: a EDPS, processo de desenvolvimento *para* reutilização e a EPPS, processo de desenvolvimento *com* reutilização. Neste trabalho, a fase de EDPS será desenvolvida descrevendo os artefatos que devem compor uma LPrS para promover a reutilização de processos de software.

O processo da fase EDPS é composto por três grandes atividades (Figura 14): (1) Análise do Domínio de Processos de Software (ADPS); (2) Projeto do Domínio de Processos de Software (PDPS) e (3) Implementação do Domínio de Processos de Software (IDPS). Tal processo deve prover a possibilidade de realizar revisões entre as atividades para manter a consistência e possibilitar refinamentos do domínio. Como proposta inicial, os artefatos são gerados com o avanço do processo pelos diferentes níveis de abstração, partindo do mais alto nível (ADPS), que trata da representação com entendimento mais conceitual da LPrS, até o mais baixo nível (IDPS) com os artefatos descritos por componentes lógicos representados em algum formalismo

de execução. À medida que os artefatos vão sendo construídos, rastros entre os diferentes níveis de abstração são registrados e mantidos pela abordagem. No entanto, há a possibilidade de um definição dos artefatos seguindo uma ordem definida pela necessidade do usuário. Para isso, os mecanismos de mapeamento entre os artefatos do domínio deverão ser bidirecionais permitindo que artefatos de mais baixo nível sejam mapeados para artefatos de mais alto nível. A viabilidade dessa proposta de mapeamento está sendo investigada.

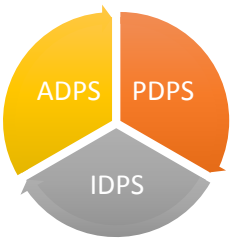



Figura 14 - Etapas da Fase de EDPS

A atividade de ***Análise do Domínio de Processos de Software*** (ADPS) se propõe a identificar e representar o conhecimento referente ao domínio através de um alto nível de abstração, representando o resultado da análise de suas similaridades, variabilidade e opcionalidades. De forma complementar, as especificações do domínio são detalhadas através do refinamento, com informações que envolvem a descrição das ações envolvidas nas diferentes possibilidades de execução das unidades de trabalho especificadas (passos); o estabelecimento de possibilidades de fluxo de execução, indicando sequencialidade, iteratividade, paralelismo; a descrição de ferramentas de apoio à execução; dentre outras. Na atividade de ***Projeto do Domínio de Processos de Software*** (PDPS), a Arquitetura de Componentes da LPrS é definida através da identificação e organização dos ativos do domínio, denominados de componentes de processos de software, estruturados através de relacionamentos e regras de comunicação via interfaces. Essa arquitetura pode vir a admitir a definição de variabilidades através de componentes não vinculados a uma única forma de realização, desde que atendam a um conjunto de restrições estabelecidas. Este tipo de variabilidade deverá ser avaliado ao decorrer do desenvolvimento do trabalho. A especificação do projeto arquitetural, em conjunto com a semântica dada pelo modelo de domínio, é a base para a atividade de ***Implementação do Domínio de Processos de Software*** (IDPS). O resultado corresponde à implementação de fato da LPrS, através da especificação dos componentes arquiteturais lógicos em uma representação executável (representação implementacional). Esta última etapa encontra-se fora do escopo desta tese.

As atividades iniciais do processo de EDPS consistem em *Identificação dos Escopos do Domínio de Processos de Software; Captura do Conhecimento do Domínio de Processos de Software; Análise de Similaridades e Diferenças do Domínio de Processos de Software; e Modelagem do Conhecimento do Domínio de Processos de Software* (Figura 15). As atividades

enumeradas devem ser realizadas sequencialmente. No entanto, existe a possibilidade de retorno à atividade anterior como forma de complementar à descrição dos elementos do domínio, através da ampliação do escopo, captura de mais informação sobre o domínio ou melhoria da análise do conhecimento adquirido. As atividades serão brevemente descritas abaixo, apontando algumas referências na literatura que podem apoiá-las. As três primeiras atividades não serão tratadas dentro do escopo desta tese. No entanto, a atividade de Modelagem do Conhecimento do Domínio de Processos de Software está sendo investigada e a definição de modelos para artefatos do domínio de uma LPrS está sendo realizada.

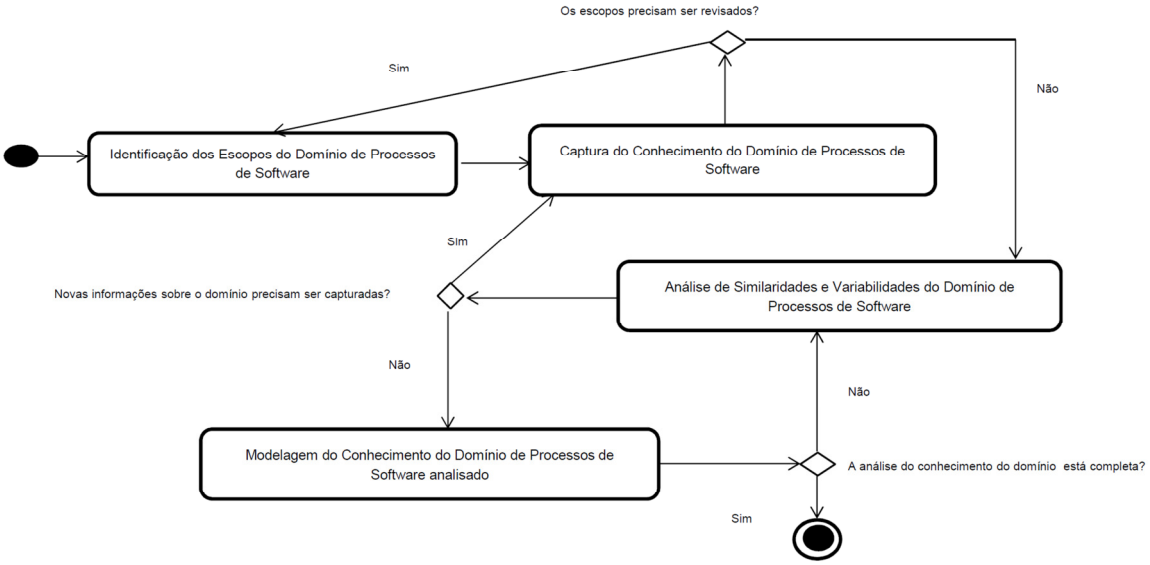



**Figura 15 - Atividades de Análise do Domínio de Processos de Software**

A atividade de ***Identificação dos Escopos do Domínio de Processos de Software*** visa situar o domínio com relação a seus escopos, delimitando as fronteiras do domínio e definindo subáreas, tendo como resultado um panorama geral do domínio no contexto da organização. A abordagem de ARMBRUST *et al*. (2008)(2009) têm como foco esta tarefa de Definição do Escopo de uma LPrS para caracterização sistemática de produtos, projetos e processos e a seleção de processos e elementos de processos para suportar de forma eficiente o desenvolvimento de produtos e a execução de projetos e minimizar o esforço do gerenciamento de processos. Determina o que incluir em uma LPrS baseado na análise de atributos de produtos, projetos e processos descritos em mapas.

A atividade ***Captura do Conhecimento do Domínio de Processos de Software*** visa capturar informações e conhecimento dentro de um domínio de processos de software. Encontra-se dividida nas seguintes tarefas: (1) *Identificação da(s) fonte(s) de conhecimento sobre o domínio*; (2) *Seleção da(s) técnica(s) de captura do conhecimento do domínio*; e (3) *Captura e Armazenamento do conhecimento do domínio* (Figura 16). Informações de processos podem ser capturadas de múltiplas fontes, das quais podemos citar: (1) Modelos de referências, tais como CMMI (CHRISSIS *et al.*, 2006), MPS.BR (SOFTEX, 2009) e ISO/IEC 12207 (ISO/IEC, 2007); (2)

Métodos ágeis, tais como XP (BECK, 2004) e Scrum (SCHWABER, 2004); e (3) Modelos de processos existentes dentro da organização ou o conhecimento recuperado de instâncias executadas dos projetos de desenvolvimento de software através da análise ou mineração de processos (*process mining*). A abordagem de mineração de processos consiste na descoberta de conhecimento do domínio a partir de repositórios de ambientes que armazenem informações sobre processos existentes. A captura de informações pode ser realizada através da mineração de repositórios de ambientes de gerência de tarefas e/ou gerência de projetos. Nesta área, técnicas similares às aplicadas na Engenharia Reversa de software podem ser utilizadas, como *clustering* e reconhecimento de padrões (GRECO *et al.*, 2008; BOSE e AALST, 2009), no entanto, adaptações devem ser realizadas para avaliar o tratamento de informações referente a execução de processos conduzidos em projetos anteriores. A manipulação de informações depende da capacidade de conhecimento para realizar a análise e combinação das informações.

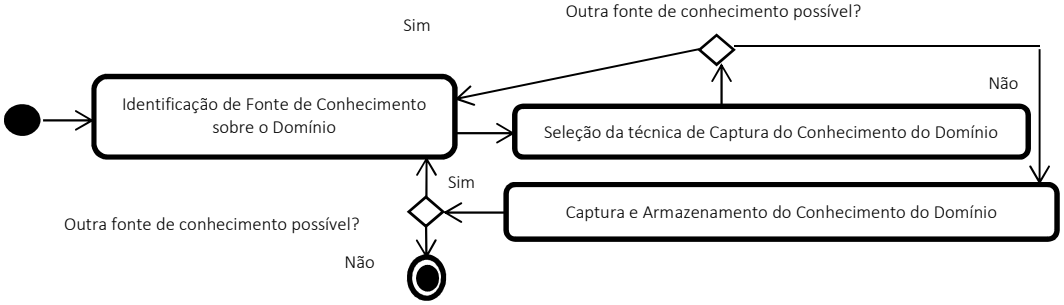



**Figura 16 - Etapas da atividade de Captura do Conhecimento do Domínio de Processos de Software**

Após essa atividade, deve ser conduzida uma ***Análise das Similaridades e Variabilidades do Domínio de Processos de Software***, que consiste na identificação de aspectos comuns e variáveis dentro do domínio de processos de software. Existem técnicas para realizar esta análise dentro da área de LPS, como exemplo a abordagem ArchToDSSA (KÜMMEL, 2007), que propõe a comparação de arquiteturas de aplicações, visando à detecção de suas similaridades, opcionalidades e variabilidades, para que, com base nestas informações, seja possível apoiar o Engenheiro de Domínio na criação de uma DSSA (*Domain Specific Software Architecture*). Essa abordagem utiliza como elementos um dicionário de sinônimos, a detecção semi-automática de variabilidades, e o apoio à definição dos elementos para compor a DSSA. Esta abordagem foi aplicada para o contexto de LPrS e a descrição das etapas para construir um exemplo de LPrS pode ser observada em MAGDALENO *et al*. (2011). Outra técnica foi desenvolvida por OCAMPO *et al*. (2005), que descreve a possibilidade de uma comparação entre processos de forma manual ou apoiada por regras com a utilização de suporte ferramental. O papel responsável pela análise consiste no engenheiro de processo, que deve comparar as entidades dos modelos de processos recuperados, entrevistar os proprietários dos processos (desenvolvedores, líderes de projetos) e identificar indícios de similaridades e

diferenças. Diferentes técnicas de comparação podem ser usadas e requerem identificar as partes que devem ser comparadas e como analisar seus elementos (critérios de comparação).

O objetivo desta atividade consiste na detecção de similaridades e diferenças (variações) entre os processos analisados. A partir das variações, as opcionalidades e variabilidades podem ser especificadas. A etapa de Detecção de Similaridades tem por objetivo identificar os elementos que representam similaridades dentro do domínio. O resultado desta etapa é insumo para a etapa de Detecção de Opcionalidades, que visa especificar os elementos que são considerados mandatórios e aqueles considerados opcionais dentro do domínio. Para isso, podemos levar em consideração o número mínimo de modelos necessários em que determinado elemento deve estar presente para ser considerado como mandatório. A etapa de Detecção de Variabilidades determina pontos dos modelos que são pontos configuráveis do domínio.

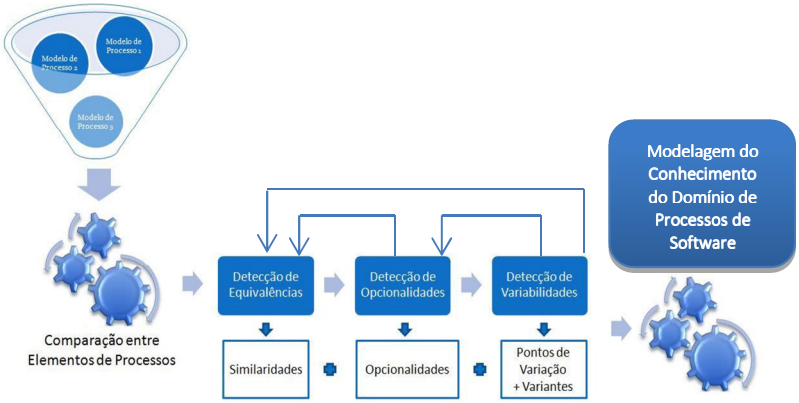



**Figura 17 - Análise de Similaridades e Variabilidades no Domínio de Processos de Software**

A atividade seguinte, foco desta tese, consiste na ***Modelagem do Conhecimento do Domínio de Processos de Software*** **(**Figura 18**)**. O domínio de processos pode ser entendido como uma coleção ou família de processos que compartilham um conjunto de aspectos comuns, ou seja, apresentam um conjunto de artefatos ou elementos de processos similares ou com objetivos similares. Desta forma, este conhecimento deve ser modelado através de uma especificação do resultado da análise das particularidades do domínio usando diferentes modelos, incluindo o Modelo de Domínio de Processos de Software em um alto nível de abstração (**Visão Conceitual da LPrS**). Este modelo deve ser expresso de forma a guiar a configuração de novos processos. Dentre os modelos de domínio, o mais conhecido na literatura de LPS é o modelo de características, o qual descreve a teoria do domínio (ARANGO e PRIETO-DIAZ, 1991). No contexto desta proposta de abordagem, a Visão Conceitual é representada através de modelo de características representados segundo o metamodelo e a notação *OdysseyProcess-FEX*, definidas em TEIXEIRA (2011). Tal representação foi adaptada neste trabalho para melhor atender a abstração ao nível que contempla.

Uma vez definido o primeiro nível de representação sobre o domínio de processos de software analisado, o processo indica duas atividades que podem ser realizadas de forma paralela e complementar: *Especificação da Visão Estrutural* e *Especificação da Visão Comportamental da LPrS*. A **Visão Estrutural** descreve a organização da estrutura de uma LPrS através da identificação das hierarquias de unidades de trabalho, com especificação e detalhamento de seus passos de realização, e associação dos papéis, produtos de trabalho e ferramentas envolvidos em suas execuções. A **Visão Comportamental** representa o fluxo de execução das unidades de trabalho especificadas na visão estrutural. Ambas podem ser derivadas a partir da Visão Conceitual e, desta forma, devem mapear as propriedades de variabilidade e opcionalidade e manter rastros com os elementos da modelagem de características. No entanto, deve ser possível realizar um mapeamento bidirecional entre os três níveis envolvidos (conceitual, estrutural e comportamental). Apesar de ser aconselhável, uma construção gradual da LPrS, alguns usuários da abordagem podem ter maior conhecimento sobre o domínio e prefiram começar a modelagem através da especificação estrutural ou comportamental. Desta forma, a atividade de Especificação da Visão Conceitual seria opcional ou poderia ser montada, posteriormente, através de um mapeamento reverso. A representação final da LPrS consiste em uma Arquitetura de Componentes da LPrS (**Visão Arquitetural**).

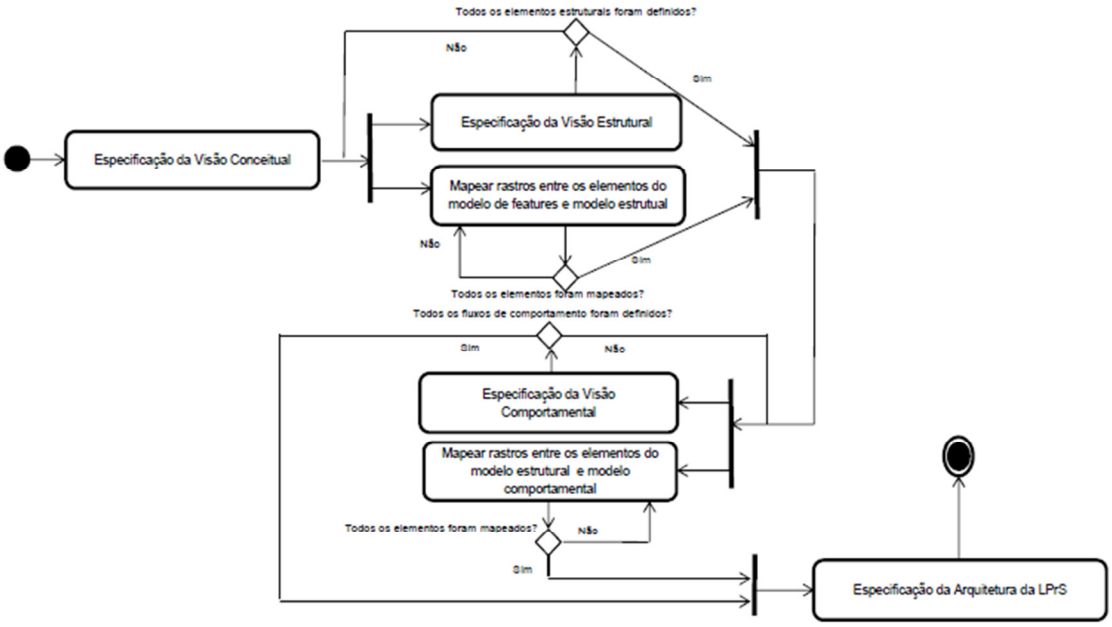



**Figura 18 - Atividades da Etapa de Modelagem do Conhecimento do Domínio de Processos de Software**

### 4.2.2. Modelagem de Variabilidades de Linha de Processos de Software (ELPS)

Diante da análise da literatura, pode-se enumerar um conjunto de observações para guiar a representação de uma LPrS e os conceitos que a contemplam. Dentre os elementos de processos de software mais referenciados pelos diferentes trabalhos de modelagem de

processos de software podemos destacar como os principais: atividade; tarefa; papel; produto de trabalho (artefato); e ferramenta.

Quanto à variabilidade, todos os elementos citados podem ser considerados passíveis de variação (configuração) diante de aspectos específicos de projeto. No entanto, o estabelecimento de alternativa de variação a um ponto de configuração deve respeitar os tipos dos elementos de processo. Desta forma, podemos listar os seguintes conjuntos de possibilidades de configuração em processos de software:

(1) *Atividade*: esse elemento pode ser classificado como ponto de variação e as suas alternativas de configuração (variantes) podem ser elementos de processos do tipo atividade ou tarefa.

(2) *Tarefa*: esse elemento pode ser classificado como ponto de variação e as suas alternativas de configuração (variantes) podem ser elementos de processos do tipo tarefa ou seus passos.

Atividades podem ser vistas como elementos de processos compostos por outras atividades ou tarefas. Sua forma de configuração pode ser entendida como a configuração obtida por um conjunto de atividades ou tarefas associadas. No caso do elemento de processo tarefa, a configuração pode ser observada pela combinação de elementos de processos do tipo passo, que constituem a definição da forma de execução das ações necessárias para atingir o propósito da tarefa.

(3) *Papel*: esse elemento pode ser classificado como ponto de variação e as suas alternativas de configuração (variantes) podem ser elementos de processos do mesmo tipo papel.

(4) *Produto de Trabalho*: esse elemento pode ser classificado como ponto de variação e as suas alternativas de configuração (variantes) podem ser elementos de processos do mesmo tipo produto de trabalho.

(5) *Ferramenta*: esse elemento pode ser classificado como ponto de variação e as suas alternativas de configuração (variantes) podem ser elementos de processos do mesmo tipo ferramenta.

Quanto à opcionalidade, todos os elementos citados acima podem ser classificados como *mandatórios* ou *opcionais*, dentro de um domínio de processos analisado.

Para tratar as diferenças que distinguem cada processo de uma LPrS é necessário criar mecanismos em que essas variações nos processo possam ser definidas (MARTÍNEZ-RUÍZ *et al*., 2008). O processo de reutilização de processos de software baseado em variabilidades necessita de uma modelagem apropriada. Tal modelagem pode estar focada em uma fase inicial de análise, como uma representação em um nível alto de abstração, ou focada em fases de projeto

e implementação com um detalhamento da família de processos e suas variabilidades em níveis mais baixos de abstração.

A notação para modelagem de variabilidade pode ser textual, gráfica ou um misto das duas formas. No entanto, a expressividade de uma representação visual permite a identificação mais fácil dos pontos de configuração dos processos a serem derivados. Em (TEIXEIRA, 2011), uma lista de requisitos desejáveis a uma notação que tenha por objetivo expressar variabilidade em LPrS foi definida a partir do conhecimento existente na área de modelagem de variabilidades em LPS, e estendida durante o desenvolvimento desta pesquisa:

(1) A representação deve ser gráfica, permitindo um reconhecimento claro dos pontos onde os processos são similares e dos pontos onde diferem, necessitando de tratamento específico;

(2) A representação deve explicitar os elementos de processos fixos (invariantes), os elementos de processos configuráveis (pontos de variação) e suas alternativas de configuração (variantes). Os tipos de elementos do ponto de variação e suas variantes devem ser compatíveis e apropriadamente relacionados (atividades devem possuir outras unidades de trabalho – atividades ou tarefas - como variantes; tarefas devem possuir outras tarefas como variantes; produtos de trabalho devem possuir produtos de trabalho como variantes; papéis devem possuir outros papéis como variantes e ferramentais devem possuir outras ferramentas como variantes).

(3) A representação deve explicitar o grau de obrigatoriedade e opcionalidade na seleção de diferentes tipos de elementos ao instanciar um processo específico de projeto (classificação de opcionalidade). Além disso, algumas relações entre unidades de trabalho (atividades e tarefas) e seus elementos de suporte (produtos de trabalho, papéis e ferramentas) podem ser classificadas como opcionais se a participação do elemento não é essencial para garantir a execução da unidade de trabalho associada.

(4) A representação deve explicitar os relacionamentos entre elementos;

(5) A variabilidade comportamental deve ser modelada através de alternativas de execução nos fluxos de controle e dados. Fluxos de controle definem o sequenciamento em que unidades de trabalho devem ser executadas. ***Fluxos alternativos*** e ***opcionais*** devem ser devidamente representados de forma a manter a consistência com a representação estrutural do processo;

(6) A representação deve explicitar relações de dependência e mútua exclusividade entre elementos, descrevendo a composição de processos expressando a necessidade da seleção conjunta de elementos de processos e quando não é permitida a combinação de determinados elementos;

(7) A classificação de variabilidade de elementos de processos é ortogonal a classificação de opcionalidade. Desta forma, uma característica pode receber uma classificação quanto à opcionalidade e outra classificação complementar quanto à variabilidade.

