ODYSSEYPROCESS-FEX: UMA ABORDAGEM PARA MODELAGEM DE VARIABILIDADES DE LINHA DE PROCESSOS DE SOFTWARE

Eldânae Nogueira Teixeira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadora: Cláudia Maria Lima Werner

Rio de Janeiro
Fevereiro de 2011

ODYSSEYPROCESS-FEX: UMA ABORDAGEM PARA MODELAGEM DE VARIABILIDADES DE LINHA DE PROCESSOS DE SOFTWARE

Eldânae Nogueira Teixeira

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

Examinada por:

_____
Prof.ª Cláudia Maria Lima Werner, D.Sc.

_____
Prof. Toacy Cavalcante de Oliveira, D.Sc.

_____
Prof.ª Carla Alessandra Lima Reis, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
FEVEREIRO DE 2011

Teixeira, Eldânae Nogueira

OdysseyProcess-FEX: Uma Abordagem para Modelagem de Variabilidades de Linha de Processos de Sofware/ Eldânae Nogueira Teixeira. – Rio de Janeiro: UFRJ/COPPE, 2011.

XII, 161 p.: il.; 29,7 cm.

Orientadora: Cláudia Maria Lima Werner

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 95 – 102.

1. Linha de Processos de Software. 2. Variabilidade. 3. Modelagem de Características. 4. Reutilização de Processos de Software I. Werner, Cláudia Maria Lima. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

"Não se deixe levar pelo extremismo.
Nem exagere para mais, nem para menos.
Saiba permanecer no meio-termo.
Se correr demais, cansará.
Se ficar muito parado, acabará consumindo o terreno
debaixo dos próprios pés, e dentro de pouco
estará pisando uma cova.
Não pare, mas também não queira correr demais.
Caminha firme e com segurança,
sem pressa, mas não se detenha jamais
na senda do progresso."

Trecho do Livro Minutos de Sabedoria

Agradecimentos

À Deus, pela fé e pelo sentido da vida.

Aos meus pais, que sempre me guiaram e dedicaram todo amor e confiança na minha capacidade e realização dos meus objetivos. Agradeço a paciência pelos momentos de angústia e a força transmitida para persistir no caminho até o final e para buscar novos desafios na jornada da vida.

Ao meu avô, que sempre se orgulhou de cada conquista e demonstrou seu incentivo através do carinho de um conselheiro.

Ao Diogo Bravo, pelo companheirismo e pela participação através da troca de ideias deste período de formação.

À professora Cláudia Werner, pela orientação, confiança, incentivo e oportunidades que contribuíram de forma essencial para o meu amadurecimento profissional e formação acadêmica.

Um agradecimento especial à Andréa Magdaleno, pela co-orientação, ainda que extra-oficial, pela participação ativa, pelas longas discussões que levaram a conclusão deste trabalho e perspectivas de trabalhos futuros.

Aos antigos e novos amigos do grupo de Reutilização de Software da COPPE/UFRJ, por todas as contribuições para o desenvolvimento do conhecimento adquirido.

Aos professores Toacy Oliveira e Carla Reis, por terem aceitado fazer parte desta banca.

A mim mesma, pela força interna que me move a não desistir dos meus sonhos e a buscar sempre mais.

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

ODYSSEYPROCESS-FEX: UMA ABORDAGEM PARA MODELAGEM DE VARIABILIDADES DE LINHA DE PROCESSOS DE SOFTWARE

Eldânae Nogueira Teixeira

Fevereiro/2011

Orientadora: Cláudia Maria Lima Werner

Programa: Engenharia de Sistemas e Computação

Diante da diversidade de objetivos e características de organizações e projetos, a tarefa de definição de processos de software torna-se não trivial. A aplicação de técnicas de reutilização de processos, como Linhas de Processos de Software, visa apoiar esta tarefa e contribuir para melhorias na produtividade, qualidade e adequação dos processos gerados, associadas à redução de riscos, esforços e custos envolvidos. Um dos desafios na construção de uma Linha de Processos de Software envolve a identificação e representação das similaridades e diferenças existentes em uma família de processos de software, atividade denominada modelagem de variabilidades.

Este trabalho de pesquisa está inserido em uma abordagem sistemática de Engenharia de Linha de Processos de Software, com foco na fase de Análise do Domínio e propõe uma representação, através de um modelo de características, das variabilidades e opcionalidades em artefatos reutilizáveis inerentes ao domínio de processos de software. Esta representação é formalizada através de um meta-modelo, que descreve a semântica dos conceitos envolvidos nessa modelagem, e uma notação, que define a simbologia gráfica dos elementos constituintes do meta-modelo, ambos denominados *OdysseyProcess-FEX*. Foi realizado um estudo preliminar para avaliar a viabilidade de aplicação do meta-modelo e notação propostos na atividade de construção de Linhas de Processos de Software. Um protótipo foi desenvolvido no contexto do ambiente Odyssey, para viabilizar a aplicação da abordagem proposta.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ODYSSEYPROCESS-FEX: AN APPROACH FOR SOFTWARE PROCESS LINE VARIABILITY MODELING

Eldânae Nogueira Teixeira

February/2011

Advisor: Cláudia Maria Lima Werner

Department: Computer and Systems Engineering

Given the diversity of objectives and features of organizations and projects, the software process definition task becomes a non-trivial one. The application of techniques for process reuse, such as Software Process Lines, aims to support this task and contribute to improvements in productivity, quality and adequacy of generated processes, associated with reduced risk, cost and effort involved. One of the challenges in the construction of a Software Process Line involves the identification and representation of the similarities and differences in a family of software processes, activity called variability modeling.

This work is part of a systematic approach for Software Process Line Engineering, focusing on the Domain Analysis phase and presents a representation of the variability and optionality in reusable artifacts of the software process domain, through a feature model. This representation is achieved through a meta-model, which describes the semantics of the concepts involved in this modeling, and a notation that defines the graphic symbols of the meta-model elements, both named *OdysseyProcess-FEX*. A preliminary study was performed to evaluate the feasibility of applying the proposed meta-model and notation on the construction activity of Software Processes Lines. A prototype was developed in the context of Odyssey environment, to facilitate the application of the proposed approach.

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 associada à flexibilidade de fornecer serviços e produtos de software de qualidade aos usuários. Abordagens, como a reutilização de software, têm sido amplamente utilizadas como formas de atender a essas necessidades. A reutilização de software é a disciplina responsável pela criação de sistemas de software a partir de software preexistente (KRUEGER, 1992). Dentro dessa área, para que a reutilização seja realizada de forma efetiva, são utilizadas técnicas que a conduzem de forma sistemática em todas as fases do desenvolvimento, 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). Ambas as técnicas incorporam uma etapa denominada Análise de Domínio (AD), cujo intuito é a identificação e análise de conhecimento sobre uma coleção de sistemas, visando explicitar seu conjunto de similaridades e diferenças. Esse conhecimento é representado através de um modelo genérico, denominado Modelo de Domínio.

Ao mesmo tempo em que os produtos de software estão se tornando cada vez maiores e mais complexos, a exigência por qualidade nestes produtos também tem aumentado. A qualidade desses produtos depende da qualidade dos processos de desenvolvimento adotados para construí-los (CUGOLA e GHEZZI, 1998, FUGGETTA, 2000, OSTERWEIL, 1987). 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).

No entanto, a diversidade de organizações e projetos, suas características e objetivos torna a tarefa de definição de processos de software não trivial. 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). 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).

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, KELLNER (1996) destaca que o conhecimento de técnicas de reutilização de produtos de software, tais como: arquiteturas povoadas com componentes reusáveis; gerenciamento de repositórios para armazenar, catalogar, procurar, acessar, etc. ativos reusáveis; gerência de configuração de ativos reusáveis, entre outros, poderiam ser aplicados a processos de software. Assim, técnicas de reutilização de software têm sido adaptadas ao contexto de definição de processos de software (KELLNER, 1996, WASHIZAKI, 2006). O propósito 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).

1.2 – CARACTERIZAÇÃO DO PROBLEMA

Uma das técnicas de reutilização de processos surgiu com base nos conceitos provenientes da abordagem de LPS. O termo *Linha de Processos* é proposto em abordagens (BARRETO, 2007, JAUFMAN e MÜNCH, 2005, ROMBACH, 2006, WASHIZAKI, 2006) que trabalham o conceito de LPS cujos produtos são processos. Segundo WASHIZAKI (2006), uma Linha de Processos pode ser definida como "um conjunto de processos em um domínio particular, ou destinados a um propósito em particular, que possuem características comuns e que são construídos baseados em artefatos de processos comuns e reusáveis".