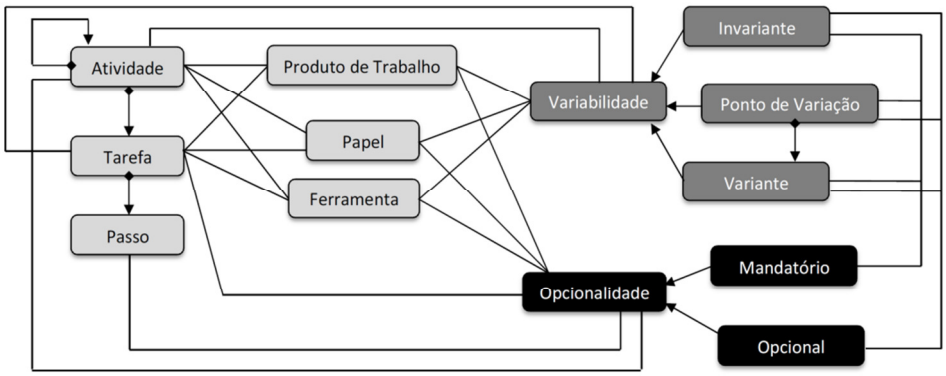



**Figura 19 - Principais elementos de processos passíveis de variação e os tipos de representação de variação**

A proposta desse trabalho está inserida no contexto de uma abordagem de LPrS que utiliza princípios da definição de processo baseada em componentes. A proposta de representação visa atender aos requisitos listados através de uma modelagem em diferentes níveis de abstração, visando permitir uma análise gradual dos diferentes aspectos que constituem uma LPrS. Para oferecer tal apoio, esta abordagem utiliza representações para cada uma das visões descritas no método acima: Visão Conceitual, Visão estrutural, Visão Comportamental e Visão Arquitetural (Figura 21).

A Visão Conceitual é representada através de um modelo de características que segundo o metamodelo e a notação *OdysseyProcess-FEX* (TEIXEIRA, 2011). Tal representação foi adaptada neste trabalho para melhor atender a abstração ao nível que contempla e será apresentada na Seção 4.2.2.1.

A Visão Estrutural deve ser representada através de modelos estruturais usando extensões do metamodelo do SPEM 2.0 (OMG, 2008). Tal representação é a mais difundida e popular para representar processos de software (RUIZ-RUBE *et al.*, 2012). No entanto, algumas extensões foram propostas para melhorar algumas deficiências no metamodelo, incluindo o suporte ao de LPrSs (RUIZ-RUBE *et al.*, 2012). Assim, com base no conhecimento existente nessas abordagens de extensão (MARTÍNEZ-RUIZ *et al.*, 2008; PAZIN, 2012; ALEGRÍA *et al.*, 2013) e algumas de suas limitações já analisadas no Capítulo 3, novas extensões são propostas neste trabalho para representar as variações possíveis nas diferentes estruturas dos processos que compõem uma linha, explicitando os pontos de variabilidade e opcionalidade de seus elementos e relações. A representação proposta é apresentada na Seção 4.2.2.2.

A Visão Comportamental deve expressar as variações nos fluxos de controle e de dados entre unidades de trabalho através de adaptações do diagrama de atividade da UML (OMG, 2005). Desta forma, as variações no comportamento dos múltiplos processos que compõem a

linha podem ser gerenciadas. A representação deste nível de abstração deve contemplar a especificação das diferentes possibilidades de sequências de execução das unidades de trabalho com a consistência da definição dos conceitos de variabilidade e opcionalidade. Com base em alguns trabalhos (RAZAVIAN E KHOSRAVI, 2008; SCHNIEDERS E PUHLMANN, 2005; PUHLMANN, 2005; SCHNIEDERS E PUHLMANN; 2006) que visam tratar esse tipo de variabilidade em modelos de processos usando extensões do diagrama de atividades da UML (OMG, 2011), esta abordagem visa utilizar a mesma representação com as adaptações necessárias para representação da visão comportamental. A UML é uma linguagem padrão, amplamente utilizada com recursos para extensão. O diagrama de atividade ainda é o mais utilizado em todas as abordagens para representação de fluxos de processos (BENDRAOU *et al.*, 2010).

O último nível, a Visão Arquitetural consiste na representação da LPrS através de uma Arquitetura de Componentes de Processos de Software. Esta proposta será trabalhada com base na abordagem de Projeto Arquitetural Baseado em Componentes no Contexto de Engenharia de Domínio definida por BLOIS (2006), realizando uma analogia do conceito empregado em LPS para o contexto de LPrS. De forma inicial, um componente de processo é definido como uma unidade de composição autocontida e claramente identificável, com comunicação via interfaces bem definidas. Descreve ou realiza uma função específica e de valor agregado ao processo que compõe através do encapsulamento de unidades de trabalho com nível de granularidade definido entre tarefa, atividade ou uma combinação desses elementos (Figura 20). Como proposto por LANNA (2009), é possível que componentes de mesmo nível de granularidade sejam agrupados para formar um componente de um nível de granularidade imediatamente superior. Por exemplo, vários componentes do tipo “Tarefa” podem ser agrupados para formarem um componente do tipo “Atividade”, que agrupado com outros componentes de mesmo tipo pode formar um componente do tipo “Processo”.

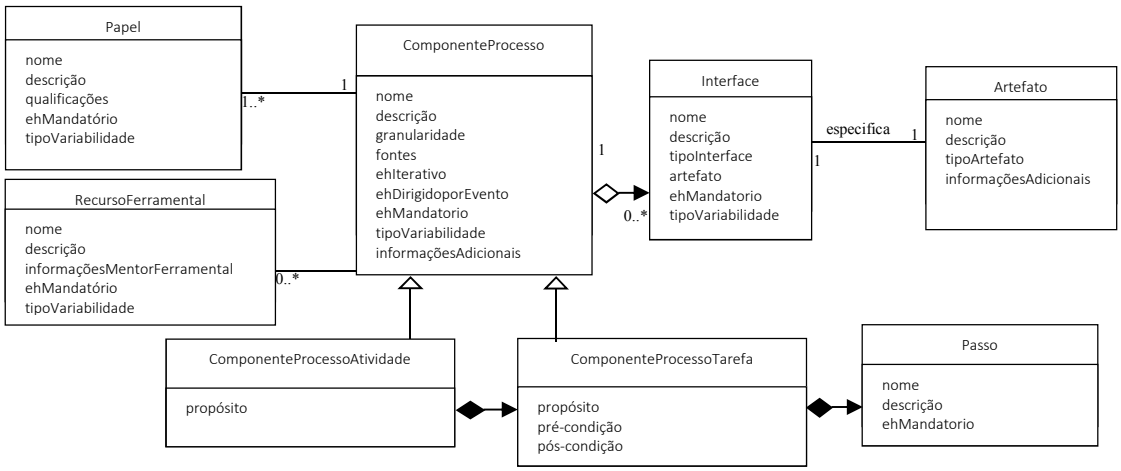



**Figura 20 - Proposta dos principais elementos para modelagem de componentes de processo de software**

Uma interface consiste no meio de comunicação do componente. Essas interfaces definem as portas do componente, indicando os elementos requeridos para execução do

componente e os resultados providos com sua execução. A Tabela 9 apresenta as definições dos principais elementos envolvidos.

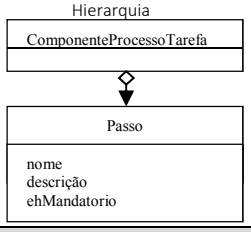
<table>
<tr><td>Elemento: ComponenteProcesso</td></tr>
<tr><td>Definição: Unidade de composição com interfaces bem definidas, representando o encapsulamento de unidades de trabalho a serem executadas, em um dado nível de granularidade, definido por Atividade e/ou Tarefa.</td></tr>
<tr><td>Atributos:<br>• nome: atributo que define o nome da característica.<br>• descrição: atributo que permite uma descrição textual sobre o componente.<br>• granularidade: indica a granularidade do componente. Pode assumir os seguintes valores:<br>Granularidade mínima: ComponenteTarefa<br>Granularidade média: ComponenteAtividade<br>Granularidade média: ComponenteProcesso<br>• fontes: lista de fontes de onde o componente foi identificado.<br>• ehIterativo: atributo é usado para definir a repetição do trabalho, por exemplo, as iterações.<br>• ehDirigidoporEvento: atributo usado para indicar que a execução do componente pode ser afetada pela ocorrência de um evento. Caso o atributo assuma o valor verdadeiro, especificações do tipo do evento e seu impacto na execução devem ser descritos para auxiliar na definição do comportamento do componente.<br>• ehMandatorio: atributo que define a classificação do componente quanto a sua opcionalidade. Indica se o componente é mandatório no domínio, isto é, se o componente estará presente em todos os processos instanciados a partir do modelo ou se o componente é Opcional.<br>• tipoVariabilidade: atributo que indica o tipo de variabilidade que o componente apresenta. Pode assumir os valores “Invariante”, “Variante” e “Ponto de Variação”.<br>• informaçõesAdicionais: atributo que permite a descrição de observações relacionadas ao componente.</td></tr>
<tr><td>Elemento: ComponenteProcessoTarefa</td></tr>
<tr><td>Definição: Componente de Processo que representa uma unidade fundamental de trabalho. Uma definição de tarefa provê explicações passo a passo de todo o trabalho que precisa ser executado para alcançar um objetivo, sem especificar quando e como esses passos devem ser executados em um processo instanciado. Desta forma, um ComponenteProcessoTarefa pode ser detalhado através de instâncias da classe Passo, que representa a especificação das partes que compõem uma tarefa.<br>Passo: Corresponde a uma especificação detalhada de como atingir o propósito definido pela tarefa a qual pertence.<br>Um passo pode ser classificado como mandatório ou opcional, isto é, se o passo estará presente em todas as diferentes execuções da tarefa ao qual está relacionado, ou não. Pode expressar alternativas de execução do fluxo principal da tarefa associada.</td></tr>
<tr><td>Atributos:<br>• propósito: atributo que estabelece a finalidade ou o objetivo a ser alcançado pela característica.<br>• pré-condição: atributo que estabelece um conjunto de restrições: condições que permitem o início da execução do componente.<br>• pós-condição: atributo que estabelece um conjunto de restrições: condições que permitem concluir a execução do componente, estabelecendo os resultados que precisam ser atingidos ao final desta execução.<br>Hierarquia<br> 
</td></tr>
<tr><td>Elemento: ComponenteProcessoAtividade</td></tr>
<tr><td>Definição: Componente de Processo que representa o agrupamento de unidades de trabalhos menores, especificadas por tarefas.</td></tr>
<tr><td>Atributos:<br>• propósito: atributo que estabelece a finalidade ou o objetivo a ser alcançado pelo componente.</td></tr>
<tr><td>Elemento: Interface</td></tr>
<tr><td>Definição: Representa o meio de comunicação de componentes de processos, descrevendo suas especificações. Corresponde a definição dos artefatos envolvidos na execução do componente. Especifica o tipo de artefato: requerido (entrada do componente) ou provido (saída do componente). Vale ressaltar que esta definição é válida para todos os níveis de granularidade de componentes de processo de software definidos, processo, atividade e tarefa. Apesar de componentes do tipo “atividade” serem maiores do que um componente do tipo “tarefa”, o consumo e a produção de artefatos ocorrerão através da execução das “tarefas” que formam os componentes maiores “atividades”.</td></tr>
<tr><td>Atributos:<br>• nome: atributo que define o nome da interface.<br>• descrição: atributo que permite uma descrição textual sobre a interface.<br>• tipoInterface: atributo que define o tipo da interface sendo descrita. Pode assumir os seguintes valores: “Requerida”, “Provida”.<br>• artefato: especifica o artefato relacionado à interface.<br>• ehMandatoria: atributo que define a classificação da interface quanto a sua opcionalidade. Indica se a interface (artefato) é mandatória na execução do componente em particular ao qual está associada, isto é, se o artefato estará presente em todas as diferentes execuções do componente, sendo indispensável para sua execução ou como resultado de sua execução, ou se o artefato é Opcional.<br>• tipoVariabilidade: atributo que indica o tipo de variabilidade que a interface apresenta. Pode assumir os valores “Invariante”, “Variante” e “Ponto de Variação”.<br>• A Variabilidade de um artefato pode ser definida através da aplicação de diferentes templates para especificar seu conteúdo e estrutura.<br>• A opcionalidade pode estar relacionada ao domínio como um todo ou apenas na participação da execução do componente de processo com o qual está relacionado.</td></tr>
</table>

**Tabela 9 - Definições dos principais elementos do metamodelo descrito na Figura 20.**

A definição de elementos arquiteturais deve ser suportada por sistemáticas para a organização de artefatos de processo reutilizáveis em elementos arquiteturais, os quais compõem uma arquitetura de componentes. Para isso é esperado o uso de critérios, métricas ou outras técnicas para apoio a criação destes elementos arquiteturais. Esse suporte deve ser definido neste trabalho de pesquisa.

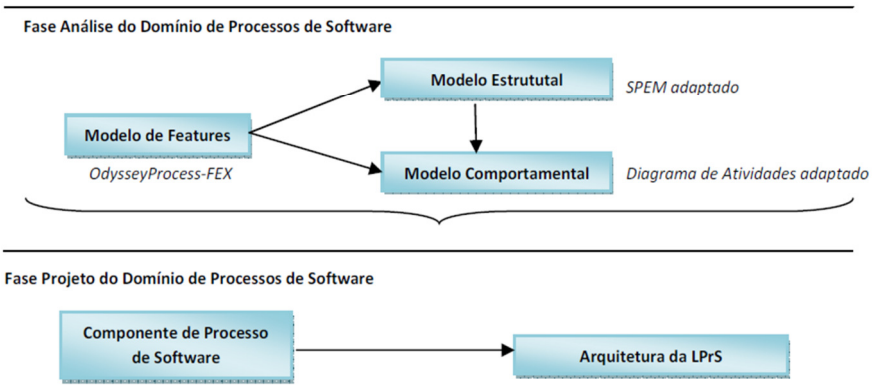



Figura 21 - Fases e Modelos da *OdysseySPrLE-PCBArch*

Para manter a consistência entre as diferentes representações envolvidas é necessário criar mecanismos de rastreamento entre os diferentes níveis estabelecidos. Um conjunto de heurísticas para apoiar tal mapeamento e manutenção da consistência de representações é proposto, estabelecendo relações entre os elementos dos diferentes níveis de abstração que compõem a linha.

Até o estágio atual da pesquisa os níveis conceitual e estrutural foram definidos com maiores detalhes e serão descritos nas seções 4.2.2.1 e 4.2.2.2. Os níveis comportamental e arquitetural encontram-se em fase de desenvolvimento e, por isso, não serão detalhados nesta proposta.

#### 4.2.2.1. Notação para Modelagem de Variabilidades da Visão Conceitual da LPrS – Modelagem de Características

O metamodelo *OdysseyProcess-FEX* foi proposto por TEIXEIRA (2011) visando representar LPrS usando um Modelo de Domínio de alto nível de abstração, um modelo de característica. Tal meta-modelo une a representação de conceitos de características, variabilidades e opcionalidades à representação dos elementos que constituem a definição de processos de software, como atividades, tarefas, papéis e produtos de trabalho. Os relacionamentos entre tais elementos são definidos de forma a montar uma estrutura conceitual de processos, além da especificação de relações de dependência e mútua exclusividade. O metamodelo possui um conjunto de restrições e propriedades, regras de boa formação que direcionam a construção e a verificação de consistência de um modelo de características do domínio de processos de software. A notação *OdysseyProcess-FEX* representa simbolicamente os conceitos formalizados pelo meta-modelo.

Neste trabalho, o meta-modelo foi revisado e algumas modificações foram propostas como forma de verificar o conjunto de elementos conceituais que precisam efetivamente estar presentes neste nível de abstração. O objetivo é definir os elementos de processos necessários para a representação de um modelo de características para o entendimento de um primeiro nível de modelagem de uma LPrS. Dentre as alterações realizadas podemos enumerar a retirada

de duas categorias de características que não contribuem para o entendimento do domínio (conceito e prática). Essas categorias podem ser colocadas como comentários associados ao modelo, e, para isso, o elemento Comentário (<<*Comment*>>) foi incluído. O elemento Comentário pode ser definido como uma anotação textual associada a um elemento com o objetivo de acrescentar alguma informação adicional para organização de conhecimento sobre o domínio. Outra alteração foi a reorganização hierárquica dos elementos classificados nas categorias Atividade e Tarefa como tipos de Unidades de Trabalho. O elemento Passo que era uma detalhamento da característica da categoria Tarefa foi eliminado por se tratar de uma descrição de forma de execução, mais apropriado para a visão estrutural. E a característica da categoria Disciplina foi reestrutruada, passando a ter a definição do conceito de pacote para organização do domínio em escopos menores ou subdomínios. Elemento que representa uma categorização de trabalho baseado em uma similaridade de interesses e propósito de resultados. O relacionamento Herança foi removido por não contemplar nenhuma especificidade de representação dentro do domínio de processos.

O metamodelo é composto por três pacotes: *Principal* (representa a taxonomia das características), *Relacionamentos* (representa as propriedades dos possíveis relacionamentos entre categorias de características) *e Regras de Composição* (estabelece a estrutura das regras que definem relações de dependência e mútua exclusividade). As características podem ser classificadas quanto à categoria (Tabela 10), variabilidade (invariante, ponto de variação ou variante) e opcionalidade (mandatória ou opcional). A representação gráfica da variabilidade é denotada através do relacionamento Alternativo entre o ponto de variação (origem da relação) e suas variantes (destinos da relação). A opcionalidade é graficamente representada através de um retângulo formado por uma linha tracejada para as características classificadas como opcionais.

| Categoria | | Ícone | Estereótipo |
|---|---|---|---|
| Atividade | Característica que representa o agrupamento de unidades de trabalhos menores, representadas por outras atividades ou por unidades elementares, especificadas por tarefas. |  | *<<activity>>* |
| Tarefa | Característica que representa uma unidade fundamental de trabalho. Sua granularidade é definida como uma unidade elementar. |  | *<<task >>* |
| Papel | Característica que representa um conjunto de habilidades, competências e responsabilidades de um indivíduo ou um conjunto de indivíduos. Não especifica um indivíduo ou recurso em particular. |  | *<<role>>* |
| Produto de Trabalho | Característica que representa um artefato consumido, modificado ou produzido por uma unidade de trabalho. |  | *<<work product>>* |

Tabela 10 - Categorias de características *OdysseyProcess-FEX*

Os relacionamentos entre as características incorporam relacionamentos da UML, relações que tornam explícitas a representação de variabilidades e relações específicas da área de processos de software (Tabela 11). Para cada relacionamento, um conjunto de restrições foi determinado de forma a especificar as possíveis combinações de categorias de características pertencentes ao relacionamento e algumas propriedades específicas.

| Relacionamento | | Representação |
|---|---|---|
| Alternativo | Relacionamento existente entre um ponto de variação e suas variantes. É representado por linhas simples entre Ponto de Variação e Variantes, interligadas por uma linha curva. |  |
| Agregação | Uma associação que representa uma agregação (isto é, um relacionamento de todo/parte). Possui a representação de associações binárias, mas diferencia-se por adicionar um diamante não-preenchido na extremidade agregada da linha de associação. |  |
| Composição | Representa um relacionamento de todo/parte mais forte do que agregação. Neste relacionamento, as partes não existem independentes do todo. Possui a representação de associações binárias, mas diferencia-se por adicionar um diamante preenchido na extremidade composta da linha de associação. |  |
| *Ligação PapelUnidadeDeTrabalho* | Estabelece um relacionamento de associação entre uma característica da categoria Unidade de Trabalho (Atividade ou Tarefa) e uma ou várias características da categoria Papel participantes na sua execução. |  <<workUnitRoleRelationship>> |
| *Ligação ProdutoDeTrabalho UnidadeDeTrabalho* | Estabelece um relacionamento de associação entre uma característica da categoria Unidade de Trabalho (Atividade ou Tarefa) e uma ou várias características da categoria ProdutoDeTrabalho que representam artefatos a serem consumidos, produzidos ou modificados pela execução da unidade de trabalho. |  <<workUnitWorkProductRelationship>> |
| *Ligação PapelProdutoDeTrabalho* | Estabelece um relacionamento de associação entre uma característica ProdutoDeTrabalho e uma ou várias características Papel com algum tipo de responsabilidade sobre o produto de trabalho. |  <<workProductRoleRelationship>> |

Tabela 11 - Relacionamento do meta-modelo *OdysseyProcess-FEX*

As Regras de Composição especificam a semântica de regras de dependência e de regras de mútua exclusividade entre características, especificadas através da estrutura: Antecedente + palavra-chave + Consequente. O antecedente e o consequente são expressões, que podem ser literais, expressões elementares que designa uma única característica, ou booleanas, que denotam uma combinação entre características através do uso de operadores booleanos: AND, OR, XOR, NOT. A palavra-chave apresentada na estrutura define o tipo de regra: Regras Inclusivas são denotadas por "*requer*", enquanto que Regras Exclusivas são denotadas por "*exclui*". As Regras de Composição são graficamente representadas por uma marcação, em todas as características pertencentes à regra, no canto inferior do retângulo que representa a característica envolvida na regra estabelecida. A marcação também apresenta uma numeração (n), que representa a ordem sequencial de criação das regras. Para Regras de Composição Inclusivas, a marcação corresponde a "R_n", no canto inferior direito. Para Regras de Composição Exclusivas, a marcação usada corresponde a "X_n", no canto inferior esquerdo.

Uma linha de processo de software de medição foi definida com base na LPrS especificada em CARDOSO, 2012 (Figura 22). No exemplo, as características Planejar Medição, Coletar e Armazenar Medidas, e Analisar Medidas Coletadas são exemplos de características mandatórias da categoria Atividade. As características Tratar Medidas Incorretas Coletadas e Realizar medições e análises adicionais são exemplos de características opcionais da categoria Tarefa. As características Planejar Medição e Reportar Resultados de Análise são exemplos de características classificadas como ponto de variação. As características Planejar Medição via GQM e Planejar Medição vis GQ(I)M são classificadas como variantes do primeiro ponto de variação definido. Tais variantes estão relacionadas ao ponto de variação Planejar Medição através do relacionamento Alternativo. Essas características são mutuamente excludentes entre si, o que pode ser observado pela marcação x no canto inferior esquerdo das características.