Embora cada abordagem tenha sua forma de tratar a reutilização de processos e construção da linha, estas devem incorporar uma atividade de análise dos aspectos comuns e variáveis dentro do domínio de processos analisado. Essa análise dá origem ao conceito de variabilidade. A variabilidade em um domínio de processos de software se mostra importante no sentido de deixar explícitos os pontos onde tais processos são similares e, portanto, podem ser reutilizados, e os pontos onde estes divergem, caso em que devem receber tratamento específico. Para tanto, esse conhecimento deve ser documentado nos diferentes artefatos que constituem toda a Linha de Processos de Software, referentes a cada fase envolvida: análise, projeto e implementação.

No entanto, diferentemente da LPS, a abordagem de Linha de Processos é recente e não está ainda totalmente consolidada na literatura (BARRETO *et al*., 2009). Em geral, as abordagens existentes dão maior ênfase na especificação em nível de projeto, modelando arquiteturas de componentes de processos. Existe pouco trabalho investigando formas de promover um maior detalhamento da fase de análise, com a representação de uma linha de processos de software e as variabilidades dos elementos que constituem o domínio envolvido em um modelo de mais alto nível de abstração, como o modelo de características, amplamente utilizado em LPS como ponto de partida para o recorte necessário à instanciação de novos elementos a partir da linha. Neste modelo, uma característica pode ser definida como um aspecto, uma qualidade, ou uma característica visível ao usuário, proeminente ou distinta, de um sistema (ou sistemas) de software (KANG *et al*., 1990).

Uma análise da literatura apontou trabalhos que propunham a utilização de alguma representação para a modelagem de variabilidades em linha de processos (WASHIZAKI, 2006, ARMBRUST *et al*., 2009, SIMIDCHIEVA *et al.,* 2007, BARRETO *et al.*, 2010, MARTÍNEZ-RUIZ *et al*., 2008, ALEIXO *et al*., 2010). Com esta análise, foi possível identificar algumas limitações nas propostas de representação existentes:

- Conceitos inerentes à variabilidade do domínio de processos de software (e.g., elementos de processos fixos, elementos de processos configuráveis, alternativas de um ponto de configuração do domínio, conceito de opcionalidade e relações de dependência) não são representados de maneira completa e explícita; e
- As representações apresentam uma semântica variada e escopos diferentes de elementos envolvidos na modelagem de variabilidades, podendo levar a interpretações ambíguas.

Essa modelagem de variabilidades requer uma representação explícita de todos os conceitos inerentes à reutilização de processos de software, de forma a viabilizar o entendimento de todos os envolvidos com esta atividade. Desta forma, deve contemplar as diferentes etapas envolvidas, estando representada nos diversos artefatos constituintes da Linha de Processos. No entanto, a maioria das representações propostas foca no nível de projeto e apresentam limitações que podem ocasionar uma representação inadequada da variabilidade, gerando uma modelagem incompleta que será posteriormente reutilizada. Esta situação pode resultar em uma ameaça a conquista dos benefícios almejados com a reutilização de processos de software.

A partir desta análise, observa-se a necessidade de uma representação de variabilidades adequada à fase de análise e que seja trabalhada para a área de reutilização de processos de software dentro da abordagem de Linha de Processos de Software.

1.3 – CONTEXTO DO TRABALHO

O trabalho de pesquisa aqui desenvolvido está inserido em um contexto de pesquisa mais amplo dentro do Grupo de Reutilização de Software do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ e do Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec) da UNIRIO. O objetivo final é apoiar o gerente de projeto na tomada de decisão sobre a melhor forma de adaptar um processo específico de projeto. Para isso, a solução proposta por Magdaleno (2010) é dividida em quatro partes principais para alcançar a adaptação de processos de desenvolvimento de software de forma balanceada, sistemática e dinâmica (Figura 1.).

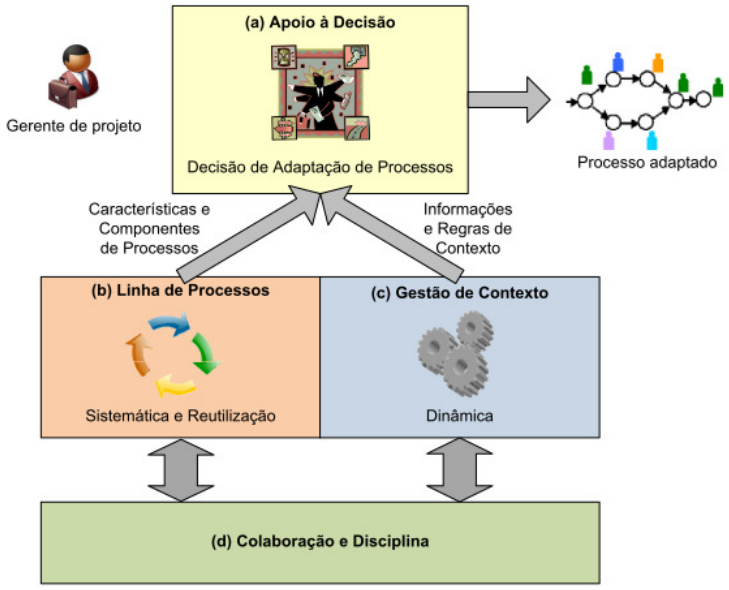


Figura 1.1 – Visão Geral do Projeto de Pesquisa de Apoio à Decisão para Adaptação Dinâmica de Processos de Software (MAGDALENO, 2010).

Este trabalho está relacionado ao tema de Linha de Processos de Software, correspondente ao desenvolvimento de uma estrutura sistemática de reutilização de processos (Figura 1.1b). Em conjunto, o grupo construiu uma definição para Linha de Processos de Software e definiu em linhas gerais a sistemática de construção, uso e gerenciamento da linha (MAGDALENO, 2010). Essa estrutura consiste na especificação das etapas que constituem as duas grandes fases da abordagem: (1) o ciclo de desenvolvimento e (2) o ciclo de aplicação de uma Linha de Processos de Software.

Na primeira fase, os artefatos de processos reutilizáveis são criados através da especificação de modelos que evidenciam os pontos comuns e variáveis em um domínio de processos. A primeira atividade a ser realizada nesta fase é denominada Análise do Domínio, e corresponde à identificação do conhecimento do domínio, seus conceitos e artefatos, e a representação explícita de suas variabilidades e opcionalidades, em um nível de abstração de fácil entendimento. É nesta atividade que se concentra este trabalho de pesquisa.

1.4 – ENFOQUE DE SOLUÇÃO

Este trabalho encontra-se inserido em uma abordagem sistemática de Engenharia de Linha de Processos de Software. Diante das questões levantadas, o foco está na fase de Análise do Domínio de Processos de Software, com a proposta de uma representação adequada de variabilidades em artefatos reutilizáveis inerentes ao domínio de processos de software, que envolve: 1) um meta-modelo, que formaliza a semântica dos conceitos envolvidos nessa modelagem; e 2) uma notação para a representação gráfica, em um modelo de características, dos conceitos definidos pelo meta-modelo.

A abordagem de Engenharia de Linha de Processos de Software possui quatro elementos principais: um método, um meta-modelo, uma notação e um ferramental de apoio. Este trabalho apresenta contribuições em cada um dos elementos mencionados. Na parte do método, o foco está na fase de Análise de Domínio de Processos de Software, definindo as atividades envolvidas e seus produtos de trabalho. Além disso, um meta-modelo é proposto com o intuito de formalizar a semântica e servir de referência para a construção de modelos de características do domínio de processos de software. Este meta-modelo visa explicitar a representação de conceitos de elementos de processos e de variabilidades do domínio. Possui um conjunto de restrições e propriedades, que juntas constituem as regras de boa formação do modelo. Essas regras direcionam a construção e a verificação de consistência de um modelo de características do domínio de processos de software. Junto ao meta-modelo proposto é apresentada a notação *OdysseyProcess-FEX,* que tem como objetivo representar simbolicamente os conceitos formalizados pelo meta-modelo.

Este trabalho foi desenvolvido dentro do contexto do ambiente Odyssey (ODYSSEY, 2011), que é uma infraestrutura de reutilização baseada em modelos de domínio. Esse ambiente contempla atividades do desenvolvimento de software *para* reutilização, processo de Engenharia de Domínio (ED), e atividades do

desenvolvimento *com* reutilização, processo de Engenharia de Aplicação (EA), com foco no reaproveitamento e adaptação dos componentes do domínio no desenvolvimento de uma aplicação. Este trabalho também propõe a adaptação do ambiente como ferramental de apoio automatizado para a construção de Linhas de Processos de Software, através da modelagem de características proposta, visando facilitar a representação das variabilidades.