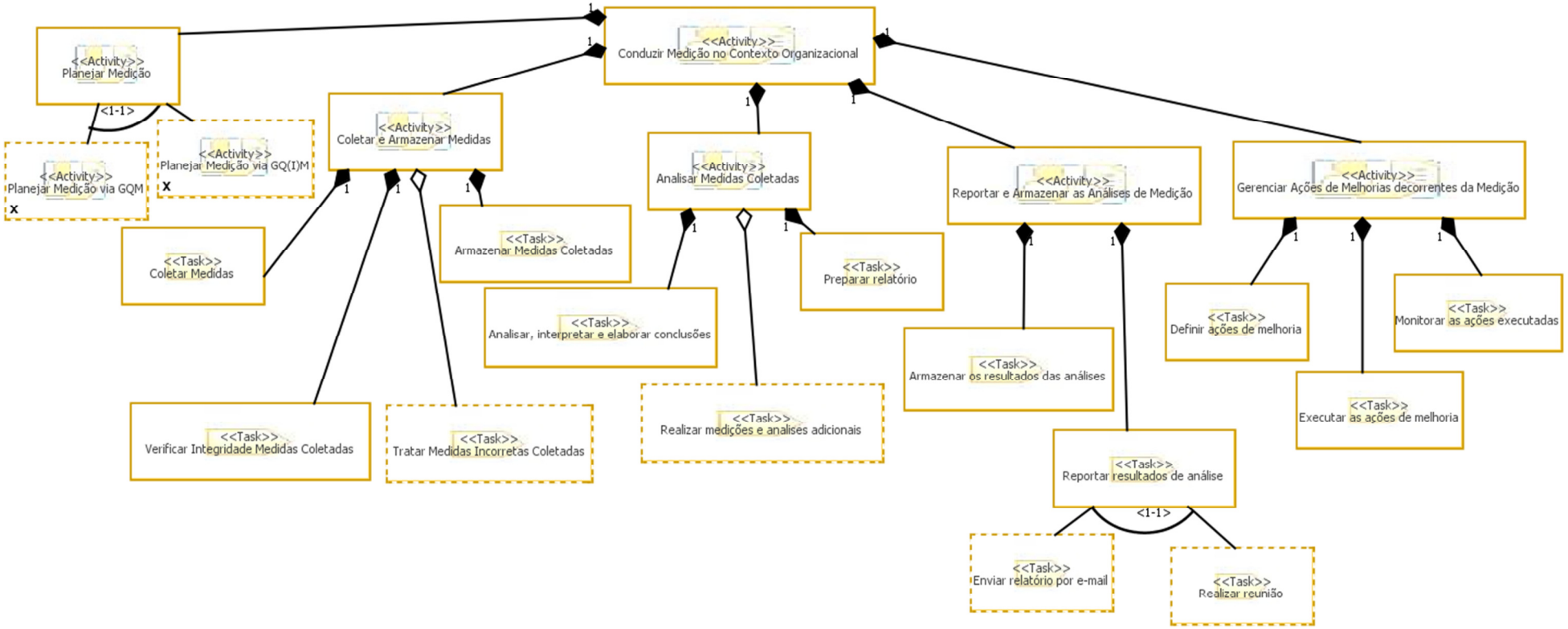



Figura 22 - LPrS Medição no Contexto Organizacional

#### 4.2.2.2. Notação para Modelagem de Variabilidades da Visão Estrutural da LPrS

A Visão Estrutural detalha como as unidades de trabalho dos processos de software da linha podem ser conduzidas através da definição dos passos envolvidos na sua execução e nas relações com outros elementos de processo (produtos de trabalho, papéis e ferramentas). Uma visão hierárquica da organização dos elementos de processos pode ser analisada neste nível de abstração. O possível apoio ferramental a unidades de trabalho também deve ser modelado neste nível, descrevendo mais detalhes de operação da família de processos.

A abordagem aqui apresentada foi definida com base em propostas (MARTÍNEZ-RUIZ *et al.*, 2008; PAZIN, 2012; ALEGRÍA *et al.*, 2013) para representação de variabilidades em LPrS com base em extensões no metamodelo SPEM 2.0 (OMG, 2008). O metamodelo SPEM 2.0 é um padrão definido pela OMG para especificação de processo de software. Possui boa expressividade para representação de elementos de processos (PORTELA *et al.*, 2012), sendo a representação mais difundida e popular para representar processos de software (RUIZ-RUBE *et al.*, 2012).

Neste sentido, esta proposta apresenta um conjunto de extensões ao metamodelo do SPEM 2.0 para representar a estrutura de múltiplos processos que compõem uma LPrS com a representação de variabilidades e opcionalidades dos elementos de processo, através de mecanismos explícitos e de fácil reconhecimento dos pontos de configuração existentes na estrutura de uma linha. As adaptações necessárias estão marcadas em negrito na Figura 23.

O tipo de variabilidade de um elemento de processo foi introduzido através do atributo inserido na classe <<*BreakdownElement*>>. A relação entre ponto de variação e suas variantes é determinada pelo relacionamento Alternativo incluído. Restrições de consistência foram definidas, especificando o tipo de elemento de processo que pode vir a ocupar um ponto de variação associado, de acordo com os requisitos listados na Seção 4.3.2. O conceito de opcionalidade já existia para alguns elementos de processo definidos no metamodelo (unidades de trabalho, produtos de trabalho e papel). Tal conceito foi introduzido ao elemento Passo e Ferramenta e a alguns relacionamentos, indicando a opcionalidade de participação de certos elementos na execução de unidades de trabalho associadas.

Desta forma, as classificações de opcionalidade e variabilidade das características de processos de software modeladas na visão conceitual, assim como relacionamentos existentes e suas opcionalidades, passam a ser representados na visão estrutural.

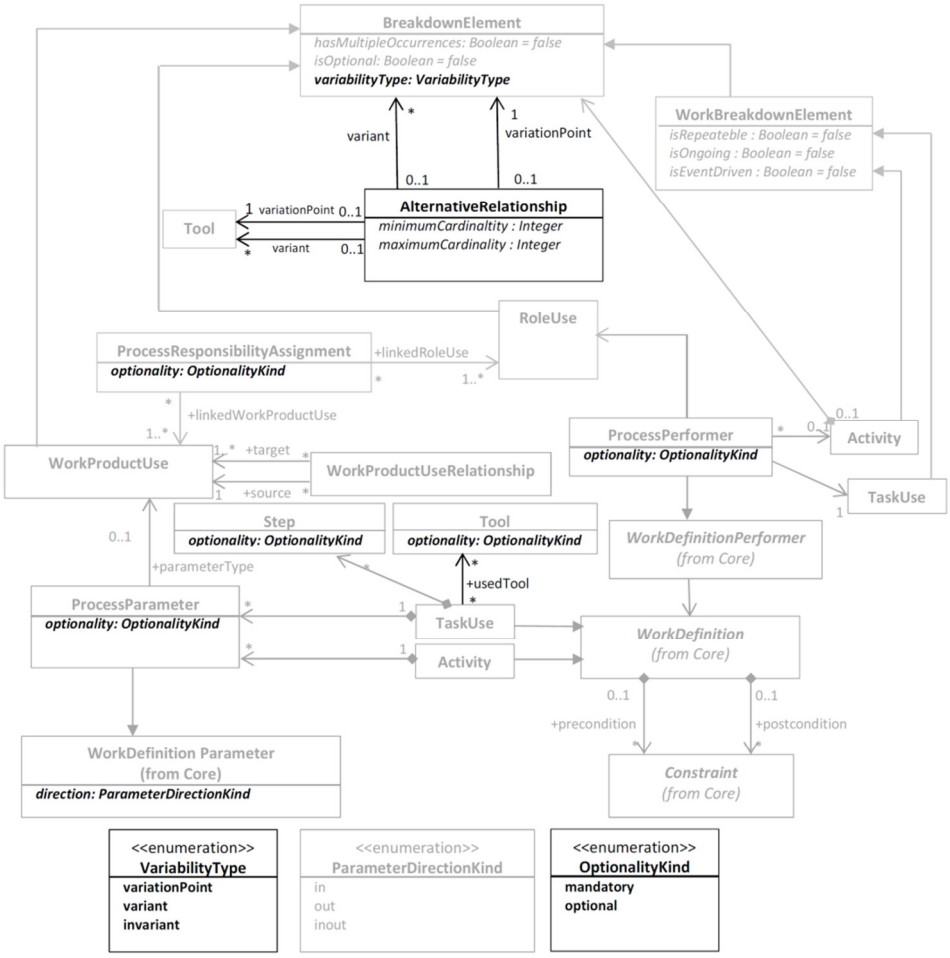



**Figura 23 - Principais classes e associações da adaptação do metamodelo SPEM 2.0 para tratamento de representação de conceitos na visão estrutural do nível de projeto.**

Como forma de evitar alterações significativas na notação já existente, optou-se pelo uso de estereótipos para representar explicitamente os conceitos de opcionalidade e variabilidade dos elementos de processo neste nível de abstração. De forma complementar a visão conceitual, regras de composição também são graficamente representadas através de estereótipos (Tabela 12).

| Categoria | Propriedade | Forma de Representação |
|---|---|---|
| **Variabilidade** | Ponto de Variação | Estereótipo <<***VP***>> |
| | Variante | Estereótipo <<***V***>> |
| | Invariante | - |
| **Opcionalidade** | Opcional | Estereótipo <<***optional***>> |
| | Mandatório | - |
| **Regras de Composição (RC)** | RC Inclusiva (RCI) | • Uso da restrição de pré-condição nos elementos de processo Atividade e Tarefa. Todos os elementos destes tipos descritos como antecedentes de uma RCI vão possuir a expressão consequente da regra como pré-condição. Observar a possibilidade de representação de relações de precedência nos relacionamentos de sequenciamento nos fluxos de trabalho.<br>• Estereótipo <<*R_n*>> nos elementos envolvidos e a descrição da regra associada. |
| | RC Exclusiva (RCE) | Estereótipo <<*X_n*>> nos elementos de processos envolvidos e a descrição da regra associada. |

**Tabela 12 - Estereótipos para tratamento de variabilidade, opcionalidade e regras de composição na representação da visão estrutural de uma LPrS**

Para manter a consistência entre os níveis de abstração especificados, um conjunto de mapeamentos foi definido como forma de rastrear os elementos do modelo de características (visão conceitual) para o conjunto de representações da visão estrutural. As ações de

mapeamento foram derivadas a partir da análise das definições dos elementos nos dois níveis de abstração, estabelecendo uma correspondência semântica entre os artefatos. Essa definição foi possível pelo metamodelo SPEM 2.0 ter sido usado como base de conhecimento para o desenvolvimento do metamodelo *OdysseyProcess-FEX*, garantindo um mapeamento semântico entre as definições dos elementos que constituem os metamodelos.

1. **Heurística *ConceitualEstrutural1* (CE1)**

Esta é uma heurística essencial, a base para que as outras heurísticas dessa fase de mapeamento sejam aplicadas. Essa heurística determina o mapeamento das diferentes categorias de características (atividade; tarefa; papel e produto de trabalho) para um elemento de processo correspondente utilizando a abordagem proposta (Tabela 13). As demais heurísticas assumem que este mapeamento foi realizado.

| **CaracterísticaAtividade (*ActivityFeature*)** | **Elemento de Processo correspondente: Atividade (*Activity*)** |
|---|---|
| Característica que representa o agrupamento de unidades de trabalhos menores, representadas por outras atividades ou por unidades elementares, especificadas por tarefas.<br>  | Uma Atividade é um elemento da estrutura de trabalho que define unidades de trabalho básicas dentro de um processo, como um próprio processo. Uma Atividade representa um agrupamento de elementos aninhados.<br>  <br>Analisar a Viabilidade do Projeto |
| **CaracterísticaTarefa (*TaskFeature*)** | **Elemento de Processo correspondente: Uso da Tarefa (*TaskUse*)** |
| Característica que representa uma unidade fundamental de trabalho. Sua granularidade é definida como uma unidade elementar.<br>  | Uso da tarefa é uma unidade de trabalho atribuível.<br>  <br>Elaborar Estudo de Viabilidade do Projeto |
| **CaracterísticaPapel (*RoleFeature*)** | **Elemento de Processo correspondente: Uso do Papel (*RoleUse*)** |
| Característica que representa um conjunto de habilidades, competências e responsabilidades relacionadas, não um indivíduo ou recurso em particular.<br>  | Uso do Papel descreve um conjunto de habilidades, competências e responsabilidades relacionadas.<br>  <br>Gerente do Projeto |
| **CaracterísticaProdutoDeTrabalho (*WorkProductFeature*)** | **Elemento de Processo correspondente: Uso do ProdutoDeTrabalho (*WorkProductUse*)** |
| Característica que representa um artefato consumido, modificado ou produzido por uma unidade de trabalho (atividade ou tarefa).<br>  | Uso do produto de trabalho descreve produtos consumidos, produzidos ou modificados por tarefas.<br>  <br>Estudo de Viabilidade do Projeto |

**Tabela 13 - Heurística *ConceitualEstrutural1* (Característica – Elemento de Processo)**

Uma característica da categoria tarefa mapeada para o elemento Uso da Tarefa pode ter seu conteúdo organizado através de partes ou subunidades denominadas passos. Um passo descreve uma parte significativa e consistente de todo o trabalho descrito por uma definição de tarefa (Tabela 14). A coleção de passos definidos para um elemento Uso da Tarefa representa todo o trabalho que deve ser feito para alcançar a meta global de desenvolvimento da definição da tarefa. Esse elemento de processo é apresentado nesse nível de abstração para um maior detalhamento de como executar o trabalho necessário para alcançar os propósitos de um processo.

| CaracterísticaTarefa (*TaskFeature*) | Elemento de Processo:<br>Uso da Tarefa (*TaskUse*) e Passos (*Step*) |
|---|---|
| Característica que representa uma unidade fundamental de trabalho. No nível de análise. a tarefa não é refinada em um conjunto de ações de granularidades menores que descrevem como executá-la.<br>  <br> | Um elemento Uso da Tarefa é detalhado através da descrição de partes significativas e consistentes (passos).<br>  <br> |

**Tabela 14 - Elemento passo da Visão Estrutural**

O elemento *Disciplina* representa o conceito de pacote para organização do domínio em escopos menores ou subdomínios. Tal elemento representa uma categorização de trabalho baseado em uma similaridade de interesses e propósito de resultados. Tal elemento do nível de análise possui um elemento correspondente de mesmo nome para o nível de projeto (Tabela 15).

| Disciplina (*Discipline*) | Elemento de Processo correspondente:<br>Disciplina (*Discipline*) |
|---|---|
| Representa o conceito de pacote para organização do domínio em escopos menores ou subdomínios. Elemento que representa uma categorização de trabalho baseado em uma similaridade de interesses e propósito de resultados. A disciplina é um conjunto de unidades de trabalho que estão relacionadas com uma grande área de interesse no âmbito do domínio como um todo.<br>  <br> | Uma disciplina é uma categorização de trabalho baseada em similaridades de conceitos e cooperação de esforço de trabalho. Uma disciplina é uma coleção de unidades de trabalho (atividades/tarefas) relacionadas por uma área maior de interesse. A separação dessas unidades em disciplinas auxilia o entendimento.<br>  <br> |

**Tabela 15 - Mapeamento do elemento Disciplina**

O elemento de processo Definição de Ferramenta (*Tool Definition*) do metamodelo SPEM 2.0 deve ser trabalhado neste nível de abstração com o intuito de descrever as capacidades de uma unidade de automação para fornecer apoio à execução de elemento de processo Uso da Tarefa (*TaskUse*) (Tabela 16). Tal elemento pode ser identificado como útil; recomendado ou necessário à execução da tarefa, através dos estereótipos <<*useful*>>, <<*recommended*>> e <<*necessary*>>, respectivamente.

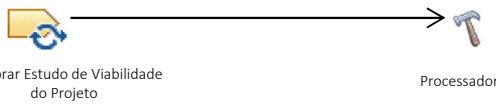
| Elemento de Processo Definição de Ferramenta (*Tool*) |
|---|
| Participação de uma ferramenta de apoio para a execução de uma tarefa. |
| Uma definição de ferramenta descreve as capacidades de uma ferramenta *CASE*, ou uma ferramenta de propósito geral, ou unidade de automação que fornece apoio à execução de uma tarefa.<br>  <br> |

**Tabela 16 - Elemento Definição de Ferramenta da Visão Estrutural**

2. **Heurística *ConceitualEstrutural2* (CE2)**

Classificações adicionais de opcionalidade e variabilidade de uma característica devem ser mapeadas para os elementos de processos correspondentes. Para cada elemento considerado opcional em todo o domínio, uma representação será adotada através da adição do estereótipo <<*optional*>> associado ao ícone do elemento (Tabela 17).

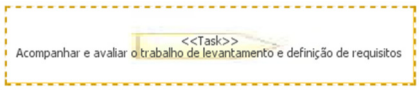
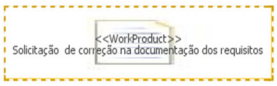
| Meta-modelo *OdysseyProcess-FEX* | Meta-Modelo *SPEM 2.0 estendido* |
|---|---|
| Atributo ***isOptional*** **(ehOpcional)** da classe *Feature* (Característica) – atributo que define a classificação da característica quanto a sua opcionalidade. Indica se a característica é mandatória no domínio, isto é, se a característica estará presente em todos os processos instanciados a partir do modelo ou se a característica é opcional. Default: *false*.<br>***isOptional = true*** | Atributo ***isOptional*** **(ehOpcional)** da classe *BreakdownElement* (Elemento decomposto) – atributo que indica que a inclusão de um elemento não é mandatória para execução de um projeto planejado segundo um processo derivado. Default: *false*.<br>***isOptional = true*** |
| **Classificação de opcionalidade de uma CaracterísticaAtividade *(ActivityFeature)*** | **Ícone para representar opcionalidade no elemento Atividade (*Activity*)** |
|  |  <br><<***optional***>><br>Acompanhar o levantamento e definição dos requisitos |
| **Classificação de opcionalidade de uma CaracterísticaTarefa *(TaskFeature)*** | **Ícone para representar opcionalidade no elemento da Tarefa (*TaskUse*)** |
|  |  <br><<***optional***>><br>Acompanhar e avaliar o trabalho de levantamento e definição de requisitos |
| **Ícone para representar opcionalidade no elemento Passo (*Step*)** | |
| Sem correspondente no nível de análise (metamodelo *OdysseyProcess-FEX*) |  <br><<***optional***>><br>Acompanhar reuniões de levantamento e definição de requisitos<br>  <br><<***optional***>><br>Avaliar documentação de requisitos modificada |
| **Classificação de opciconalidade de uma CaracterísticaPapel *(RoleFeature)*** | **Ícone para representar opcionalidade no elemento Uso do Papel (*RoleUse*)** |
|  |  <br><<***optional***>><br>Fornecedor de Requisitos |
| **Classificação de opcionalidade de uma CaracterísticaProdutoDeTrabalho*(WorkProductFeature)*** | **Ícone para representar opcionalidade no elemento Uso do Produto de Trabalho (WorkProductUse)** |
|  |  <br><<***optional***>><br>Solicitação de correção na documentação dos requisitos |
| **Ícone para representar opcionalidade no elemento Ferramenta (*Tool*)** | |
| Sem correspondente no nível de análise (metamodelo *OdysseyProcess-FEX*) |  <br><<***optional***>><br>Sistema de Gestão de Documentos e Controle de Versão |

**Tabela 17 - Heurística *ConceitualEstrutural2* (Classificação de opcionalidade)**

A classificação quanto à variabilidade permite especificar os pontos comuns e os pontos variáveis dentro do domínio representado pela LPrS. A classificação consiste em: (1) ponto de variação; (2) variante; e (3) invariante. O elemento invariante não apresenta variação dentro do domínio e será representado através dos ícones pertencentes à notação do metamodelo SPEM 2.0. Para cada elemento considerado ponto de variação, o estereótipo <<*VP*>> será adicionado. Para cada elemento variante, o estereótipo <<*V*>> será adicionado (Tabela 18).

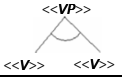
| Meta-modelo *OdysseyProcess-FEX* | Meta-Modelo *SPEM 2.0 estendido* |
|---|---|
| Atributo ***variabilityType*** **(tipoVariabilidade)** – atributo que indica o tipo de variabilidade que a característica apresenta. Pode assumir os valores *invariant* (invariante), *variant* (variante) e *variation point* (ponto de variação). O tipo de variabilidade é representado pela lista enumerada *VariabilityType (TipoVariabilidade)*. | Atributo ***variabilityType*** **(tipoVariabilidade)** da classe *BreakdownElement* (Elemento decomposto) – atributo incluído por esta abordagem para tratar de forma explícita o conceito de variabilidade no metamodelo SPEM2.0. O tipo de variabilidade é representado pela lista enumerada *VariabilityType (TipoVariabilidade)*, acrescentada nesta extensão. |
| Representação gráfica: Uso do relacionamento Alternativo (*Alternative*) | Representação gráfica: Uso do relacionamento Alternativo (*Alternative*) – representação acrescida à notação do metamodelo SPEM 2.0 e dos estereótipos <<***VP***>> para um elemento classificado como ponto de variação (*variationPoint*) e <<***V***>> para elemento classificado como variante (*variant*). |
|  |  |

**Tabela 18 - Heurística *ConceitualEstrutural2* (Classificação de variabilidade)**

Um elemento de nota (com o estereóitpo <<*variability*>>) pode ser associado a um elemento de processo classificado como ponto de variação com um conjunto de atributos

(Figura 24): *name* (nome do elemento); *minCardinality* (correspondente ao atributo *minimumCardinality* – cardinalidade mínima da classe *AlternativeRelationship* (relacionamento Alternativo) do metamodelo *OdysseyProcess-FEX e do metamodelo SPEM 2.0 estendido*); *maxCardinality* (correspondente ao atributo *maximumCardinality* – cardinalidade máxima da classe *AlternativeRelationship* (relacionamento Alternativo) do meta-modelo *OdysseyProcess-FEX e do met-modelo SPEM 2.0 estendido).*

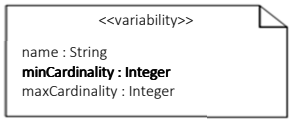



**Figura 24 - Elemento de nota <<*variability*>> que pode ser associado a um elemento de processo classificado ponto de variação**

No caso do ponto de variação ser classificado como *opcional*, o valor no item *minCardinality*, associado ao elemento de processo classificado como ponto de variação, assumiria o valor zero (Figura 25).

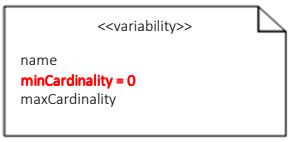



**Figura 25 - Elemento de nota *(<<variability>>)* associado a um ponto de variação opcional**

No caso do ponto de variação só ter a possibilidade de ser configurado utilizando apenas uma variante das opções a ele associadas, o valor no item *maxCardinality*, associado ao elemento de processo classificado como ponto de variação, assumiria o valor um. E para uma representação explícita, o ícone <<***XOR>>*** seria agregado a todos elementos variantes envolvidos.