1.5 – ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho está organizado em seis capítulos. O Capítulo 2 apresenta uma revisão das abordagens de Linha de Processos existentes. Neste capítulo, são identificados requisitos desejáveis para uma representação de variabilidades e, com base nesses requisitos, uma análise é conduzida sobre as representações existentes na literatura desta área.

O Capítulo 3 trata da abordagem proposta. Apresenta em linhas gerais a abordagem de Engenharia de Linha de Processos de Software e descreve a fase de Análise do Domínio de Processos de Software. O meta-modelo e a notação *OdysseyProcess-FEX* são especificados e apresentados, junto às regras de boa formação.

O Capítulo 4 descreve o planejamento e a realização de um estudo de observação que visa analisar a viabilidade de aplicação do meta-modelo e notação propostos na atividade de construção de Linhas de Processos de Software.

O Capítulo 5 descreve a implementação da proposta no contexto do ambiente Odyssey. Foram realizadas adaptações para flexibilizar o ambiente a trabalhar com Linhas de Processos de Software. O maior volume de trabalho se concentrou na disponibilização da notação *OdysseyProcess-FEX* para viabilizar a modelagem de características do domínio de processos de software.

O Capítulo 6 apresenta as contribuições, limitações e trabalhos futuros.

Por fim, os Anexos incluem especificações detalhadas do meta-modelo *OdysseyProcess-FEX* proposto e os documentos utilizados para aplicação do estudo descrito no Capítulo 4.

Capítulo II
REVISÃO DA LITERATURA

2.1 – INTRODUÇÃO

Abordagens, como a reutilização de software, têm sido amplamente utilizadas como formas de atender as crescentes demandas impostas pela necessidade de fornecer serviços e produtos de software de qualidade aos usuários. Dentro da área de reutilização, para que esta seja efetiva, destacamos técnicas que a conduzem de forma sistemática em todas as fases do desenvolvimento, tais 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). Ambas as técnicas incorporam uma etapa de Análise de Domínio (AD), que pode ser definida como o processo de identificar e organizar o conhecimento a respeito de uma classe de problemas, de maneira a apoiar a descrição e solução de tais problemas (ARANGO e PRIETO-DIAZ, 1991). Como resultado desta etapa, temos a identificação, aquisição e representação do conhecimento referente a um domínio, destacando suas variabilidades representadas em um modelo denominado Modelo do Domínio. Tal modelagem de variabilidades é abordada na literatura por diversos trabalhos que orientam sua representação em diferentes artefatos do ciclo de vida de um software, desde modelos de casos de uso, passando pelo uso de extensões da UML para a representação em modelos de classes, até trabalhos que abordam a representação de variabilidades em um modelo de características (OLIVEIRA, 2006).

Diante da aplicação da analogia existente entre produto de software e processo de software (OSTERWEIL, 1987) no campo da reutilização, técnicas como as mencionadas anteriormente têm sido adaptadas ao contexto de definição de processos de software (KELLNER, 1996, WASHIZAKI, 2006). Assim, a abordagem de *Linha de Processos* surgiu como uma técnica sistemática de reutilização de processos e foi apresentada em abordagens (BARRETO, 2007, JAUFMAN e MÜNCH, 2005, ROMBACH, 2006, WASHIZAKI, 2006) que aplicam a ideia de LPS em processos. Encontramos na literatura algumas definições e apresentação de ideias similares ao termo proposto. No entanto, nenhum consenso foi alcançado ainda. A abordagem de Linha de Processos é recente na literatura e não há muitos trabalhos publicados sobre o assunto (BARRETO *et al*., 2009). Ainda, segundo WASHIZAKI (2006), esta definição ainda não está bem definida e não é suficiente para a criação de um *framework* concreto.

Assim, o objetivo deste capítulo é apresentar uma análise dos trabalhos propostos na área de Linha de Processos de Software, com destaque para o enfoque dado por cada solução. Uma análise da modelagem de variabilidades dentro de cada técnica foi realizada, levando em consideração alguns requisitos, necessários à representação de variabilidades em um modelo de características dentro do domínio de processos de software, conforme destacado na Seção 2.4.

Este capítulo dedica-se a abordar conceitos que formam a base para a elaboração deste trabalho. A Seção 2.2 resume as técnicas de ED e LPS e os principais conceitos envolvidos na modelagem de variabilidades do domínio de software. A Seção 2.3 trata dos conceitos de processos de software e da área de reutilização de processos. A Seção 2.4 aborda a técnica de Linha de Processos de Software, suas diferentes formas de representação e a modelagem de variabilidades no domínio de processos de software. Por fim, na Seção 2.5, são feitas algumas considerações a partir dos conceitos e análises observados.

2.2 – REUTILIZAÇÃO DE SOFTWARE

A Reutilização de Software é a disciplina responsável pela criação de sistemas de software a partir de software preexistente (KRUEGER, 1992). Aponta como benefícios melhores índices de produtividade, melhoria na qualidade e confiabilidade do software, redução no tempo e nos custos envolvidos no desenvolvimento de software (FRAKES e KANG, 2005), dentre outros. As abordagens de ED e LPS, conforme visto anteriormente são consideradas formas sistemáticas de promover a reutilização de software em todas as fases do desenvolvimento. Estas duas abordagens são detalhadas nas próximas seções.

2.2.1 – Engenharia de Domínio

Segundo ARANGO e PRIETO-DIAZ (1991), a ED é o processo de identificação e organização do conhecimento sobre uma classe de problemas, o domínio do problema, para suportar sua descrição e solução. O conceito de domínio pode ser entendido como uma classe de sistemas que apresentam funcionalidades similares.

O processo de ED pode ser dividido em três fases principais: Análise do Domínio (AD), Projeto do Domínio e Implementação do Domínio (KANG *et al*., 1990, GRISS *et al.*, 1998, SIMOS e ANTHONY, 1998, BRAGA, 2000). A etapa de AD abrange a identificação, aquisição e representação do conhecimento referente a um domínio, destacando suas variabilidades dentro de uma família de aplicações. A

especificação da infraestrutura, ou Projeto do Domínio, gera como resultado uma arquitetura que permite identificar os ativos de um domínio, elementos relacionados ao ciclo de vida de um software que foram projetados para utilização em diferentes contextos (i.e., componentes que são relevantes para a reutilização, a forma de relacionamento entre eles, assim como as formas de customização disponíveis, a distribuição destes componentes pelos repositórios e outras especificações). A Implementação do Domínio compreende o processo de geração dos modelos implementacionais, que inclui a identificação, aquisição, extração de sistemas existentes, adaptação e/ou produção e manutenção dos artefatos reutilizáveis.

As atividades envolvidas na ED consistem no processo de desenvolvimento *para* reutilização que gera artefatos reutilizáveis pelo processo de desenvolvimento *com* reutilização, caracterizado pela Engenharia de Aplicação (EA - Figura 2.1).

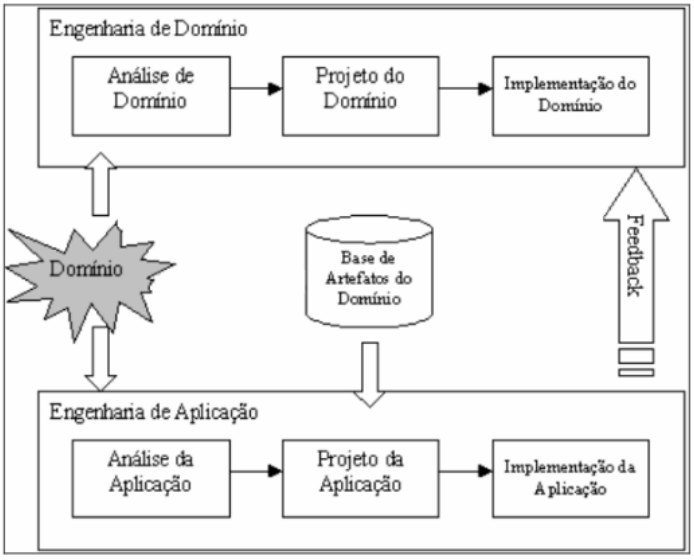


Figura 2.1 - Ciclo de vida de Engenharia de Domínio e Engenharia de Aplicação (Adaptado de ATKINSON *et al*., 2002)

A EA consiste, inicialmente, na avaliação do domínio e da afinidade da família de sistemas com o produto ou aplicação que será construído. A seguir, a seleção dos componentes necessários à aplicação é feita em um alto nível de abstração, por meio do modelo do domínio, descendo gradualmente em níveis de abstração, até conseguir atingir os componentes implementados ou semi-desenvolvidos do domínio (MILER, 2000). Esse recorte é fortemente influenciado pela variabilidade modelada no domínio, uma vez que as características variáveis em artefatos do domínio determinam o tipo de aplicação que será instanciada (OLIVEIRA, 2006).