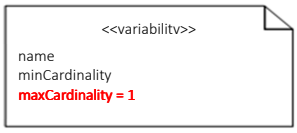



**Figura 26 - Elemento de nota <<*variability*>> associado a um ponto de variação com variantes mutuamente excludentes entre si**

Todas as categorias de características (*atividade, tarefa, papel, produto de trabalho*) derivadas da classe Característica (*Feature*) do metamodelo *OdysseyProcess-FEX* podem ser classificadas quanto à variabilidade. A mesma classificação pode ser atribuída aos elementos correspondentes no metamodelo SPEM 2.0 (atividade – *Activity*; tarefa – *TaskUse*; papel – RoleUse; produto de trabalho – *WorkProductUse*). O elemento passo (*Step*) introduzido no nível de projeto não possui esse tipo de classificação. O elemento ferramenta (*Tool*) possui esse tipo de classificação.

Algumas restrições devem ser seguidas para configurar um ponto de variação a um conjunto de tipos específicos de elementos. Elementos de processo do tipo atividade (*Activity*) definidos como ponto de variação podem estar associados apenas a elementos de processo do

tipo atividade (*Activity*) como variantes. Elementos de processo do tipo tarefa (*TaskUse*) definidos como ponto de variação podem estar associados apenas a elementos de processo do tipo tarefa (*TaskUse*) como variantes. Elementos de processo do tipo papel (*RoleUse*) definidos como ponto de variação podem estar associados apenas a elementos de processo do tipo papel (*RoleUse*) como variantes. Elementos de processo do tipo produto de trabalho (*WorkProductUse*) definidos como ponto de variação podem estar associados apenas a elementos de processo do tipo produto de trabalho (*WorkProductUse*) como variantes. Elementos de processo do tipo ferramenta (*Tool*) definidos como ponto de variação podem estar associados apenas a elementos de processo do tipo ferramenta (*Tool*) como variantes.

| Característica | Elemento de Processo<br>Estrutura de processo | |
|---|---|---|
| CaracterísticaAtividade (*ActivityFeature*) como Ponto de Variação Mandatório |  <br><<*VP*>> | Variantes: elementos de processo do tipo atividade (*Activity*)<br>  <br><<*V*>><br>  <br><<*V*>><br><<*optional*>> |
| CaracterísticaAtividade (*ActivityFeature*) como Ponto de Variação Opcional |  <br><<*VP*>><br><<*optional*>> | Variantes: elementos de processo do tipo atividade (*Activity*). Nenhum dos elementos associados como variante podem possuir a classificação de opcionalidade como mandatória.<br>  <br><<*V*>><br><<*optional*>> |
| CaracterísticaTarefa (*TaskFeature*) como Ponto de Variação Mandatório |  <br><<*VP*>> | Variantes: elementos de processo do tipo tarefa (*TaskUse*)<br>  <br><<*V*>><br>  <br><<*V*>><br><<*optional*>> |
| CaracterísticaTarefa (*TaskFeature*) como Ponto de Variação Opcional |  <br><<*VP*>><br><<*optional*>> | Variantes: elementos de processo do tipo tarefa (*TaskUse*). Nenhum dos elementos associados como variante podem possuir a classificação de opcionalidade como mandatória.<br>  <br><<*V*>><br><<*optional*>> |
| CaracterísticaPapel (*RoleFeature*) como Ponto de Variação Mandatório |  <br><<*VP*>> | Variantes: elementos de processo do tipo papel (*RoleUse*)<br>  <br><<*V*>><br>  <br><<*V*>><br><<*optional*>> |
| CaracterísticaPapel (*RoleFeature*) como Ponto de Variação Opcional |  <br><<*VP*>><br><<*optional*>> | Variantes: elementos de processo do tipo tarefa (*RoleUse*). Nenhum dos elementos associados como variante podem possuir a classificação de opcionalidade como mandatória.<br>  <br><<*V*>><br><<*optional*>> |
| CaracterísticaProdutoDeTrabalho (*WorkProductFeature*) como Ponto de Variação Mandatório |  <br><<*VP*>> | Variantes: elementos de processo do tipo produto de trabalho (*WorkProductUse*)<br>  <br><<*V*>><br>  <br><<*V*>><br><<*optional*>> |
| CaracterísticaProdutoDeTrabalho (*WorkProductFeature*) Ponto de Variação Opcional |  <br><<*VP*>><br><<*optional*>> | Variantes: elementos de processo do tipo produto de trabalho (*WorkProductUse*). Nenhum dos elementos associados como variante podem possuir a classificação de opcionalidade como mandatória.<br>  <br><<*V*>><br><<*optional*>> |
| - |  <br><<*VP*>> | Variantes: elementos de processo do tipo ferramenta (*Tool*)<br>  <br><<*V*>><br>  <br><<*V*>><br><<*optional*>> |
| - |  <br><<*VP*>><br><<*optional*>> | Variantes: elementos de processo do tipo ferramenta (*Tool*). Nenhum dos elementos associados como variante podem possuir a classificação de opcionalidade como mandatória.<br>  <br><<*V*>><br><<*optional*>> |

Tabela 19 - Heurística ConceitualEstrutural2 (Variabilidade em elementos de Processos de Software)

3. **Heurística *ConceitualEstrutural3* (CE3)**

Relacionamentos contemplados no modelo de características devem ser mapeados para o nível de projeto.

As ligações *WorkProductRoleRelationship* (*LigaçãoPapelProdutoDeTrabalho)* e *WorkUnitTaskRelationship* (*LigaçãoPapelTarefa)* podem ter o relacionamento classificado quanto a diferentes tipos de responsabilidades definidos através dos seguintes estereótipos: *<<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>.* O equivalente no metamodelo SPEM 2.0, seria especificar o tipo de responsabilidade através da classe *Kind* (qualificação da participação do papel).

A ligação *WorkUnitWorkProductRelationship* (*LigaçãoProdutoDeTrabalhoTarefa)* possui um conjunto de estereótipos que define o tipo de participação do produto de trabalho na tarefa relacionada: *<<in>>, <<out>>, <<inout>>.* O equivalente no meta-modelo SPEM 2.0 corresponde à classe *Parameter Direction Kind*.

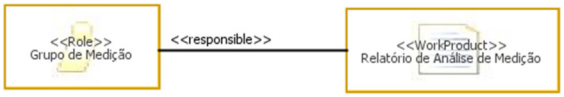
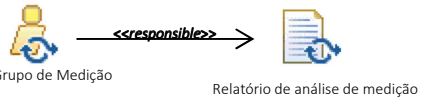
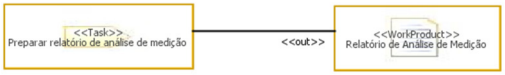
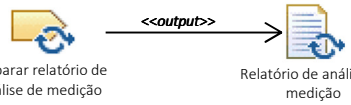
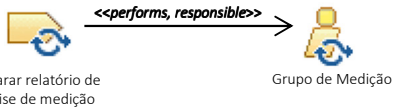
| ***WorkProductRoleRelationship* (LigaçãoPapelProdutoDeTrabalho)** | ***Process Responsibility Assignment*** (Atribuição de Responsabilidade de Processo) |
|---|---|
| Estabelece um relacionamento de associação entre uma característica da categoria Produto de Trabalho (*WorkProductFeature*) e uma ou várias características da categoria Papel (*RoleFeature*) com algum tipo de responsabilidade sobre o produto de trabalho.<br>  <br>* Tal relação possui o atributo *isOptional* (*ehOpcional)* que indica a opcionalidade da relação entre uma característica papel e um produto de trabalho associado. | O relacionamento de atribuição de responsabilidade de processo liga elementos do tipo papel (*RoleUse*) a elementos do tipo produto de trabalho (*WorkProductUse*), indicando uma relação de responsabilidade de um ou mais papéis a um produto de trabalho.<br>  <br>Tal relação possui o atributo *optionality* (opcionalidade) introduzido nesta abordagem como uma extensão para indicar a opcionalidade da relação na LPrS modelada. |
| ***WorkUnitWorkProductRelationship* (LigaçãoProdutoDeTrabalhoTarefa)** | ***Process Parameter*** (Parâmetro de Processo) |
| Estabelece um relacionamento de associação entre uma característica da categoria Unidade de Trabalho (Atividade ou Tarefa) e uma ou várias características da categoria ProdutoDeTrabalho que representam artefatos a serem consumidos, produzidos ou modificados pela execução da unidade de trabalho.<br>  <br>* Tal relação possui o atributo *ehOpcional* que indica a opcionalidade de participação (como artefato produzido, consumido ou modificado) da característica Produto de Trabalho associada a uma característica Tarefa. | Define um parâmetro de entrada e/ou saída para um ou um conjunto de produtos de trabalho (*WorkProductUse*) associados a um elemento do tipo tarefa (*TaskUse*) ou do tipo atividade (*Activity*).<br>  <br>Tal relação possui o atributo *optionality* (opcionalidade) introduzido nesta abordagem como uma extensão para indicar a opcionalidade da relação na LPrS modelada |
| ***WorkUnitRoleRelationship* (LigaçãoPapelTarefa)** | ***Process Performer*** (Executor de Processo) |
| Estabelece um relacionamento de associação entre uma característica da categoria Unidade de Trabalho (Atividade ou Tarefa) e uma ou várias características da categoria Papel participantes na sua execução.<br>  <br>* Tal relação possui o atributo *ehOpcional* que indica a opcionalidade de participação da característica Papel na característica Tarefa associada. | Representa uma relação entre um elemento de processo do tipo atividade (*Activity*) ou do tipo tarefa (*TaskUse*) e um ou mais elementos de processo do tipo papel (*RoleUse*).<br>  <br>Tal relação possui o atributo *optionality* (opcionalidade) introduzido nesta abordagem como uma extensão para indicar a opcionalidade da relação na LPrS modelada |

**Tabela 20 - Mapeamento dos relacionamentos específicos de processos**

### 4. Heurística *ConceitualEstrutural4* (CE4)

Regras de Composição representam restrições de dependência e mútua exclusividade.

| Regra de Composição Inclusiva (RCI): Indica uma relação de dependência entre dois ou mais elementos de processos. | |
|---|---|
| **RCI entre características das categorias atividade ou tarefa** | Para cada RCI, devem ser descritas pré-condições especificadas nos elementos de processos que equivalem às características que fazem parte do antecedente da regra.<br>Nesta descrição de pré-condição, deve ser registrado que o elemento de processo requer um ou um conjunto de outros elementos que equivalem ao consequente da regra de composição original. |
| **RCI** | Para cada RCE, deve existir um estereótipo <<*R_n*>> nos elementos de processo que correspondem às características envolvidas. O estereótipo corresponde ao ícone que representa uma regra. |
| Regra de Composição Exclusiva: Indica uma relação de mútua exclusividade entre dois ou mais elementos de processos. | |
| **RCE** | Para cada RCE, deve existir um estereótipo <<*X_n*>> nos elementos de processo que correspondem às características envolvidas. O estereótipo corresponde ao ícone que representa uma regra. |

**Tabela 21 - Heurística ConceitualEstrutural4 (Regras de Composição)**

Um exemplo complementando os elementos especificados na linha de medição representada pelo modelo de características na Figura 22 é apresentado (Figura 27). Neste exemplo, algumas atividades e tarefas são refinadas mostrando maiores detalhes de execução, por exemplo, Planejar Medição via GQM e Planejar Medição via GQ(I)M. A ferramenta de apoio ao armazenamento (Sistema de Banco de Dados) também foi representada.

## 4.3. Passos de Pesquisa

Os próximos resultados a serem atingidos consistem no desenvolvimento dos elementos que compõem os níveis comportamentais e arquiteturais. Os mapeamentos entre tais níveis e os anteriormente descritos também serão especificados. Os procedimentos de agrupamento de elementos em componentes e sua organização em uma estrutura arquitetural ainda serão desenvolvidos.

A implementação do suporte ferramental para as etapas descritas será conduzido de forma paralela a construção dos demais níveis de abstração.

Avaliações iniciais devem ser conduzidas para analisar a viabilidade de aplicação das modelagens propostas e a integração das diferentes visões da LPrS.

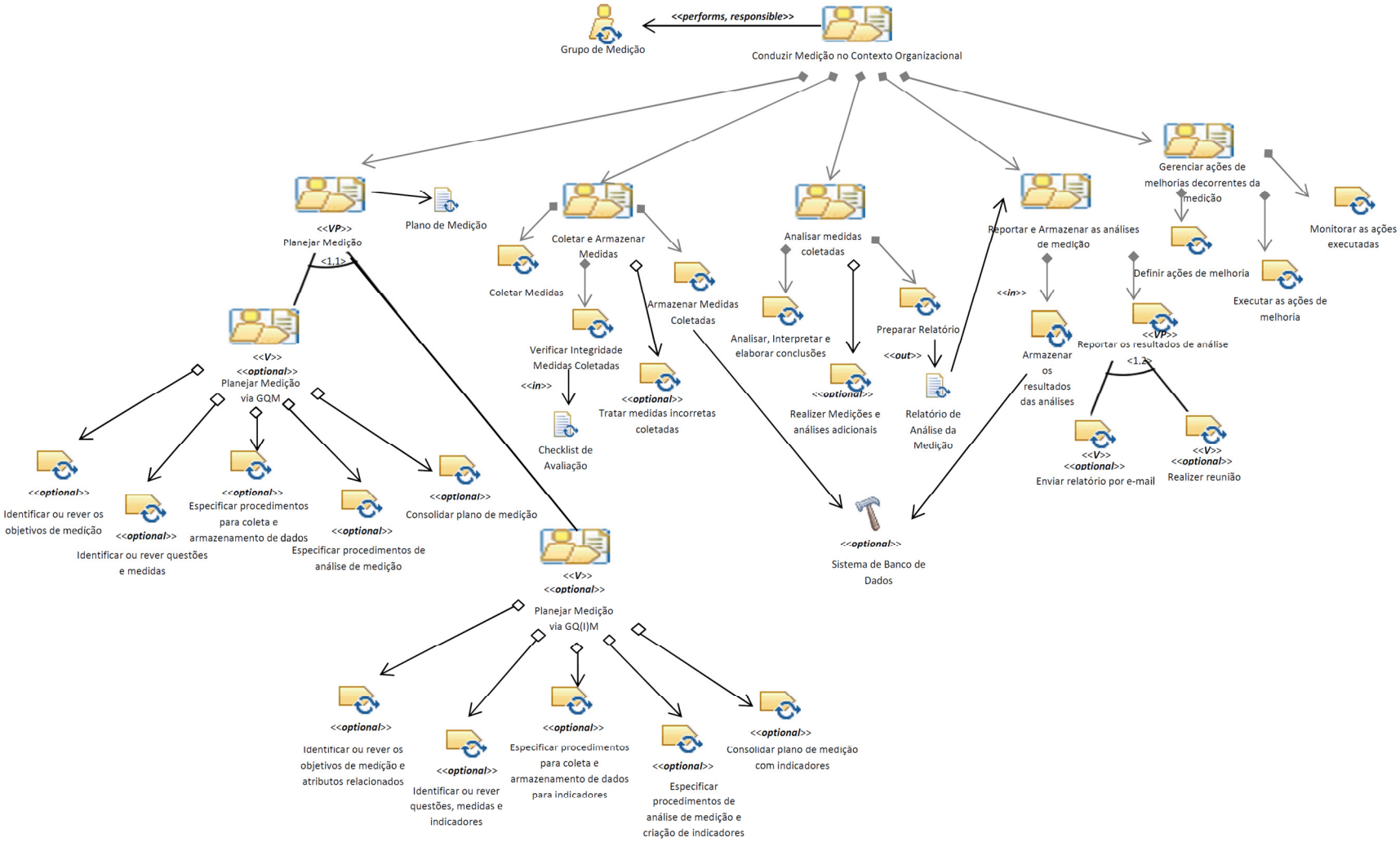



Figura 27 - Visão Estrutural da LPrS de Medição no Contexto Organizacional

# CAPÍTULO V
# CONCLUSÃO

## 5.1. EPÍLOGO

Diante da relação entre qualidade de produtos de software e os processos usados para construí-los, as organizações têm investido em especificar e melhorar seus processos de software. No entanto, a atividade de definição de processos de software é de extrema complexidade por lidar com especificidades de cada projeto e características particulares dos principais elementos envolvidos com a sua especificação, condução e monitoramento. Dentro desse contexto, técnicas de reutilização têm sido propostas como forma de auxiliar o engenheiro de processo e o gerente de projeto a tomar decisões mais adequadas na derivação de processos específicos de projeto. O propósito da área de reutilização de processos de software é facilitar a definição de processos, diminuindo o custo e o esforço associado à atividade, além de possivelmente aumentar a qualidade dos processos gerados, inclusive tornando a realização da atividade acessível a profissionais menos experientes (BARRETO, 2007). A dificuldade na reutilização de processos de software reside no fato que a variabilidade dos requisitos dos projetos frequentemente implica na variabilidade dos processos de software (ROUILLÉ *et al.*, 2013).

Diante do estudo realizado de duas técnicas de reutilização de processos de software (LPrS e DPBC), um desafio identificado consiste em promover a reutilização através de uma sistemática que trate as propriedades de variabilidades e opcionalidades inerentes a famílias de processos de software e estruturar o conhecimento do domínio através de uma arquitetura baseada em componentes de processos. A ideia central é unir os benefícios da área de LPrS com os da área de DPBC, promovendo a derivação de processos específicos de processos com base em conhecimento adquirido, adaptados a situações específicas através da combinação de blocos de construção, denominados componentes de processos de software.

As abordagens de LPrS existentes, em geral, não fornecem apoio a todo o ciclo de vida de processos e não pretendem combinar os aspectos das técnicas de Engenharia de Domínio e Componentes de Processos. Os trabalhos apontam diretrizes, mas não especificam técnicas concretas e sistemáticas dos processos de desenvolvimento *para* e *com* reutilização, nem contemplam a construção dos artefatos em diferentes níveis de abstração complementares (análise, projeto e implementação). Além disso, tais trabalhos não apresentam uma semântica completa para a representação das variabilidades do domínio. A organização da Linha de Processos de Software através de uma arquitetura de componentes de processos ainda não é devidamente apoiada. A definição de seus elementos arquiteturais deve ser suportada por

sistemáticas para o agrupamento de artefatos de processo reutilizáveis por meio de critérios, métricas ou outras técnicas.

A questão de pesquisa que orienta o desenvolvimento desta tese consiste em: *Como apoiar a definição de processos de software de forma sistemática, utilizando técnicas de reutilização de processos de software, combinando aspectos de adaptação, variabilidade e composição de processos de software?*

## 5.2. Principais Contribuições e Resultados Esperados

Diante do exposto, o objetivo central deste trabalho é definir, implementar e avaliar uma sistemática de reutilização de processos de software através de combinação das abordagens de LPrS e DPBC. A proposta envolve a definição de uma Engenharia de LPrS, um método sistemático para desenvolver e utilizar uma LPrS, especificando uma Arquitetura de Domínio Baseada em Componente de Processos de Software.

Tal abordagem é composta por cinco elementos principais: (1) um método; (2) uma representação para modelagem de variabilidades de uma LPrS; (3) mecanismos de rastreabilidade para estabelecer os mapeamentos entre os artefatos do domínio; (4) critérios de suporte a organização dos elementos que compõem a Arquitetura da LPrS baseada em componentes; e (5) uma infraestrutura de reutilização.

Esta proposta de abordagem tem como contribuições principais:

1. **Apoio à definição de processos específicos de projeto através de uma sistemática de reutilização.** A descrição dos procedimentos necessários tanto para construção quanto uso de uma LPrS visa fornecer apoio ao Engenheiro de Processos de Software no desenvolvimento e armazenamento dos artefatos do domínio de processos que serão reutilizados, e ao Gerente do Projeto, na definição do processo específico de um projeto com base no arcabouço oferecido. Para isso, cada etapa envolvida na fase de desenvolvimento e utilização da abordagem de Engenharia de Linha de Processos de Software deve ser especificada, contendo a descrição das atividades, de seus produtos esperados e dos mecanismos de solução e suporte à sua execução.

2. **Apoio à modelagem de variabilidades em processos de software.** O entendimento do conceito de variabilidades em processos de software e a análise das abordagens de modelagem visam fornecer conhecimento necessário para a definição de representações de modelagem do domínio de processos de software através de diferentes níveis de abstração de uma LPrS com suporte a representação de uma Arquitetura de Componentes de Processos de Software. A definição dos requisitos necessários para modelagem de variabilidades em LPrS e das notações para suportar uma representação em níveis

complementares deve apoiar a representação dos artefatos de processos de software reutilizáveis com maior entendimento pelos envolvidos na definição de processos. A representação deve contemplar tanto a parte estrutural de processos como comportamental (fluxos de controle e de dados), assim como sua organização em unidades de composição de processos de software passíveis de adaptação (componentes de processos).

3. Mapeamento dos diferentes artefatos do domínio através da especificação de mecanismos de rastreabilidade. A captura do relacionamento de uma dada abstração com um componente reutilizável deve ser feita através de um mecanismo denominado de "realização", permitindo sua descrição através de diferentes visões, desde representações abstratas de suas unidades de trabalho e conceitos, passando por um detalhamento maior, através de diagramas adicionais representativos de sua estrutura e seu comportamento, alcançando sua representação executável. Mecanismos de rastreabilidade bidirecionais devem ser estabelecidos para apoiar o mapeamento entre os artefatos do domínio, e viabilizar o entendimento facilitado do componente através da navegação entre os diversos modelos que o descrevem em um mesmo nível de abstração ou níveis complementares.

4. Representação de uma LPrS através de um arcabouço especificado por uma Arquitetura de Componentes de Processos de Software. Apesar de o uso de componentes para composição de processos aparentar fornecer muitas vantagens, partir de unidades tão pequenas para compor grandes processos parece ainda ser insuficiente. Por isso, o entendimento dos elementos que compõem uma LPrS deve ser apoiado por uma Arquitetura de Componentes de Processos de Software, que consiste em um arcabouço conceitual para descrever e relacionar elementos de processos. Tal arquitetura deve indicar a compatibilidade de um elemento de processo de software reutilizável em estruturas derivadas com base em sua organização. Uma definição do conceito de componente de processo de software deve ser derivada como resultado desta abordagem. Assim como o estabelecimento dos mecanismos de comunicação entre tais elementos. Critérios, métricas ou outras técnicas para apoio a criação de componentes arquiteturais são esperados.

5. A construção de apoio ferramental como infraestrutura de utilização da abordagem proposta. Levando em consideração a complexidade da atividade de definição de processos, ainda que apoiada por uma sistemática de reutilização, uma infraestrutura de apoio ao desenvolvimento e utilização de LPrS é prevista como contribuição desta abordagem. O apoio visa prover uma visão complementar e conjunta dos diferentes elementos que

compõem uma LPrS e fornecer algum apoio automatizado à utilização dos procedimentos definidos pelo método desta abordagem e a construção dos artefatos reutilizáveis e suas ligações. A fase de definição de um processo específico deverá ser também apoiada com um melhor entendimento dos impactos dos sucessivos recortes nos diferentes níveis de abstração. Este trabalho será desenvolvido dentro do contexto do ambiente Odyssey (ODYSSEY, 2014). A ideia é criar um ambiente unificado de apoio as diferentes fases da reutilização de processos de software.