2.2.2 – Linha de Produtos de Software

A LPS é definida, segundo o SEI (*Software Engineering Institute*), como sendo 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). Uma LPS funciona como uma fábrica, que instancia produtos com características similares, mas, ao mesmo tempo, com algumas características específicas que os diferenciam, através da composição de componentes existentes (GARG *et al.*, 2003, NORTHROP, 2002). A abordagem pode ser dividida em três atividades essenciais, fortemente relacionadas entre si: o Desenvolvimento do Núcleo de Artefatos, o Desenvolvimento de Produtos e o Gerenciamento da Linha de Produtos (Figura 2.2).

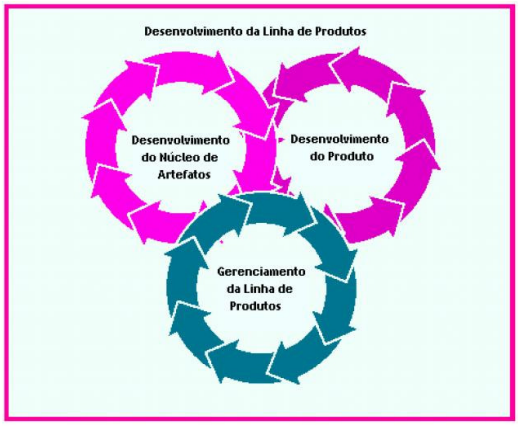


Figura 2.2 - Atividades Essenciais da Linha de Produtos
(Adaptado de NORTHROP, 2002)

O Núcleo de Artefatos constitui a base do paradigma de Linha de Produtos e inclui uma arquitetura de software de referência, componentes de software reutilizáveis, modelos de domínio, requisitos, cronogramas, orçamentos, planos e casos de teste, planejamento e descrição de processos, dentre outros artefatos da linha.

O Desenvolvimento de Produto se utiliza dos resultados gerados pela atividade de Desenvolvimento do Núcleo de Artefatos e envolve atividades como a análise baseada no modelo de domínio, a instanciação da arquitetura do produto, que gera a arquitetura específica do produto em questão, a partir da seleção incremental dos pontos de variabilidade de uma arquitetura genérica, e a atividade de povoamento da arquitetura.

O Gerenciamento de LPS consiste na atividade de supervisionar, coordenar e documentar as atividades anteriormente mencionadas. É necessário que esse compromisso seja estabelecido pela organização para que a abordagem de LPS seja viável e executada com sucesso.

2.2.3 – Análise do Domínio de Software e a Modelagem de Características

Ambas as técnicas mencionadas incorporam no processo a fase de AD, que visa coletar informações sobre o domínio e explicitar seus aspectos similares e diferentes. A análise de tais aspectos dá origem ao conceito de variabilidade. A variabilidade em uma família de sistemas de software se mostra importante no sentido de deixar explícitos os pontos onde tais sistemas se assemelham, e, portanto, podem ser reutilizados, e os pontos onde eles diferem entre si, devendo receber tratamento específico (OLIVEIRA, 2006). O conceito de variabilidade pode ser então 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). Desta forma, a variabilidade oferece flexibilidade para diferenciar e diversificar os produtos (CHEN *et al.,* 2009).

O conceito de opcionalidade, que visa indicar a obrigatoriedade ou não da presença de determinado elemento em um produto específico, ou aplicação, a ser desenvolvido dentro do domínio como um todo, também é de relevante importância na identificação e representação das variabilidades de um domínio.