6. Avaliação da viabilidade de aplicação da abordagem para o desenvolvimento e representação de uma LPrS através da execução de estudos experimentais e definição de mecanismos de verificação dos diferentes artefatos que a compõem.

## 5.3. Próximos Passos de Pesquisa

Um conjunto de passos de desenvolvimento da abordagem foi especificado visando apoiar a realização das atividades de desenvolvimento do trabalho de pesquisa.

1. Realização de uma revisão da literatura nas principais áreas envolvidas no desenvolvimento deste trabalho de pesquisa.

A revisão da literatura buscou criar um arcabouço teórico dos principais temas envolvidos. Desta forma, uma revisão *ad-hoc* da literatura foi conduzida como forma de entender áreas fundamentais sobre processos de software, incluindo a definição de processos de software, o ciclo de vida de processos de software, modelagem de processos de software, variabilidade de processos de software e uma visão geral da área de reutilização de processos de software. O resultado sumarizado do conhecimento adquirido foi apresentado no Capítulo 2 desta proposta.

A área de LPrS foi analisada através de um procedimento de revisão da literatura buscando reunir os trabalhos selecionadas em duas revisões sistemáticas anteriormente realizadas em abordagens correlatas a esse tema de pesquisa (BARRETO,2011; ALEIXO, 2013). Um conjunto de critérios, focados no tema de pesquisa conduzido neste trabalho, foi especificado e aplicado a maior parte do grupo de artigos identificados nas revisões citadas. Alguns artigos mais recentes da área foram incluídos e analisados durante o desenvolvimento desta proposta. O resultado desta análise foi apresentado no Capítulo 3. Através de um trabalho de parceria com uma equipe de pesquisados do Rio Grande do Norte, essa análise deve ser ainda estendida e seus resultados publicados ao longo do desenvolvimento da tese. A área de modelagem de variabilidade de processos de software foi especificamente analisada como forma de identificar requisitos para modelagem de LPrS e pontos ainda não trabalhados na

literatura existente, como a falta de apoio a variabilidade na parte comportamental de uma família de processos de software.

A área de Componentes de Processos de Software foi amplamente analisada em uma disciplina durante a formação inicial do doutorado. Durante esta análise forma identificadas diferentes abordagens e o resultado foi apresentado no Capítulo 3 desta proposta. De acordo com o andamento da pesquisa, novas análises desta literatura podem ser identificadas para complementar a definição de um dos níveis de abstração da abordagem proposta.

2. Especificação do método de Engenharia de LPrS, através da descrição das atividades de desenvolvimento de uma LPrS combinada com conceitos de DPBC, focando na definição dos artefatos que a compõem,

Algumas atividades já foram alcançadas como resultados preliminares desta proposta. No entanto, uma maior formalização dos procedimentos deve ser realizada com a descrição de todas as atividades através de diagramas de atividades e formulários complementares com informações adicionais para a execução das atividades previstas. Essa atividade visa descrever processo de desenvolvimento *para* reutilização (EDPS). Parte de tais procedimentos deve ser instanciada dentro da infraestrutura de reutilização a ser desenvolvida como forma de apoio aos usuários da abordagem.

3. Especificação da representação da LPrS, através da definição dos modelos em diferentes níveis de abstração com gerenciamento dos conceitos de opcionalidade e variabilidade.

Os requisitos para modelagem de variabilidades em LPrS foram definidos como resultado da análise da literatura realizada neste trabalho. Os níveis de abstração que correspondem a Visão Conceitual e a Visão Estrutural foram, inicialmente, definidos. Os outros níveis devem ser trabalhados e suas formas de representação devidamente especificadas.

4. Definição dos procedimentos de mapeamento entre os diferentes artefatos do domínio dos níveis de abstração.

Mecanismos de mapeamentos entre os artefatos dos diferentes níveis de abstração devem ser definidos de forma a atender uma rastreabilidade bidirecional para melhor apoiar usuários da abordagem e desenvolvedores da LPrS. Tal mecanismo deve permitir um melhor entendimento das múltiplas visões de uma LPrS e como os conceitos se complementam entre si. Mapeamentos preliminares entre as visões Conceitual e Estrutural já foram definidos e apresentados neste trabalho.

5. Definição de procedimentos para estruturar os elementos de processo de software em componentes da Arquitetura da LPrS.

A especificação do conceito de componente de processo de software e de Arquitetura de Componentes de LPrS deve ser estabelecido como resultado do trabalho. Critérios, métricas ou técnicas para organizar elementos de processos em componentes de processos de software devem ser especificados, assim como as formas de comunicação e relação desses componentes, como forma de criar o arcabouço arquitetural da LPrS.

6. Estabelecimento das estratégias de avaliação da abordagem e dos diferentes artefatos que a compõem.

O planejamento e execução de estudos experimentais deve ser conduzido de forma a apoiar a avaliação da viabilidade de aplicação da abordagem para o desenvolvimento e representação de uma LPrS com resultado final uma Arquitetura de Componentes de Processos de Software. A avaliação poderá apontar se houve maximização do potencial de reutilização da LPrS desenvolvida. Os resultados da realização dos estudos serão obtidos de forma gradual através da utilização da estratégia de validação sugerida pela metodologia de Shull *et al*. (2001), que prevê uma realização incremental de estudos experimentais de viabilidade, de observação, estudos de caso em um ciclo de vida e, finalmente, estudos de caso na indústria. De forma inicial, um exemplo de uso da abordagem deve ser planejado, seguindo os procedimentos descritos pelo método e as notações propostas nos diferentes níveis de abstração.

A verificação dos artefatos criados é de extrema importância para evitar a propagação de erros através da linha e dos processos a serem derivados. Desta forma, mecanismos de análise dos artefatos devem ser previstos como forma de avaliação. Para avaliar o atributo de reusabilidade dos artefatos de uma LPrS derivada, uma técnica de inspeção baseada em *checklist* está sendo construída. A visão conceitual pode ser avaliada através de um *checklist* derivado a partir da técnica FMCheck (DE MELLO *et al*., 2012), uma técnica de inspeção baseada em *checklist* para verificação semântica de modelos de características. Uma versão preliminar do *checklist* para verificação de modelo de características de LPrS pode ser observada no Anexo II. Para os outros níveis de abstração, *checklists* similares devem ser desenvolvidos com o objetivo de verificação e detecção de defeitos por método de inspeção. A Arquitetura de Componentes de Processos de Software da LPrS poderá ser avaliada através de um método derivado de um processo de inspeção da abordagem ArchMine (VASCONCELOS e WERNER, 2007). O critério para avaliação deve suportar a detecção de discrepâncias que possam comprometer a reusabilidade dos componentes de processo que compõem a arquitetura a ser analisada.

Outros estudos devem ser conduzidos visando analisar a viabilidade da aplicação das técnicas neste trabalho propostas.

7. Desenvolvimento da infraestrutura de reutilização, através da adaptação do ambiente Odyssey.

Este trabalho será desenvolvido dentro do contexto do ambiente Odyssey (ODYSSEY, 2014). O ambiente sofreu adaptações para suportar a construção de Linhas de Processos de Software, através da modelagem de características proposta em TEIXEIRA (2011), visando facilitar a representação das variabilidades em um alto nível de abstração. Novas extensões precisam ser realizadas. Para isso, um conjunto de avaliações do ambiente precisa ser realizado e as atividades de adaptação devem ser organizadas e realizadas para apoiar a construção de LPrS através dos diferentes artefatos que a compõem e apoio a sua utilização na derivação de um processo de software específico de projeto.

8. Publicação dos resultados do trabalho.

A proposta de tese foi apresentada no III Workshop de Teses e Dissertação de Software do CBSOFT.

- Teixeira, E. N. e Werner, C. (2013) "*OdysseyPrLE-CBArch: A Process Reuse Approach Combining the Software Process Line and Process Component Based Architecture Approaches*.", III Workshop de Teses e Dissertação de Software do CBSOFT (WTDSOFT 2013), pp. 86 - 92.

Os resultados alcançados com o andamento do trabalho devem ser publicados e discutidos com a comunidade acadêmica como forma de aprimorar a abordagem desenvolvida, através de eventos ou periódicos científicos de abrangência nacional e internacional.

- Simpósio Brasileiro de Qualidade de Software (SBQS);
- Simpósio Brasileiro de Engenharia de Software (SBES);
- *International Conference on Software Engineering* (ICSE);
- *International Conference on Software and Systems Process* (ICSSP);
- *International Conference on Product Focused Software and Process Improvement;*
- *European Systems and Software Process Improvement and Innovation* (EuroSPI); e
- *International Conference on Software Reuse* (ICSR).

## 5.4. Cronograma

| Atividades | 03/14 | 04/14 | 05/14 | 06/14 | 07/14 | 08/14 | 09/14 | 10/14 | 11/14 | 12/14 | 01/15 | 02/15 | 03/15 | 04/15 | 05/15 | 06/15 | 07/15 | 08/15 | 09/15 | 10/15 | 11/15 | 12/15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Defesa do Exame de Qualificação** | | | | | | | | | | | | | | | | | | | | | | |
| Refinamento do Método: Descrição da EDPS | | | | | | | | | | | | | | | | | | | | | | |
| Nível Comportamental: Descrever os requisitos comportamentais de uma LPrS e Especificar a notação para representação de variabilidade da Visão Comportamental | | | | | | | | | | | | | | | | | | | | | | |
| Nível Arquitetural: Definir o conceito de *Componente de Processos de Software* | | | | | | | | | | | | | | | | | | | | | | |
| Definir procedimentos de organização dos elementos de processo em componentes de processos de software | | | | | | | | | | | | | | | | | | | | | | |
| Definir o conceito de Arquitetura de LPrS | | | | | | | | | | | | | | | | | | | | | | |
| Definição do Procedimento de Mapeamento Bidirecional entre Visão Conceitual X Visão Comportamental | | | | | | | | | | | | | | | | | | | | | | |
| Definição do Procedimento de Mapeamento Bidirecional entre Visão Estrutural X Visão Comportamental | | | | | | | | | | | | | | | | | | | | | | |
| Definição do Procedimento de Mapeamento para a Visão Arquitetural | | | | | | | | | | | | | | | | | | | | | | |
| Desenvolvimento da Infraestrutura de Reutilização de Processos de Software: Implementação no Ambiente Odyssey | | | | | | | | | | | | | | | | | | | | | | |
| Definição do Procedimento de Verificação dos Artefatos – Construção dos *Checklist* de Avaliação | | | | | | | | | | | | | | | | | | | | | | |
| Exemplo de Uso da Abordagem | | | | | | | | | | | | | | | | | | | | | | |
| Planejamento e execução de avaliações complementares | | | | | | | | | | | | | | | | | | | | | | |
| Publicação de Resultados: Escrita de Artigos | | | | | | | | | | | | | | | | | | | | | | |
| Defesa da Tese | | | | | | | | | | | | | | | | | | | | | | |

## 5.5. Considerações Finais

Dada a complexidade envolvendo a definição de processos de software específicos de projeto, uma sistemática de reutilização de processos pode assistir tal atividade introduzindo benefícios via uma reutilização planejada que combina conceitos de LPrS e DPBC. Os objetivos do uso de uma Engenharia de LPrS são, assim como todas abordagens de reutilização, melhorar a previsibilidade, reduzir custo, tempo e riscos (ROMBACH, 2006). A combinação das duas técnicas de reutilização visa melhorar o apoio à reutilização e a atividade de definição de processos de software, como já realizado no contexto de LPS (BLOIS, 2006).

ACUÑA, S., FERRÉ, X., 2001, "Software Process Modelling". In: Proceedings of the 5th. World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001). Orlando Florida, USA, pp. 1-6.

ALEGRÍA, J.A.H., BASTARRICA, M. C., QUISPE, A., OCHOA, S. F., 2011, "An MDE approach to software process tailoring". In: *Proceedings of the 2011 International Conference on Software and Systems Process*, pp. 43-52.

ALEGRÍA, J.A.H., BASTARRICA, M.C., 2012, "Building software process lines with CASPER". In: Proceedings of International Conference on Software and System Process (ICSSP), pp. 170 –179.

ALEGRÍA, J. A.H., BASTARRICA, M. C., QUISPE, A., OCHOA, S. F., 2013, "MDE-based process tailoring strategy". *Journal of Software: Evolution and Process*.

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

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

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

AMBLER, S.W., 1998, "Process Patterns: Building Large-Scale Systems Using Object Technology", New York, United States, Cambridge University Press.

ARANGO, G., PRIETO-DIAZ, R., "Introduction and Overview: Domain Analysis Concepts and Research Direction". In: G.Arango, R.P.-D.A. (eds), Domain Analysis and Software Systems Modeling, IEEE Computer Society Press, pp. 9-25, 1991.

ARMBRUST, O.; KATAHIRA, M.; MIYAMOTO, Y.; *et al.*, 2008, "Scoping Software Process Models - Initial Concepts and Experience from Defining Space Standards", In: Proceedings of the International Conference on Making Globally Distributed Software Development a Success Story, Berlin / Heidelberg: Springer, pp. 160-172.

ARMBRUST, O.; KATAHIRA, M.; MIYAMOTO, Y.; et al., 2009, "Scoping Software Process Lines". In: Software Process: Improvement and Practice, v. 14, Issue 3, pp. 181-197, New York, NY, USA.

ARMBRUST, O., 2010, "Determining Organization-Specific Process Suitability". In: *New Modeling Concepts for Today's Software Processes*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, p. 26–38.

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

BASILI, V. R.; ROMBACH, H. D., 1987, "Tailoring the Software Process to Project Goals and Environment". In: Proceedings of the 9th International Conference on Software Engineering (ICSE'87), Monterey, CA. pp. 345-357.

BARRETO, A., 2007, "Uma Abordagem para Definição de Processos de Software Baseada em Reutilização". Exame de Qualificação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BARRETO, A. S., MURTA, L. G. P., ROCHA, A. R., 2009, "Componentizando Processos Legados de Software Visando a Reutilização de Processos", In: VIII Simpósio Brasileiro de Qualidade de Software, pp. 189-203, Ouro Preto, MG, Brasil, Junho.

BARRETO, A. S.; NUNES, E.; ROCHA, A. R. C.; MURTA, L. G. P., 2010, "Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines". In: *Proceedings of the International Conference on the Quality of Information and Communications Technology* (QUATIC), pp. 15-24, Porto, Portugal.

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

BARROS, M., 2001, "Gerenciamento de Projetos baseado em Cenários: Uma abordagem de modelagem dinâmica e simulação", Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BECK, K., 2004, Extreme Programming Explained: Embrace Change. 2 ed. Boston, MA, USA, Addison-Wesley.

BENDRAOU, R., JEZEQUEL, J., GERVAIS, M. P., BLANC, X., 2010, "A comparison of six uml-based languages for software process modeling". In: *Software Engineering, IEEE Transactions on*, *36*(5), 662-675.

BERGER, P. M., 2003, Instanciação de Processos de Software em Ambientes Configurados na Estação TABA. Dissertação de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BERTOLLO, G., SEGRINI, B., FALBO, R. de A., 2006, "Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias". In: V Simpósio Brasileiro de Qualidade de Software, SBQS, 6, p.72–86.

BHUTA, J.; BOEHM, B.; MEYERS, S., 2005, "Process Elements: Components of Software Process Architectures." In: Proceedings of the Software Process Workshop, pp. 332-346, Beijing, China.

BOSCH, J., 2000, *Design and use of software architectures: adopting and evolving a product-line approach*, 1st ed. New York, United States, Addison Wesley Professional.

BOSCH, J., 2004, "Software Variability Management". In: *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, pp. 720-721, Scotland, UK.

BLOIS, A. P., 2006, "Uma Abordagem de Projeto Arquitetural baseado em Componentes no contexto de Engenharia de Domínio". Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

BOSE, R.P.J.C., VAN DER AALST, W.M.P., 2009, “Abstractions in Process Mining: A Taxonomy of Patterns”. In: Business Process Management (BPM). V. 5701 of LNCS, pp. 159–175.

BRAGA, R.M.M., 2000, “Busca e Recuperação de Componentes em Ambientes de Reutilização de Software”, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.

BROWN, A., 2000, Large-Scale Component-Base Development, Prentice Hall.

CARDOSO, F.S., 2012, “Definição de Processos Reutilizáveis para Projetos com Aquisição”, Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

CHRISSIS, M. B., KONRAD, M., SHRUM, S., 2006, CMMI: Guidelines for Process Integration and Product Improvement. 2 ed. Boston, USA, Addison-Wesley.

COSTA, A., SALES, E., REIS, C.A.L., *et al.*, 2007, "Apoio a Reutilização de Processos de Software através de Templates e Versões". In: *VI Simpósio Brasileiro de Qualidade de Software*, pp. 47-61, Porto de Galinhas, Brasil, Junho.

COULETTE B., CRÉGUT X., DONG T. B. T. AND TRAN D. T., 2000, “RHODES, a Process Component Centered Software Engineering Environment”, In: *Proceedings of the 2nd International Conference on Enterprise Information System (ICEIS2000)*, July.

COULETTE B., CRÉGUT X., DONG T. B. T. AND TRAN D. T., 2001, “Managing process through a base of reusable components”, ICEIS2001, Setubal, July.

CUGOLA, G., GHEZZI, C., 1998, “Software Processes: a Retrospective and a Path to the Future”. In: Software Process Improvement and Practice. p. 101–123.

CURTIS et al., 1992, “Process Modelling”. In: Communications of the ACM, New York, Vol. 35, N. 9, pp. 75-90.

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

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

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

DE MELLO, R. M., TEIXEIRA, E. N., SCHOTS, M., WERNER, C. M. L., TRAVASSOS, G. H., 2012, “Checklist-based Inspection Technique for Feature Models Review”. In: *Sixth Brazilian Symposium on Software Components Architectures and Reuse (SBCARS),* pp. 140-149.

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

DURÁN, A., BENAVIDES, D., BERMEJO, J., 2004, “Applying system families concepts to requirements engineering process definition”. In: *Software Product-Family Engineering*, pp. 140-151. Springer Berlin Heidelberg.

ELLMER, E., MERKL, D., QUIRCHMAYR, G.*, et al.*, 1996, "Process Model Reuse to Promote Organizational Learning in Software Development". In: *Annual International Computer Software And Applications Conference, Compsac*, pp. 21-26.

FALBO, R. A.; BERTOLLO, G., 2005, "Establishing a Common Vocabulary for Software Organizations Understand Software Processes". EDOC International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE), pp. 1-8, Enschede, The Netherlands.

FEILER, P.H., HUMPHREY, W.S., 1993, "Software process development and enactment: Concepts and definitions". In: Proceedings of the Second International Conference on Software Process, pp. 28-40.

FIORI, D. R.; GIMENES, I. M. S.; MALDONADO, J.; OLIVEIRA JUNIOR, E. A., 2012, "Variability Management in Software Product Line Activity Diagrams". In: *International Conference on Distributed Multimedia Systems* (DMS), p. 89-94.

FUGGETTA, A., 2000, "Software Process: A Roadmap". In: Proceedings of the Conference on The Future of Software Engineering. ICSE '00. New York, NY, USA: ACM, p. 25–34.

FUSARO, P., VISAGGIO, G., TORTORELLA, M., 1998, "REP - ChaRacterizing and Exploiting Process Components: Results of Experimentation". In: Working Conference on Reverse Engineering, pp. 20-29, Honolulu, United States, October.

GARY, K., DERNIAME, J.C., LINDQUIST, T., AND KOEHNEMANN, H., 1998, "Component-Based Software Process Support". In: *Proceedings of the 13th Conference on Automated Software Engineering (ASE'98)*, Honolulu, Hawaii, October.

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

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

GINSBERG, M., QUINN, L., 1995, Process Tailoring and the Software Capability Maturity ModelCMU/SEI-94-TR-024,SEI-CMU. Disponível em: http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.024.html.

GRECO, G., GUZZO, A., PONTIERI, L., 2008, "Mining Taxonomies of Process Models". Data Knowl. Eng. 67(1), pp. 74–102.

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

HENNINGER, S., 1998, "An environment for reusing software processes". In: *Proceedings of Fifth International Conference on Software Reuse,* pp. 103-112.

HOLLENBACH, C., FRAKES, W., 1996, "Software Process Reuse in an Industrial Setting". In: 4th International Conference on Software Reuse, pp. 22-30, Orlando, United States, April.

HUMPHREY, W. S., 1989, “Managing the Software Process”. Boston, MA, USA, Addison-Wesley.

IIDA H., TANAKA Y., 2002, “A Compositional Process Pattern Framework for Component-based Process Modeling Assistance”, In [SDPP02].

ISO/IEC-12207, 2008, "Systems and Software Engineering - Software Life Cycle Process", The International Organization for Standardization and the International Electrotechnical Commission, v. ISO/IEC 12207, Genebra, Suíça.

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

JØRGENSEN, H.; CARLSEN, S., 2001, “Writings in Process Knowledge Management: Management of Knowledge Captured by Process Models”. In: Oslo: SINTEF Telecom and Informatics. Technical Report.

JUNIOR, E. A. O., PAZIN, M. G., GIMENES, I. M., KULESZA, U., & ALEIXO, F. A., 2013, “SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines”. In Product-Focused Software Process Improvement, pp. 169-183, Springer Berlin Heidelberg.

KANG, K.C., COHEN, S.G., HESS, J.A., *et al*., 1990, “Feature-Oriented Domain Analysis (FODA) – Feasibility Study”, Technical Report CMU/SEI-90-TR-21, Software Engineering Institute (SEI).

KANG, K.C., LEE, J., DONOHOE, P., 2002, “Feature-Oriented Product Line Engineering”, IEEE Software, v.9, n.4 (Jul/August 2002), pp 58-65.

KELLNER, M.I., 1996, “Connecting reusable software process elements and components”. In: Proceedings of 10th International Software Process Workshop, pp. 8-11, Dijon, France, June.

KÜMMEL, G., 2007, Uma Abordagem para a Criação de Arquiteturas de Referência de Domínio a partir da Comparação de Modelos Arquiteturais de Aplicações, Dissertação de M.Sc., COPPE,UFRJ, Rio de Janeiro, Brasil.

LANNA, A. L. P. M, 2009, “Reuso de Processos de Software Baseado na Componentização de Processos e Conhecimento”. Dissertação de mestrado em Engenharia Elétrica, PUC-MG, Belo Horizonte, Brasil.

LANNA, A. L. P. M., PIETROBON, C. A. M., 2010, “Desenvolvimento de um Método e uma Ferramenta para a Reutilização de Processos de Desenvolvimento de Software”. *Qualidade e Produtividade em Software Projetos Ciclo*.

LINDVALL, M., RUS., I., 2000, “Process Diversity in Software Development”, In: IEEE Software, n. 17, v.4, pp. 14-18.

LONCHAMP, J., 1993, “A Structured Conceptual and Terminological Framework for Software Process Engineering”. In: Proceedings of the Second International Conference on the Software Process, Vol. 2 (1993), pp. 41-53, Berlin, Germany.

MACHADO, L. F. D. C., 2000, Modelo para Definição de Processos de Software na Estação TABA. Dissertação de M. Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MAGDALENO, A. M., 2010, “Apoio à Decisão para o Balanceamento de Colaboração e Disciplina nos Processos de Desenvolvimento de Software”. Exame de Qualificação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MAGDALENO, A. M., DE ARAUJO, R. M., & WERNER, C., 2012, “COMPOOTIM: An Approach to Software Processes Composition and Optimization”. In *CIbSE*, pp. 42-55.

MAGDALENO, A. M., 2013, “COMPOOTIM: Em Direção ao Planejamento, Acompanhamento e Otimização da Colaboração na Definição de Processos de Software”. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MARTÍNEZ-RUIZ, T., GARCÍA, F., PIATTINI, M., 2008, “Towards a SPEM v2. 0 extension to define process lines variability mechanisms”. In: *Software engineering research, management and applications*, pp. 115-130. Springer Berlin Heidelberg.

MARTÍNEZ-RUIZ, T., GARCÍA, F., PIATTINI, M., 2011, “Managing process diversity by applying rationale management in variant rich processes”. In: *Product-Focused Software Process Improvement,* pp. 128-142. Springer Berlin Heidelberg.

MARTÍNEZ-RUIZ, T., MÜNCH, J., GARCÍA, F., & PIATTINI, M., 2012, “Requirements and constructors for tailoring software processes: a systematic literature review”. In: Software Quality Journal, 20(1), 229-260.

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

MONTERO, PENA, RUIZ-CORTÉS, 2007, "Business Family Engineering: Does it make sense?", In: *II Congreso Español de Informática* (Cedi 2007), n. *Actas de los Talleres de las Jornadas de Ingeniería del Softwarey Bases de Datos* (Tjisbd), pp. 34-40.

MURTA, L., 2002, CHARON: Uma Máquina de Processos Extensível baseada em Agentes Inteligentes. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

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

NORTHROP, L., 2002, “SEI’s Software Product Line Tenets”, IEEE Software, v.19, n. 4, pp. 32-40, July/August.

NUNES, V. T.; WERNER, C.; SANTORO, F. M., 2010, "Context-Based Process Line". In: International Conference on Enterprise Information Systems (ICEIS), pp. 277-282, Funchal, Madeira, Portugal.

NUNES, V. T.; WERNER, C. M. L.; SANTORO, F. M., 2012, "Mediating process adaptation through a goal-oriented context-aware approach". In: 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), p. 160 – 167.

OCAMPO, A., BELLA, F., & MÜNCH, J., 2005, “Software Process Commonality Analysis”. *Software Process: Improvement and Practice*, *10*(3), 273-285.

ODYSSEY, 2014, "Projeto Odyssey", In: http://reuse.cos.ufrj.br/odyssey.

OLIVEIRA, R.F., 2006, Formalização e Verificação de Consistência na Representação de Variabilidades. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

OLIVEIRA JUNIOR, E. A.; GIMENES, I. M. S.; MALDONADO, J. C., 2010, "Systematic Management of Variability in UML-based Software Product Lines". In: *Journal of Universal Computer Science (Print)*, v. 16, 2010.p. 2374-2393.

OMG, 2008, "Software Process Engineering Metamodel". Disponível em: http://www.omg.org/technology/documents/formal/spem.htm.

OSTERWEIL, L., 1987, "Software processes are software too". In: Proceedings of the 9th international conference on Software Engineering. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press, p. 2–13.

PAZIN, M.G., 2012, "SMartySPEM: Uma Extensão da Abordagem SPEM para Gerenciamento de Variabilidades em Linhas de Processo de Software", Trabalho de Conclusão de Curso, Graduação em Bacharel em Informática, Universidade Estadual de Maringá.

PEDREIRA, O., PIATTINI, M., LUACES, M. R., *et al.*, 2007, "A systematic review of software process tailoring", SIGSOFT Software Engineering Notes, v. 32, n. 3, pp. 1-6.

PILLAT, R. M., OLIVEIRA, T. C., FONSECA, F. L., 2012, "Introducing Software Process Tailoring to BPMN: BPMNt". In: *International Conference on Software and System Process (ICSSP),* pp. 58-62. IEEE.

PORTELA, C., VASCONCELOS, A., SILVA, A., SINIMBÚ, A., SILVA, E., RONNY, M., OLIVEIRA, S., 2012, "A Comparative Analysis between BPMN and SPEM Modeling Standards in the Software Processes Context". In: *Journal of Software Engineering & Applications*, *5*(5).

PUHLMANN, F., SCHNIEDERS, A., WEILAND, J., WESKE, M., 2005, "Variability mechanisms for process models". *PESOA-Report TR*, *17*, 10-61.

RAZAVIAN, M., KHOSRAVI, R., 2008, "Modeling Variability in Business Process Models Using UML". In: *Fifth International Conference on Information Technology: New Generations* (ITNG'08), pp. 82 –87.

REGEV, G., SOFFER, P., SCHMIDT, R., 2006, "Taxonomy of flexibility in business processes". In: *Proceedings of the 7th Workshop on Business Process Modelling, Development and Support (BPMDS'06)*, vol. 236.

REIS, R. Q., REIS, C. A. L., NUNES, D. J., 2001, "Automated support for software process reuse: Requirements and early experiences with the APSEE model". In. *Proceedings of Seventh International Workshop on Groupware*, pp. 50-55.

REIS, R. Q., 2002, APSEE-Reuse: um meta-modelo para apoiar a reutilização de processos de software. Tese de D. Sc., Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brasil.

REIS, C.A., 2003, "Uma Abordagem Flexível para Execução de Processos de Software Evolutivos". Tese de Doutorado, Porto Alegre: PPGC- UFRGS.

ROLLAND, C., NURCAN, S., 2010, “Business Process Lines to Deal with the Variability”. In: *2010 43rd Hawaii International Conference on System Sciences (HICSS)*, p. 1 –10.

ROMBACH, D., 2006, "Integrated Software Process and Product Lines". In: Unifying the Software Process Spectrum, Berlin/Heidelberg: Springer-Verlag, pp. 83-90.

ROMBACH, D., 2013, “Integrated Software Process and Product Lines”. In *Perspectives on the Future of Software Engineering*, pp. 359-366. Springer Berlin Heidelberg.

ROUILLÉ, E., COMBEMALE, B., BARAIS, O., TOUZET, D., JÉZÉQUEL, J. M., 2013, “Improving Reusability in Software Process Lines”. In: Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on (pp. 90-93).

RUIZ-RUBE, I., DODERO, J. M., PALOMO-DUARTE, M., RUIZ, M., GAWN, D., 2012, “Uses and applications of SPEM process models. A systematic mapping study”. *Journal of Software Maintenance and Evolution: Research and Practice*, *1*(32), 999-1025.

RU-ZHI *et al*., 2003*,* “*Research on matching algorithm for XML-based software component query*”, Journal of Software, July, pp.1195-1202.

RU-ZHI, X., TAO, H., DONG-SHENG, C*., et al.*, 2005, "Reuse-Oriented Process Component Representation and Retrieval". In: *International Conference on Computer and Information Technology*, pp. 911-315, Shangai, China, September.

RU-ZHI, X., PEIGUANG L., ZHIKUN Z., LEQIU Q., 2010, “An Approach of Reuse-based Software Process Improvement”. In: *Journal of Computational Information Systems Volume 6 Number 6*, pp. 1897-1906. Available at http://www.Jofcis.com

SAMETINGER, J., 1997, Software Engineering with Reusable Components, Springer-Verlag New York, Inc.

SCACCHI, W., MI, P., “Process Life Cycle Engineering: Approach and Support Environment”, Intern. J. Intelligent Systems in. Accounting, Finance, and Management, 6:83-107, 1997.

SCHNIEDERS, A., PUHLMANN, F., 2005, “Activity Diagram Inheritance1”.

SCHNIEDERS, A., PUHLMANN, F., 2006, “Variability Mechanisms in E-Business Process Families”. *BIS*, *85*, 583-601.

SCHWABER, K., 2004, Agile Project Management with Scrum. Washington, DC, USA, Microsoft Press.

SEGRINI, B.M. , 2009, *Definição de Processos Baseada em Componentes*. Dissertação de M.Sc., Universidade Federal do Espírito Santo, Vitória, ES, Brasil.

SHULL, F., CARVER, J., TRAVASSOS, G.H., 2001, "An empirical methodology for introducing software processes", *ACM*, pp. 288-296, Vienna, Austria.

SIMIDCHIEVA, B.I., CLARKE, L.A., OSTERWEIL, L., 2007, "Representing Process Variation with a Process Family". In: International Conference on Software Process, v. Lecture Notes in Computer Science 4470, pp. 109-120, Minneapolis, Estados Unidos, Maio.

SIMMONDS, J. e BASTARRICA, M.C., 2011, "Modeling Variability in Software Process Lines". Santiago, Chile, Universidad de Chile.

SIMMONDS, J., BASTARRICA, M.C., SILVESTRE, L. E QUISPE, A., 2011, "*Analyzing Methodologies and Tools for Specifying Variability in Software Processes*". Santiago, Chile, Universidad de Chile.

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

SOFTEX, 2012, Melhoria de Processo do Software Brasileiro – Guia Geral de Software, Modelo de Qualidade Disponível em: http://www.softex.br.

STERMAN, J.D., 1992, "System Dynamics Modeling for Project Management", Relatório Técnico, MIT System Dynamics Group, Cambridge, M.

STOROLLI, A., GUIDINI, J., ZANOLLA, G., TEREZINHA, B., 2009, "Modelagem de Processos de Software". In: Octava Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2009), 10 - 13 de Julho, Orlando, Florida, EUA.

SUTTON, S., OSTERWEIL, L., 1996, "Product families and process families". In: *Proceedings of the 10th International Software Process Workshop* (ISPW), pp. 109-111, Dijon, France.

TEIXEIRA, E.N., 2011, "ODYSSEYPROCESS-FEX: uma abordagem para modelagem de variabilidades de linha de processos de software", Dissertação de M.Sc., Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro.

TERNITÉ, T., 2009, "Process Lines: A Product Line Approach Designed for Process Model Development". In: *Software Engineering and Advanced Applications. SEAA '09. 35th Euromicro Conference on*, pp. 173-180, 27-29 Aug.

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

VAN EIJNDHOVEN, T., IACOB, M.-E., PONISIO, M.L., 2008, "Achieving Business Process Flexibility with Business Rules". In: *12th International IEEE Enterprise Distributed Object Computing Conference (EDOC '08)*, p. 95 –104.

VASCONCELOS, A., 2007, " Uma Abordagem de apoio à Criação de Arquiteturas de Referência de Domínio baseadas na Análise de Sistemas Legados". Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

VASCONCELOS, A., WERNER, C., 2007, "Architecture recovery and evaluation aiming at program understanding and reuse". In: *Software Architectures, Components, and Applications*, pp. 72-89. Springer Berlin Heidelberg.

VERVUURT, Mark, 2007, "Modeling Business Process Variability", University of Twente.

WASHIZAKI, H., 2006a, "Building software process line architectures from bottom up". In: Proceedings of the 7th International Conference on Product-Focused Software Process Improvement, PROFES 2006, pp. 415-421, Amsterdam, Netherlands, June.

WASHIZAKI, H., 2006b, "Deriving project-specific processes from process line architecture with commonality and variability". In: 2006 IEEE International Conference on Industrial Informatics, INDIN'06, August 16, 2006 - August 18, 2006, pp. 1301-1306, Singapore, Singapore.

WERNER, C., MATTOSO, M., BRAGA, R., et al., 1999, "Odyssey: Infra-estrutura de reutilização Baseado em Modelos de Domínios". In: Caderno de Ferramentas do XIII Simpósio Brasileiro de Engenharia de Software, pp. 17-20, Florianópolis, outubro.

WERNER, C.M.L., 2005, Projeto Reuse: Técnicas e Ferramentas de apoio à Reutilização de Software, Projeto Integrado CNPq.

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

ZAMLI, K. Z., LEE, P.A., 2001, "Taxonomy of Process Modelling Languages". In: Proc. of the ACS/IEEE Inter. Conf. on Computer Systems and Applications (AICCSA'01) Beirut, Lebanon, June.

# ANEXO I

## *ODYSSEYPROCESS-FEX*: PACOTE PRINCIPAL

O pacote Principal (*Core*) do meta-modelo *OdysseyProcess-FEX* (Figura 28) é constituído por características, suas diversas categorias e atributos específicos. No total, é composto por quatro categorias de características: *Característica Atividade (ActivityFeature), Característica Tarefa (TaskFeature), Característica Papel (RoleFeature) e Característica Produto de Trabalho (workProductFeature)*. Um elemento de nota (classe *Comment*) também é apresentado neste pacote como forma de agregar informações de apoio para entendimento do domínio. A classe Disciplina (*Discipline*) representa uma forma de organização do conteúdo de unidades de trabalho em pacotes.

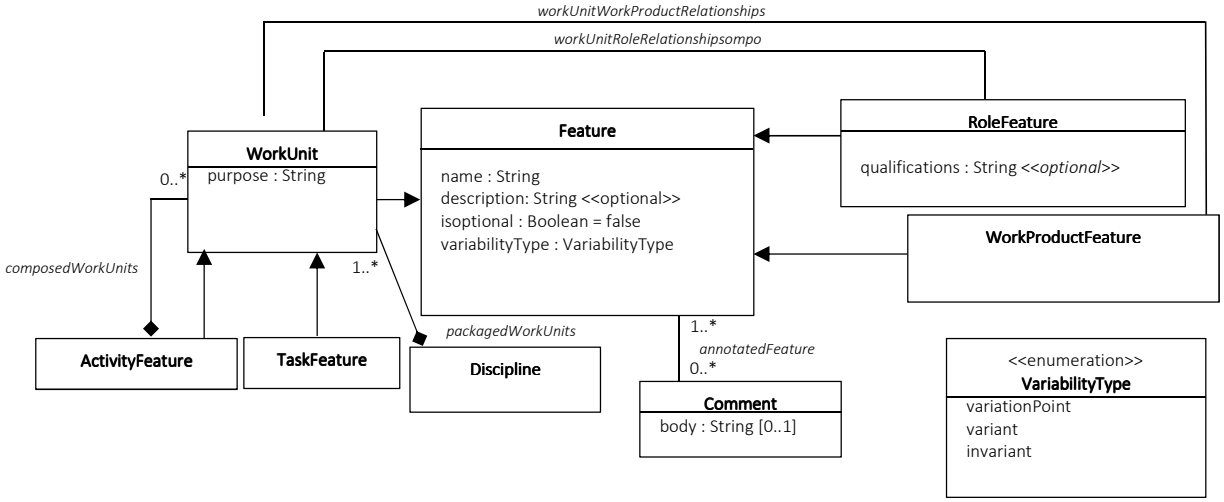



**Figura 28 - Meta-modelo *OdysseyProcess-FEX*: Pacote Principal**

A seguir uma descrição detalhada de cada elemento deste pacote é apresentada, seguindo uma estrutura divida em cinco itens de descrição: Descrição, Hierarquia, Atributos, Associações e Restrições.

### a. *Feature* (Característica)

#### Descrição

Representa elementos estruturais do domínio de processos de software.

#### Hierarquia

Subclasses: *WorkUnit (UnidadeDeTrabalho), RoleFeature (Característica Papel) e WorkProductFeature (Característica Produto de Trabalho).*

#### Atributos

- ***name*** **(nome)**: String[1] – atributo que define o nome da característica.
- ***description*** **(descrição)**: String[0..1] – atributo que permite uma descrição textual sobre a característica.
- ***isOptional*** **(ehOpcional)**: Boolean – atributo que define a classificação da característica quanto a sua opcionalidade. Indica se a característica é mandatória no domínio, isto é, se a característica estará presente em todos os processos instanciados a partir do modelo ou se a característica é opcional. Default: *false*.

- ***variabilityType*** **(tipoVariabilidade)**: *VariabilityType*[1] (TipoVariabilidade[1]) – atributo que indica o tipo de variabilidade que a característica apresenta. Pode assumir os valores *invariant* (invariante), variant (variante) e *variation point* (ponto de variação). O tipo de variabilidade é representado pela lista enumerada *VariabilityType (TipoVariabilidade)*. Default: *invariant* (invariante).

**Associações**

Sem associações específicas.

**Restrições**

[1] As classificações de características com relação à opcionalidade são mutuamente excludentes entre si. Desta forma, uma característica não pode receber simultaneamente dois tipos diferentes de classificação quanto à opcionalidade.

[2] As classificações de características com relação à variabilidade são mutuamente excludentes entre si. Desta forma, uma característica não pode receber simultaneamente dois tipos diferentes de classificação quanto à variabilidade.

[3] As classificações quanto à opcionalidade e variabilidade são ortogonais entre si (Figura 29). Desta forma, uma característica pode receber uma classificação quanto à opcionalidade e outra classificação complementar quanto à variabilidade.

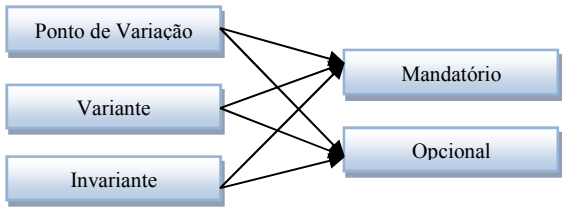



Figura 29 - Classificação ortogonal das características

b. VariabilityType (TipoVariabilidade)

**Descrição**

*VariabilityType (TipoVariabilidade*) é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo de variabilidade presente em uma Característica. Pode assumir os seguintes valores: *variation point* (ponto de variação), *variant* (variante) e *invariant* (invariante), onde (OLIVEIRA, 2006):

1. *Pontos de variação*: são características que refletem a parametrização no domínio de uma maneira abstrata. São configuráveis através das variantes.
2. *Variante*s: são elementos NECESSARIAMENTE ligados a um ponto de variação, que atuam como alternativas para configurar determinado ponto de variação.
3. *Invariantes*: são elementos "fixos", que não são configuráveis no domínio.

**Hierarquia**

Sem hierarquia específica.

**Atributos**

Sem atributos.

**Associações**

Sem associações específicas.

**Restrições**

Sem restrições específicas.

c. *WorkUnit* (Unidade de Trabalho)

Descrição

Elemento que representa o conceito de uma unidade de trabalho a ser executada para alcançar determinado valor de resultado dentro de um processo. Representa uma unidade de ação.

Hierarquia

Superclasse: *Feature* (*Característica)*

Subclasses: *ActivityFeature* (CaracterísticaAtividade) e *TaskFeature* (CaracterísticaTarefa)

Atributos

- ***purpose*** **(propósito)**: String [0..1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela unidade de trabalho a ser executada para alcançar determinado valor de resultado dentro do processo.

Associações

- ***workUnitRoleRelation*** (relação entre papel e unidade de trabalho) : *WorkUnitRoleRelationship* [1..*] (LigaçãoPapelUnidadeDeTrabalho [1..*]) – especifica relações de responsabilidades entre unidades de trabalho e papéis.
- ***workUnitWorkProductRelation*** (relação entre produto de trabalho e unidade de trabalho) : *WorkUnitWorkProductRelationship*[1..*] (LigaçãoProdutoDeTrabalhoUnidadeDeTrabalho [1..*]) – especifica relações entre unidades de trabalho e produtos de trabalho.

As classes *WorkUnitRoleRelationship* e *WorkUnitWorkProductRelationship* pertencem ao pacote Relacionamentos.

Restrições

Sem restrições específicas.

d. *ActivityFeature* (CaracterísticaAtividade)

Descrição

Característica que representa o agrupamento de unidades de trabalhos menores, representadas por outras atividades ou por unidades elementares, especificadas por tarefas.

Hierarquia

Superclasse: *WorkUnit (UnidadeDeTrabalho)*

Atributos

- ***purpose*** **(propósito)**: String [0..1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela unidade de trabalho a ser executada para alcançar determinado valor de resultado dentro do processo.

Associações

- ***composedWorkUnits*** (unidades de trabalhos compostas) : *WorkUnit* [1..*] (UnidadeDeTrabalho [1..*]) – especifica relações de composição da atividades por outras unidades de trabalho.

Restrições

[1] Uma *ActivityFeature* (*CaracterísticaAtividade*) representa o agrupamento de trabalho definido por outras unidades de trabalho (*WorkUnit*) que são descritas por características da categoria atividade (*AcitivityFeature*) ou por unidades elementares representadas por características da categoria tarefa (*TaskFeature*).

e. *TaskFeature* (CaracterísticaTarefa)

Descrição

Característica que representa uma unidade fundamental de trabalho. Sua granularidade é definida como uma unidade elementar.

Hierarquia

Superclasse: *WorkUnit (UnidadeDeTrabalho)*

Atributos

- ***purpose*** **(propósito)**: String [0..1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela unidade de trabalho a ser executada para alcançar determinado valor de resultado dentro do processo.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

f. *RoleFeature* (CaracterísticaPapel)

Descrição

Característica que representa um conjunto de habilidades, competências e responsabilidades de um indivíduo ou um conjunto de indivíduos. Não especifica um indivíduo ou recurso em particular.

Hierarquia

Superclasse: *Feature* (*Característica*)

Atributos

- ***qualifications*** **(qualificações)**: String [0..1] - atributo que descreve o conjunto de competências e habilidades disponibilizadas pelo papel representado.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

g. *WorkProductFeature* (CaracterísticaProdutoDeTrabalho)

Descrição

Característica que representa um artefato consumido, modificado ou produzido por uma unidade de trabalho.

Hierarquia

Superclasse: *Feature* (*Característica*)

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

h. *Discipline* (Disciplina)

Descrição

Representa o conceito de pacote para organização do domínio em escopos menores ou subdomínios. Elemento que representa uma categorização de trabalho baseado em uma similaridade de interesses e propósito de resultados. A disciplina é um conjunto de

unidades de trabalho que estão relacionadas com uma grande área de interesse no âmbito do domínio como um todo.

**Hierarquia**

Sem hierarquia específica.

**Atributos**

Sem atributos.

**Associações**

- ***packagedWorkUnits***(UnidadesDeTrabalhoEmpacotadas): *WorkUnit*[*](UnidadeDeTrabalho [*]) – especifica as unidades de trabalho que foram agrupadas e pertencem a uma disciplina.

**Restrições**

Sem restrições específicas.

i. *Comment* (Nota)

**Descrição**

Um comentário é uma anotação textual associada a um elemento com o objetivo de acrescentar alguma informação adicional para organização de conhecimento sobre o domínio. Um conjunto de informações de processos de software não constitui um elemento de processo em si, mas seu conteúdo auxilia a entender partes do processo ou o processo como um todo. Um exemplo de informação adicional poderia ser a definição de uma *prática* (forma ou estratégia comprovada de realizar um trabalho para atingir um objetivo que tem um impacto positivo sobre um produto de trabalho ou sobre a qualidade do processo. As práticas podem resumir os aspectos que impactam muitas partes diferentes de um processo) ou a descrição de um *conceito* (descreve uma ideia-chave, aborda temas gerais ou princípios básicos que permeiam os diferentes elementos que constituem um processo).

**Hierarquia**

Sem hierarquia específica.

**Atributos**

- ***body*** (corpo do comentário) : String [0..1] – especifica o conteúdo da nota associada a uma característica.

**Associações**

- ***annotatedFeature*** (característicaComentada) : *Feature*[*] (Característica[*]) – referência as características que possuem comentários associados.

**Restrições**

Sem restrições específicas.

O pacote Relacionamentos (*Relationships*) consiste na representação das relações disponibilizadas entre as características. Está estrutura na classe abstrata *Relacionamento* e suas especializações: *AlternativeRelationship (Alternativo), Aggregation (Agregação), Composition (Composição), WorkUnitRoleRelationship (LigaçãoPapelUnidadeDeTrabalho), WorkUnitWorkProductRelationship (LigaçãoProdutoDeTrabalhoUnidadeDeTrabalho) e WorkProductRoleRelationship (LigaçãoPapelProdutoDeTrabalho)* (Figura 1).

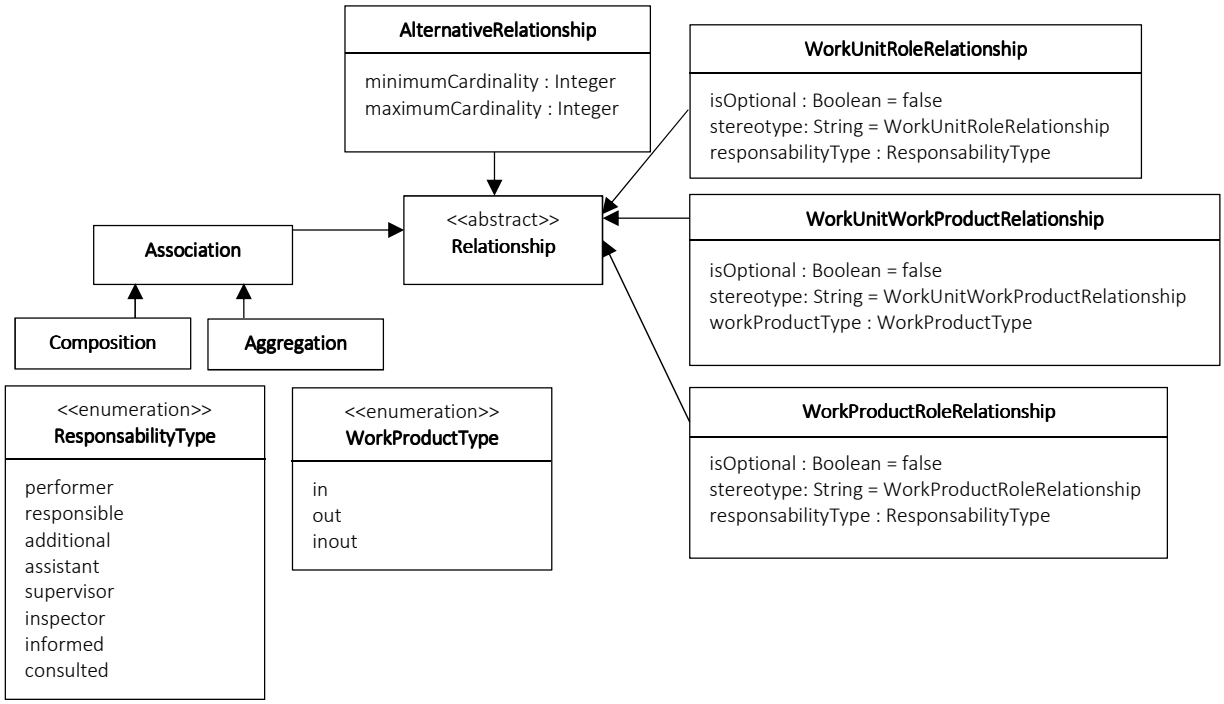



**Figura 30 - Meta-modelo OdysseyProcess-FEX: Pacote Relacionamentos (*Relationship*)**

A seguir cada elemento desse pacote será descrito de forma detalhada, seguindo uma estrutura de cinco itens de descrição: Descrição, Hierarquia, Atributos, Associações e Restrições.

a. ***Relationship*** **(Relacionamentos)**

**Descrição**

*Relationship* (Relacionamento) é um conceito abstrato que especifica algum tipo de relacionamento entre elementos (OMG, 2005). Classe Abstrata.

**Hierarquia**

SubClasses*: AlternativeRelationship (Alternativo), Association (Associação), Aggregation (Agregação), Composition (Composição), WorkUnitRoleRelationship (LigaçãoPapelTarefa), WorkUnitWorkProductRelationship (LigaçãoProdutoDeTrabalhoTarefa) e WorkProductRoleRelationship (LigaçãoPapelProdutoDeTrabalho).*

**Atributos**

Sem atributos.

**Associações**

Sem associações específicas.

**Restrições**

Sem restrições específicas.

b. *AlternativeRelationship* (Alternativo)

Descrição

Relacionamento existente entre um ponto de variação e suas variantes. Denota a pertinência de uma variante a um determinado ponto de variação (OLIVEIRA *et al.,* 2005). A Figura 31 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de características que podem participar desse tipo de relacionamento. As restrições quanto à combinação das categorias de características são apresentadas no item [10] do campo Restrições deste relacionamento.

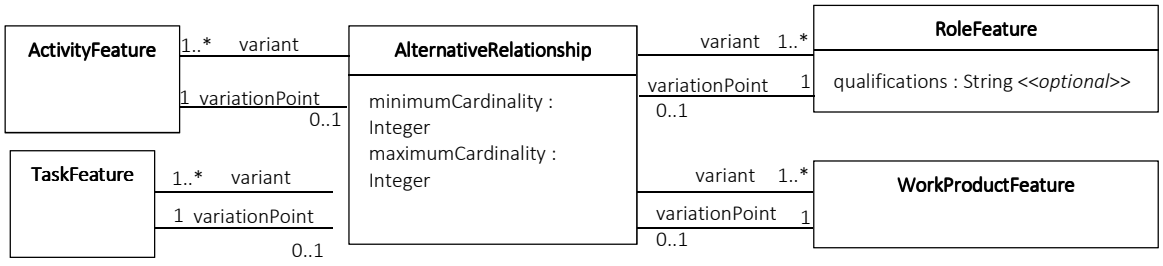



Figura 31 - Meta-modelo *OdysseyProcess-FEX*: *Alternative Relationship* (Alternativo)

Hierarquia

*Superclasse*: *Relationship* (Relacionamento).

Atributos

*Cardinalidade*: atributo que define o número de instâncias mínimas de variantes que poderão ocupar o ponto de variação. As cardinalidades são representadas através da definição de um intervalo, em que o valor mínimo e máximo são determinados, ou seja, podem ser representadas através de intervalos fixos. Desta forma, este atributo é definido pela combinação dos seguintes atributos:

- **minimumCardinality (cardinalidadeMínima)**: Integer[1] - atributo que define o número mínimo de instâncias de variantes que poderão ocupar o ponto de variação.
- **maximumCardinality (cardinalidadeMáxima)**: Integer[1] - atributo que define o número máximo de instâncias de variantes que poderão ocupar o ponto de variação.

Associações

- **variant (variante)**: Feature[1..*] – Referencia as características do tipo Variante que fazem parte do relacionamento Alternativo.
- **variationPoint (pontoVariação)**: Feature[1] - Referencia a característica do tipo Ponto de Variação que faz parte do relacionamento Alternativo.

Restrições

[1] O relacionamento Alternativo só poderá ocorrer entre Características do tipo Ponto de Variação: *Feature.variabilityType = variationPoint (Característica.tipoVariabilidade = pontoVariação*) e do tipo Variante: *Feature.variabilityType = variant* (*Característica.tipoVariabilidade=variante*).

[2] Características classificadas como Ponto de Variação: *Feature.variabilityType = variationPoint* (*Característica.tipoVariabilidade = pontoVariação*) são caracterizadas como origem do relacionamento do tipo *Alternativo*.

[3] Características classificadas como Variante: *Feature.variabilityType = variant* (*Característica.tipoVariabilidade = variante*) são caracterizadas como destino do relacionamento do tipo *Alternativo*.

[4] Características classificadas como Variante: *Feature.variabilityType = variant* (*Característica.tipoVariabilidade = variante*) e Mandatória: *Feature.isOptional = false* (*Característica.ehOpcional = false*) devem ter um relacionamento do tipo *Alternativo* com outra característica classificada como Ponto de Variação: *Feature.variabilityType = variationPoint* (*Característica.tipoVariabilidade = pontoVariação*) NECESSARIAMENTE Mandatória: *Feature.isOptional = false* (*Característica. ehOpcional = false*).

[5] Características classificadas como Variante: *Feature.variabilityType = variant* (*Característica.tipoVariabilidade = variante*) e Opcionais: *Feature.isOptional = true* (*Característica.ehOpcional = true*) podem ter um relacionamento do tipo Alternativo com outra característica classificada como Ponto de Variação: *Feature.variabilityType = variationPoint* (*Característica.tipoVariabilidade = pontoVariação*) que seja Mandatória: *Feature.isOptional = false* (*Característica. ehOpcional = false*). Neste caso, a obrigatoriedade do ponto de variação indica que pelo menos uma das variantes ligadas a ele deve ser selecionada na aplicação.

[6] Características classificadas como Ponto de Variação: *Feature.variabilityType = variationPoint* (*Característica.tipoVariabilidade = pontoVariaçã*o) e Opcionais: *Feature.isOptional = true* (*Característica.ehOpcional = true*) devem ter um relacionamento do tipo *Alternativo* com outras características classificadas como Variantes: *Feature.variabilityType = variant* (*Característica.tipoVariabilidade = variante*) NECESSARIAMENTE Opcionais: *Feature.isOptional = true* (*Característica.ehOpcional = true*).

[7] Relacionamentos Alternativos com cardinalidade máxima com valor igual a um (*maximumCardinality* = 1), representam relacionamentos de mútua exclusividade entre as variantes do ponto de variação associado.

[8] Relacionamentos Alternativos com cardinalidade mínima com valor igual a zero (*minimumCardinality* = 0), representam relacionamentos em que as características que representam o ponto de variação são classificadas como opcionais.

[9] Relacionamentos Alternativos com cardinalidade mínima com valor superior a zero (*minimumCardinality* > 0), representam relacionamentos em que as características que representam o ponto de variação são classificadas como mandatórias.

[10] O relacionamento Alternativo só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 22).

**Tabela 22- Relacionamento Alternativo: Restrições das combinações de categorias de Características**

| **Tipo de Relacionamento** | **Categoria de Característica** | **Categoria de Característica** |
|---|---|---|
| | *Origem:* ***Ponto de Variação*** | *Destino:* ***Variante*** |
| | CaracterísticaAtividade | CaracterísticaAtividade |
| | CaracterísticaTarefa | CaracterísticaTarefa |
| | CaracterísticaPapel | CaracterísticaPapel |
| | CaracterísticaProdutoDeTrabalho | CaracterísticaProdutoDeTrabalho |

c. *Association* (Associação)

Descrição

Uma associação descreve um conjunto de tuplas cujos valores se referem a instâncias tipadas (OLIVEIRA *et al.*, 2005). Uma instância de uma associação é chamada ligação. Essa ligação representa a relação, uni ou bidirecional, entre duas características. O relacionamento pode possuir um estereótipo que especifica o tipo da ligação.

Hierarquia

Superclasse: *Relationship* (Relacionamento).

Atributos

Sem atributos específicos.

Associações

Sem associações específicas.

Restrições

[1] Esse relacionamento não será aplicado entre características, apenas suas especializações: *Composition* (Composição) e *Aggregation* (Agregação).

d. Agregação

Descrição

Uma associação pode representar uma agregação (isto é, um relacionamento de todo/parte) (OLIVEIRA *et al.*, 2005). A Figura 32 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de características que podem participar desse tipo de relacionamento. As restrições quanto à combinação das categorias de características são apresentadas no item [2] do campo Restrições.

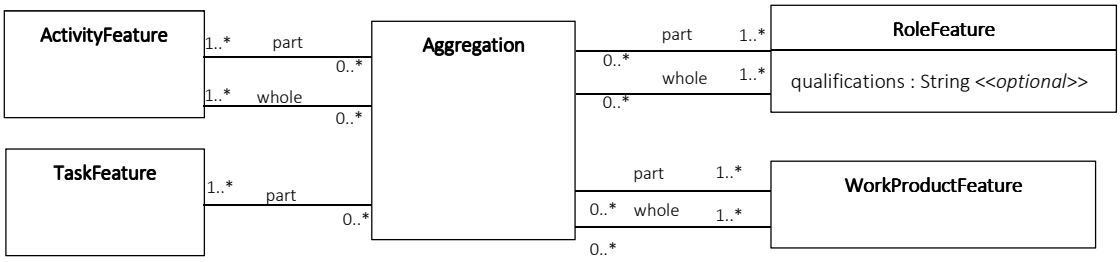



Figura 32 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento *Aggregation* (Agregação)

Hierarquia

Superclasse: *Association* (*Associação)*

Atributos

Sem atributos.

Associações

- ***whole*** **(extremidadeTodo)**: Referencia a característica que representa o todo no relacionamento *Aggregation* (Agregação).
- ***part*** **(extremidadeParte)**: Referencia a característica que representa parte no relacionamento *Aggregation* (Agregação).

Restrições

[1] Somente associações binárias podem ser um relacionamento *Aggregation* (Agregação).

[2] O relacionamento *Aggregation* (Agregação) só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 23):

Tabela 23- Relacionamento *Aggregation* (Agregação): Restrições das combinações de categorias de Características

| Tipo de Relacionamento *Aggregation (Agregação)* | Categoria de Característica | Categoria de Característica |
|---|---|---|
| | *Origem*: ExtremidadeTodo | *Destino*: ExtremidadeParte |
| | Atividade | Atividade |
| | Atividade | Tarefa |
| | Papel | Papel |
| | ProdutoDeTrabalho | ProdutoDeTrabalho |

e. Composição

Descrição

Uma associação pode representar uma composição (isto é, um relacionamento de todo/parte). A composição é um relacionamento mais forte do que agregação, e requer que em um dado momento, uma instância esteja incluída em no máximo uma composição (OLIVEIRA *et al*., 2005). Neste relacionamento, as partes não existem independentes do todo. A Figura 33 apresenta o relacionamento e as categorias de características envolvidas.

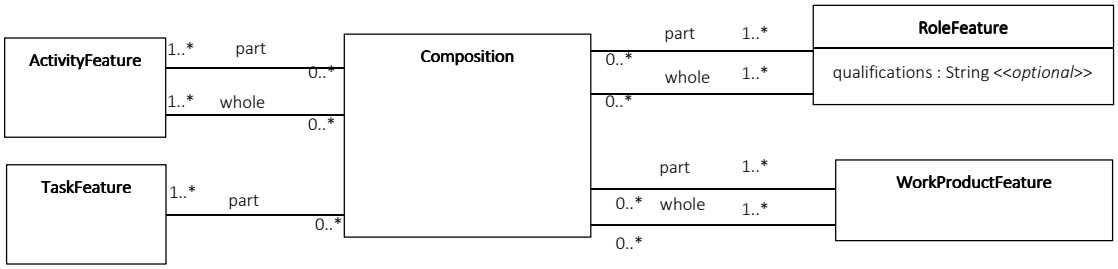



Figura 33 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento *Composition* (Composição)

Hierarquia

Superclasse: *Association* (Associação)

Atributos

Sem atributos.

Associações

- ***whole*** **(extremidadeTodo)**: Referencia a característica que representa o todo no relacionamento *Composition* (Composição).
- ***part*** **(extremidadeParte)**: Referencia a característica que representa parte no relacionamento *Composition* (Composição).

Restrições

[1] Somente associações binárias podem ser um relacionamento *Composition* (Composição).

[2] Não deve haver relacionamento *Composition* (Composição) entre características quando a característica que representa o "todo" é opcional e a característica que representa a "parte" é mandatória.

[3] O relacionamento *Composition* (Composição) só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 24):

Tabela 24 - Relacionamento *Composition* (Composição): Restrições das combinações de categorias de Características

| Tipo de Relacionamento *Composition* (Composição) | Categoria de Característica | Categoria de Característica |
|---|---|---|
| | *Origem*: ExtremidadeTodo | *Destino*: ExtremidadeParte |
| | Atividade | Atividade |
| | Atividade | Tarefa |
| | Papel | Papel |
| | ProdutoDeTrabalho | ProdutoDeTrabalho |

f. *WorkUnitRoleRelationship* (LigaçãoPapelUnidadeDeTrabalho)

Descrição

Estabelece um relacionamento de associação entre uma característica da categoria Unidade de Trabalho (Atividade ou Tarefa) e uma ou várias características da categoria Papel participantes na sua execução. A Figura 34 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de características que podem participar desse tipo de relacionamento.

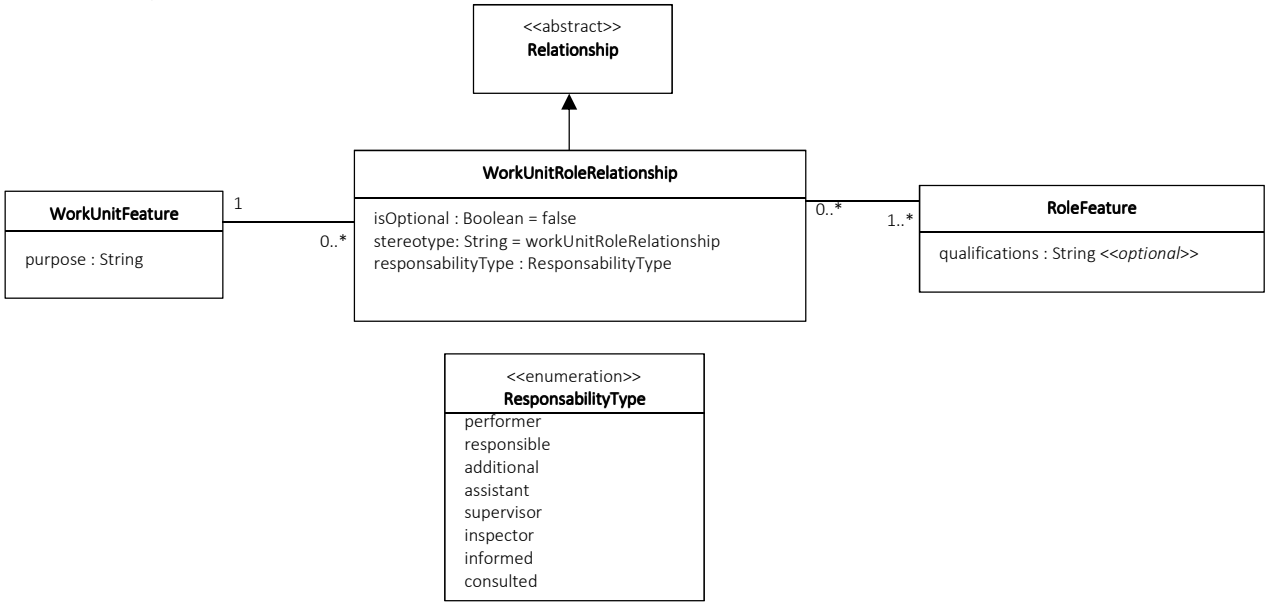



Figura 34 - Meta-modelo *OdysseyProcess-FEX*: *WorkUnitRoleRelationship* (LigaçãoPapelTarefa)

Hierarquia

Superclasse: *Relationship* (Relacionamento*)*

Atributos

- ***isOptional*** **(ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação da característica Papel (*RoleFeature*) na execução da unidade de trabalho (*WorkUnit*) representada pela característica atividade (*ActivityFeature*) ou tarefa (*TaskFeature*) associada. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype*** **(estereótipo):** String = *workUnitRoleRelationship* – atributo que especifica o tipo de associação criada.
- ***responsabilityType*** (tipoResponsabilidade): ResponsabilityType. A classe ResponsabilityType representa uma enumeração dos tipos de responsabilidades que podem ser assumidos por um papel participante da execução de uma unidade de trabalho. De forma inicial foi definido um conjunto de responsabilidades possíveis representadas pelos estereótipos: *<<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>* (Tabela 25).

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Tabela 25 - Tipos de Responsabilidades dos papéis envolvidos na execução de unidades de trabalho

| Estereótipo | Descrição: descreve o tipo de participação do papel envolvido na tarefa |
|---|---|
| *<<performer>>* | Executante da tarefa (indivíduo considerado o executante primário da tarefa) |
| *<<responsible>>* | Responsável pela tarefa |
| *<<additional>>* | Adicional (indivíduo considerado um executante secundário da tarefa) |
| *<<assistant>>* | Assistente (auxilia o executante na realização da tarefa) |
| *<<supervisor>>* | Supervisor (responsável pela infra-estrutura geral de realização da tarefa) |
| *<<inspector>>* | Inspetor (responsável pela verificação dos resultados gerados pela tarefa) |
| *<<informed>>* | Informado (indivíduo com interesses diretos que precisa ser informado da execução da tarefa) |
| *<<consulted>>* | Consultado (indivíduo com conhecimento relevante que precisa ser consultado para a execução da tarefa) |

### g. *WorkUnitWorkProdutRelationship* (LigaçãoProdutoDeTrabalhoUnidadeDeTrabalho)

#### Descrição

Estabelece um relacionamento de associação entre uma característica da categoria Unidade de Trabalho (Atividade ou Tarefa) e uma ou várias características da categoria ProdutoDeTrabalho que representam artefatos a serem consumidos, produzidos ou modificados pela execução da unidade de trabalho. A Figura 35 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de características que podem participar desse tipo de relacionamento.

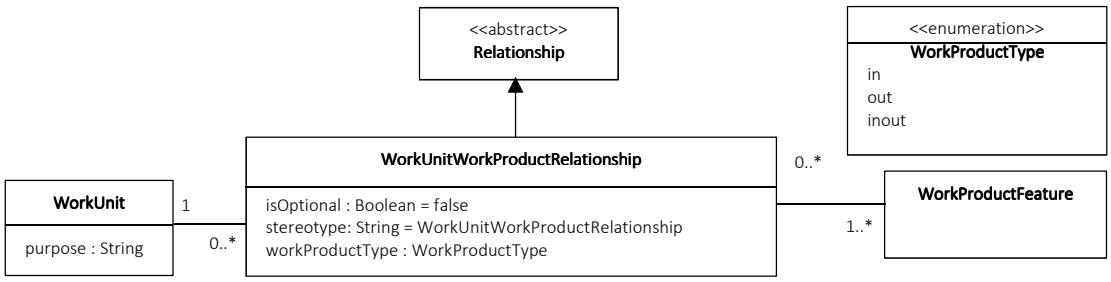



**Figura 35 - Meta-modelo *OdysseyProcess-FEX*: WorkUnitWorkProductRelationship (LigaçãoProdutoDeTrabalhoUnidadeDeTrabalho)**

#### Hierarquia

Superclasse: *Relacionamentos*

#### Atributos

- ***isOptional*** **(ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação da característica ProdutoDeTrabalho (*WorkProduct**Feature***) na execução da unidade de trabalho (*WorkUnit*) representada pela característica atividade (*ActivityFeature*) ou tarefa (*TaskFeature*) associada. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype*** **(estereótipo):** String = *workUnitWorkProductRelationship* – atributo que especifica o tipo de associação criada.
- ***workProductyType*** (tipoProdutoDeTrabalho): *WorkProductType*. A classe *WorkProductType* representa uma enumeração dos tipos que produtos de trabalho podem assumir, representados pelos estereótipos: *<<in>>, <<out>>, <<inout>>* (Tabela 26).

#### Associações

Sem associações específicas.

#### Restrições

Sem restrições específicas.

Tabela 26 - Tipos dos produtos de trabalho envolvidos na execução de unidades de trabalho

| Estereótipo | Descrição |
|---|---|
| *<<in>>* | Representa um insumo para a realização de uma tarefa.<br>*Produto de Trabalho => Entrada* |
| *<<out>>* | *Representa um resultado do trabalho realizado em uma tarefa.*<br>*Produto de Trabalho => Saída* |
| *<<inout>>* | Representa um artefato modificado durante a realização de uma tarefa.<br>*Produto de Trabalho => Entrada e Saída* |

### h. *WorkProductRoleRelationship* (LigaçãoPapelProdutoDeTrabalho)

#### Descrição

Estabelece um relacionamento de associação entre uma característica da categoria Produto de Trabalho (*WorkProductFeature*) e uma ou várias características da categoria Papel (*RoleFeature*) com algum tipo de responsabilidade sobre o produto de trabalho. A Figura 34 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de características que podem participar desse tipo de relacionamento.

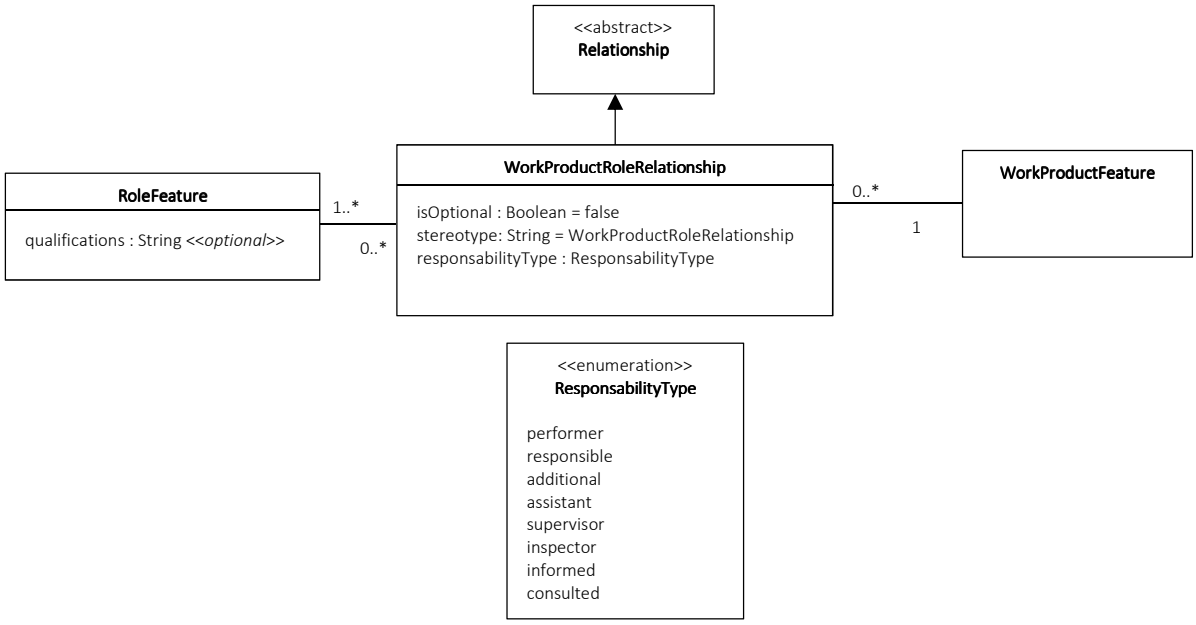



Figura 36 - Meta-modelo *OdysseyProcess-FEX: WorkProductRoleRelationship* **(LigaçãoPapelProdutoDeTrabalho)**

#### Hierarquia

Superclasse: *Relacionamentos*

#### Atributos

- ***isOptional*** **(ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação da característica Papel (*RoleFeature*) no consumo, produção ou modificação de um produto de trabalho (*WorkProductFeature*) por uma unidade de trabalho (*WorkUnit*). Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype*** **(estereótipo):** String = *workProductRoleRelationship* – atributo que especifica o tipo de associação criada.
- ***responsabilityType*** (tipoResponsabilidade): ResponsabilityType. A classe ResponsabilityType representa uma enumeração dos tipos de

responsabilidades que podem ser assumidos por um papel que possui algum tipo de responsabilidade sobre o produto de trabalho associado. De forma inicial foi definido um conjunto de responsabilidades possíveis representadas pelos estereótipos: *<<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>* (Tabela 25).

**Associações**

Sem associações específicas.

**Restrições**

Sem restrições específicas.

O pacote de Regras de Composição (*Composition Rules*) corresponde ao mesmo pacote da notação *Odyssey-FEX* (OLIVEIRA, 2006). Esse pacote contém as classes que definem a semântica das regras de dependência e mútua exclusividade entre características e pode ser visualizado na Figura 37.

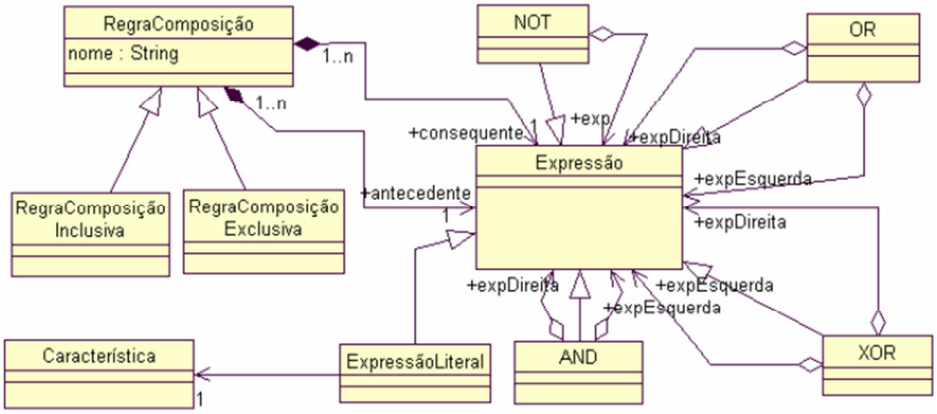



**Figura 37 - Meta-modelo OdysseyProcess-FEX: Pacote Regras de Composição (OLIVEIRA, 2006)**

A seguir cada elemento desse pacote será descrito de forma detalhada, seguindo uma estrutura de cinco itens de descrição: Descrição, Hierarquia, Atributos, Associações e Restrições. O detalhamento da descrição foi extraído da descrição do Pacote de Regras de Composição encontrada em (OLIVEIRA *et al.*, 2005).

**a. RegraComposição**

**Descrição**

Regras que definem restrições existentes entre características. Tais regras incluem relações do tipo "*exclui*" e "*requer*" entre características ou quaisquer conjuntos de características.

**Hierarquia**

Subclasses: *RegraComposiçãoInclusiva* e *RegraComposiçãoExclusiva*.

**Atributos**

- **nome**: String[1] – atributo que referencia o nome da Regra de Composição.

**Associações**

- **antecedente**: Expressão[1] – Indica a expressão antecedente de uma RegraComposição.
- **consequente**: Expressão[1] – Indica a expressão consequente de uma RegraComposição.

**Restrições**

[1] Uma Regra de Composição é formada por duas Expressões, uma como antecedente e outra como consequente.

[2] Uma Regra de Composição não pode ser contraditória, i.e., não pode ter antecedente e consequente iguais.

[3] Uma Regra de Composição não pode ter antecedente definido e consequente nulo ou vice versa.

[4] Características dependentes entre si não podem ser mutuamente exclusivas,

e vice-versa.

b. RegraComposiçãoInclusiva

Descrição

Regras de composição que indicam dependência entre duas ou mais características. Indicam as regras do tipo "*requer*".

Hierarquia

Superclasse: *RegraComposição*

Atributos

Sem atributos específicos.

Associações

Sem associações específicas.

Restrições

[1] Em uma Regra de Composição Inclusiva, o consequente só poderá ser opcional se o antecedente for opcional.

[2] Regras de Composição Inclusivas não são bidirecionais. Por exemplo, se uma característica A requer a característica B, e a característica B requer a característica A, existirão duas Regras de Composição.

c. RegraComposiçãoExclusiva

Descrição

Regras de composição que indicam mútua exclusividade entre duas ou mais características. Indicam as regras do tipo "*exclui*".

Hierarquia

Superclasse: *RegraComposição*

Atributos

Sem atributos específicos.

Associações

Sem associações específicas.

Restrições

[1] Uma Regra de Composição Exclusiva não deve envolver características mandatórias, somente características opcionais.

d. Expressão

Descrição

Expressões que constituem as Regras de Composição. Podem ser Booleanas ou Literais.

Hierarquia

Subclasses: *AND, OR, XOR, NOT, ExpressãoLiteral.*

Atributos

Sem atributos específicos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

e. ExpressaoLiteral

Descrição

Expressão Literal é a expressão mais elementar de uma Regra de Composição. É representada por uma única característica.

Hierarquia

Superclasse: *Expressão*

Atributos

- característica: Característica[1] - Característica que constitui a expressão literal.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

f. AND

Descrição

Expressão que representa o AND lógico.

Hierarquia

Superclasse: *Expressão*

Atributos

Sem atributos específicos.

Associações

- **expEsquerda**: Expressão[1] - representa a expressão que vem antes do conector AND. Pode ser uma expressão literal ou de qualquer outro tipo booleano.
- **expDireita**: Expressão[1] - representa a expressão que vem depois do conector AND. Pode ser uma expressão literal ou de qualquer outro tipo booleano.

Restrições

Sem restrições específicas.

g. OR

Descrição

Expressão que representa o OR lógico.

Hierarquia

Superclasse: *Expressão*

Atributos

Sem atributos específicos.

Associações

- **expEsquerda**: Expressão[1] - representa a expressão que vem antes do conector OR. Pode ser uma expressão literal ou de qualquer outro tipo booleano.
- **expDireita**: Expressão[1] - representa a expressão que vem depois do conector OR. Pode ser uma expressão literal ou de qualquer outro tipo booleano.

Restrições

Sem restrições específicas.

h. XOR

Descrição

Expressão que representa o XOR lógico.

Hierarquia

Superclasse: *Expressão*

Atributos

Sem atributos específicos.

Associações

- **expEsquerda**: Expressão[1] - representa a expressão que vem antes do conector XOR. Pode ser uma expressão literal ou de qualquer outro tipo booleano.

- **expDireita**: Expressão[1] - representa a expressão que vem depois do conector XOR. Pode ser uma expressão literal ou de qualquer outro tipo booleano.

**Restrições**

Sem restrições específicas.

i. **NOT**

**Descrição**

Expressão que representa o NOT lógico.

**Hierarquia**

Superclasse: *Expressão*

**Atributos**

Sem atributos específicos.

**Associações**

- **exp**: Expressão[1] - representa a expressão que vem depois do conector NOT. Pode ser uma expressão literal ou de qualquer outro tipo booleano.

**Restrições**

Sem restrições específicas.

# ANEXO II

## CHECKLISTS PARA VERIFICAÇÃO DOS MODELOS DE ANÁLISE DE DOMÍNIO DE PROCESSOS DE SOFTWARE

**Instruções:**

- Avalie cada modelo de domínio de acordo com os itens de avaliação abaixo correspondentes:

  1. Modelo de Características – Checklist para Visão Conceitual (Modelo de Características)

  2. Diagrama de Atividades – Checklist para Visão Comportamental

- A opção N.A. (Não Aplicável) deve ser marcada se for considerado que não é possível aplicar o item para avaliar o modelo em questão.
- Para cada opção Não ou Parcialmente marcada, justificar descrevendo o motivo da opção assinalada. Algumas perguntas adicionais ao item descrito podem auxiliar na descrição do problema encontrado. De forma geral, os elementos de processos que indicam o motivo devem ser citados na justificativa.
- Ao terminar a inspeção dos modelos, registrar o tempo utilizado:
  - Tempo Inspeção Modelo de Características: _____
  - Tempo Inspeção Modelo Comportamental: _____
  - Tempo Total de Inspeção: _____

## Checklist Verificação do Modelo de Features

| # | Item de verificação: Taxonomia | Resposta |
|---|---|---|
| 1 | **Todas as características do modelo foram descritas com clareza (semanticamente é possível entender o significado da característica)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Por quais características do modelo o item de verificação não foi atendido? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 2 | **A opcionalidade/ obrigatoriedade das características do modelo estão em conformidade com o descrito pelo domínio?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características foram classificadas de forma incorreta como mandatórias sendo opcionais?<br>Quais características foram classificadas de forma incorreta como opcionais sendo mandatórias? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 3 | **Os elementos de processo que representam atividades do domínio estão devidamente representados no modelo como *características da categoria atividade (Activity Feature)*?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características foram classificadas de forma incorreta?<br>Qual a classificação seria mais indicada para cada característica apontada? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 4 | **Os elementos de processo que representam tarefas do domínio de processos estão devidamente representados no modelo como *características da categoria tarefa (Task Feature)*?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características foram classificadas de forma incorreta?<br>Qual a classificação seria mais indicada para cada característica apontada? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 5 | **Os elementos de processo que representam papel do domínio de processos estão devidamente representados no modelo como *características da categoria papel (Role Feature)*?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características foram classificadas de forma incorreta?<br>Qual a classificação seria mais indicada para cada característica apontada? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 6 | **Os elementos de processo que representam produtos de trabalho do domínio de processos estão devidamente representados no modelo como *características da categoria produto de trabalho (Work Product Feature)*?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características foram classificadas de forma incorreta?<br>Qual a classificação seria mais indicada para cada característica apontada? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 7 | **As características que representam tarefas do domínio (*Task Feature*) possuem pelo menos uma característica da categoria papel (*Role Feature*) associada?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características tarefa não possuem papel associado? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 8 | **As características que representam tarefas do domínio (*Task Feature*) possuem pelo menos uma característica da categoria produto de trabalho (*WorkProduct Feature*) associada?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características tarefa não possuem produto de trabalho associado? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 9 | **As características que representam atividades do domínio (*Activity Feature*), quando não especificadas por unidades de trabalho de menor granularidade (tarefas), possuem pelo menos uma característica da categoria papel (*Role Feature*) associada?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características atividade não possuem papel associado? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 10 | **As características que representam atividades do domínio (*Activity Feature*), quando não especificadas por unidades de trabalho de menor granularidade (tarefas), possuem pelo menos uma característica da categoria produto de trabalho (*WorkProduct Feature*) associada?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características atividade não possuem produto de trabalho associado? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 11 | **Toda característica da categoria papel (*Role Feature*) está associada a pelo menos uma característica da categoria tarefa (*Task Feature*) ou da categoria atividade (*Activity Feature*)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características não possuem papel associado? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 12 | **Toda característica da categoria produto de trabalho (*Work Product Feature*) está associada a pelo menos uma característica da categoria tarefa (*Task Feature*) ou da categoria atividade (*Activity Feature*)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características não possuem produto de trabalho associado? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 13 | **Um pacote especificado como disciplina no domínio (*Discipline*) é composto / agrega unidades de trabalho: características da categoria atividade (*Activity Feature*) e características da categoria tarefa (*Task Feature*)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar. | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 14 | **Alguma característica do modelo, embora correta, está fora do escopo do modelo, não contribuindo para o entendimento do domínio?**<br>Neste caso, ao selecionar a resposta positiva, justificar enumerando as características identificadas. | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 15 | **Algum conceito relevante do domínio deixou de ser incluído no modelo? (Omissão)**<br>Neste caso, ao selecionar a resposta positiva, justificar enumerando as possíveis características a serem incluídas. | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 16 | **Existem características distintas no modelo que representam um mesmo elemento do domínio? (Duplicidade de Informação)**<br>Neste caso, ao selecionar a resposta positiva, justificar enumerando as características duplicadas. | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |

| # | Item de verificação: Relacionamentos | Resposta |
|---|---|---|
| 17 | **As situações do domínio em que uma ou mais de uma característica podem ser escolhidas dentro de um grupo de alternativas (ponto de variação e variantes) estão devidamente representadas no modelo?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características não foram devidamente agrupadas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 18 | **As cardinalidades dos pontos de variação do modelo estão corretas?**<br>• Para características classificadas como ponto de variação mandatório, o valor mínimo da cardinalidade foi definido com um valor >=1?<br>• Para características classificadas como ponto de variação opcional, o valor mínimo da cardinalidade foi definido com um valor = 0?<br>• Para características classificadas como um ponto de variação em que apenas uma alternativa pode ser escolhida, o valor máximo da cardinalidade foi definido com um valor = 1?<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características e suas cardinalidades não foram devidamente representadas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 19 | **Os relacionamentos de composição/agregação envolvendo uma característica da categoria atividade (*Activity Feature*) como origem da relação possuem apenas características da categoria atividade (*Activity Feature*) ou da categoria tarefa (*Task Feature*) como destino?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características não foram devidamente representadas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 20 | **Os relacionamentos de composição/agregação envolvendo uma característica da categoria tarefa (*Task Feature*) como origem da relação possuem apenas características da categoria tarefa (*Task Feature*) como destino?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características não foram devidamente representadas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 21 | **Os relacionamentos de composição/agregação envolvendo uma característica da categoria papel (*Role Feature*) como origem da relação possuem apenas características da categoria papel (*Role Feature*) como destino?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características não foram devidamente representadas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 22 | **Os relacionamentos de composição/agregação envolvendo uma característica da categoria produto de trabalho (*Work Product Feature*) como origem da relação possuem apenas características da categoria produto de trabalho (*Work Product Feature*) como destino?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais características não foram devidamente representadas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 23 | **Para cada característicada categoria papel (*Role Feature*) associada a uma característica da categoria atividade (*Activity Feature*) ou da categoria tarefa (*Task Feature*) ou da categoria produto de trabalho (*Work Product Feature*), a responsabilidade a ela atribuída na execução de tal unidade de trabalho foi devidamente estabelecida?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Em quais características a responsabilidade não foi devidamente representada? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 24 | **Para cada característica da categoria produto de trabalho (*Work Product Feature*) associada a uma característica da categoria atividade (*Activity Feature*) ou da categoria tarefa (*Task Feature*), seu tipo na execução de tal unidade de trabalho foi devidamente estabelecido (produto de trabalho requerido <<*in*>>, produzido <<*out*>> ou modificado< <*inout*>>)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Em quais características o tipo do artefato não foi devidamente representado? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 25 | **Duas ou mais características do modelo estão reunidas em um relacionamento, mas não é possível identificar este relacionamento no domínio?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais relacionamentos se encaixam nesse item? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 26 | **Existe algum relacionamento entre características que deixou de ser informado no modelo?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais relacionamentos foram omitidos? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |

| # | Item de verificação: Regras de Composição (Relações de Dependência e Mútua Exclusividade) | Resposta |
|---|---|---|
| 27 | **Relações de dependência foram devidamente representadas através de Regras de Composição Inclusivas?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais relacionamentos foram omitidos ou representados incorretamente? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 28 | **Relações de mútua exclusividade foram devidamente representadas através de Regras de Composição Exclusivas?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais relacionamentos foram omitidos ou representados incorretamente? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 29 | **Alguma regra de composição do modelo contraria outra regra de composição do mesmo modelo?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais relacionamentos foram representados incorretamente? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 30 | **Alguma regra de composição do modelo não se aplica ao domínio, embora possa estar correta?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais relacionamentos foram representados incorretamente? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 31 | **Todas as regras de composição necessárias para descrever o domínio foram devidamente representadas no modelo?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais relacionamentos foram omitidos? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |

Checklist Verificação do Modelo Comportamental

| # | Item de verificação | Resposta |
|---|---|---|
| 1 | **Todas as características do modelo de *features* que representam unidades de trabalho (atividades e tarefas) foram mapeadas para elementos de processos no modelo comportamental?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais elementos não foram representados no modelo comportamental? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 2 | **Os elementos de processos representados no modelo comportamental estão devidamente classificados quanto a sua variabilidade (ponto de variação, variante e invariante)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais elementos não foram devidamente classificados quanto à variabilidade? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 3 | **Os elementos de processos representados no modelo comportamental estão devidamente classificados quanto a sua opcionalidade (opcional ou mandatório)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais elementos não foram devidamente classificados quanto à opcionalidade? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 4 | **O fluxo de controle entre elementos de processos que representam unidades de trabalho (atividades/tarefas) foi especificado de acordo com o domínio descrito?**<br>• Cada unidade de trabalho possui pelo menos outra unidade de trabalho como sucessora?<br>Neste caso, apenas unidades de trabalho consideradas como finais de execução do fluxo podem não possuir outra unidade de trabalho na sequência de execução.<br>• Cada unidade de trabalho possui pelo menos outra unidade de trabalho como predecessora?<br>Neste caso, apenas unidades de trabalho consideradas como finais de execução do fluxo podem não possuir outra unidade de trabalho na sequência de execução.<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais elementos não possuem nenhum predecessor ou sucessor foram devidamente representadas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 5 | **Fluxos de controle opcionais foram devidamente representados como fluxos de controle conectados por um ponto de decisão em sua origem? Verificar trechos do modelo que contenham elementos classificados como opcionais.**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais fluxos opcionais não possuem nós de ponto de decisão? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 6 | **Unidades de trabalho que representam pontos de variação foram representadas por nós de decisão respeitando a propriedade de opcionalidade e cardinalidade?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais pontos de variação não estão corretamente representados? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 7 | **Em fluxos que apresentam restrições de dependência ou mútua exclusidade com outros fluxos, tais restrições foram devidamente representadas?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Em quais fluxos tais restrições não estão corretamente representadas ou estão omissas? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 8 | **Os fluxos de controle de unidades de trabalho que podem ser realizadas em paralelo foram devidamente representados através de nós de bifurcação (*fork node*)?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais fluxos paralelos não possuem nós de bifurcação? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 9 | **Os fluxos de controle entre unidades de trabalho síncronas foram devidamente representados através de nós de Junção (*join node)*?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais fluxos síncronos não possuem nós de junção? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |
| 10 | **Todos os fluxos de controle possíveis a partir de uma unidade de trabalho foram devidamente representados?**<br>Em caso da opção não atender ou atender parcialmente o item de verificação, justificar.<br>Quais fluxos não foram representados? | Sim ( ) Não ( )<br>Parcialmente ( )<br>N.A. ( ) |