Alguns conceitos são inerentes à variabilidade e sua modelagem, tais como (OLIVEIRA, 2006):

- **Pontos de variação**: são as características que refletem a parametrização no domínio de uma maneira abstrata. São configuráveis por meio das variantes;
- **Variantes**: são características que atuam como alternativas para se configurar um ponto de variação;
- **Invariantes**: são as características fixas, que representam elementos não configuráveis em um domínio. Na literatura, em geral, não há uma nomenclatura bem definida para esses elementos do domínio;
- **Elementos Opcionais**: descrevem elementos que podem ou não estar presentes em produtos desenvolvidos em uma LPS, ou em aplicações instanciadas a partir de um domínio; e

- **Elementos Mandatórios**: descrevem elementos que devem obrigatoriamente estar presentes em produtos desenvolvidos em uma LPS, ou em aplicações instanciadas a partir de um domínio.

Dessa forma, pode-se notar a importância do processo de identificação das variabilidades de um domínio em uma abordagem de reutilização. Sua modelagem deve permitir o entendimento de todos os envolvidos no processo, em todas as fases do ciclo de vida de um software. Esse conhecimento resultante da AD é representado através de um modelo genérico, denominado Modelo de Domínio.

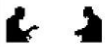
A notação para modelagem de variabilidade pode ser gráfica, textual ou a combinação de ambas as formas de representação. Uma das maneiras de especificar esse conhecimento adquirido pode ser pelo enfoque de modelagem de características. A modelagem de características é uma abordagem que trata da complexidade em expressar diversos requisitos em forma de características e da sua estruturação hierárquica em diagramas de características (MASSEN e LICHTER, 2004). Tal modelo é considerado um modelo de mais alto nível, bem como o ponto de partida para a reutilização de artefatos em um processo de ED ou LPS (OLIVEIRA, 2006). Pela definição de KANG *et al*. (1990), uma característica representa "um aspecto, uma qualidade, ou uma característica visível ao usuário, proeminente ou distinta, de um sistema (ou sistemas) de software". Esse modelo provê a base para o desenvolvimento, parametrização e configuração de artefatos reutilizáveis (KANG *et al*., 2002).

Essa modelagem de características pode ser realizada através de diferentes notações, como a notação FODA (KANG *et al*., 1990), a notação FORM (KANG *et al.,* 2002, LEE *et al.,* 2002), a notação FeatuRSEB (GRISS *et al.,* 1998), a notação de RIEBISCH (RIEBISCH *et al.,* 2002), a notação de CECHTICKY (CECHTICKY *et al.,* 2004), a notação de CZARNECKI (CZARNECKI *et al.,* 2004, 2005), a notação *Odyssey-FEX* (OLIVEIRA, 2006) e a notação definida por GOMAA (2004), dentre outras. Essas notações apresentam alguns conceitos com a mesma semântica, mas, por outro lado, apresentam particularidades que influenciam na modelagem do domínio e as difere entre si. A escolha da notação mais apropriada pode estar relacionada a uma série de fatores, tais como: maior conhecimento e familiaridade da equipe, popularização de uma determinada notação, disponibilidade de ambiente de desenvolvimento que suporte a notação, ou maior adequação aos requisitos que atendam à modelagem. Por isso, não existe uma notação padrão para a modelagem de características (TEIXEIRA, 2008).

Com destaque pela sua expressividade de representação (TEIXEIRA, 2008), a notação *Odyssey-FEX* define a semântica dos elementos que representam conceitos, funcionalidades e tecnologias utilizadas em um domínio, incluindo sua variabilidade, bem como os relacionamentos entre eles (OLIVEIRA, 2006). A definição desta notação teve como base o trabalho de KANG *et al.* (1990), que propõe o método FODA (*Feature Oriented Domain Analysis),* um método de ED precursor das notações baseadas em modelos de características. Além disso, incorpora e estende elementos do modelo de características do ambiente Odyssey, definidos na proposta de MILER (2000).

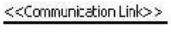
Na notação *Odyssey-FEX*, as características podem ser classificadas quanto à sua categoria, variabilidade e opcionalidade. As categorias propostas na notação classificam as características de acordo com as diferentes fases de desenvolvimento do software, dividindo em características de análise e características de projeto (Tabela 2.1). Quanto à variabilidade, as características classificam-se em pontos de variação; variantes e invariantes. Quanto à classificação de opcionalidade, as características podem ser classificadas como mandatórias ou opcionais.

Tabela 2.1 - Tipos de características na notação *Odyssey-FEX* (OLIVEIRA, 2006)

Ícone	Tipo de Característica	
	**Características de Domínio** – Características intimamente ligadas à essência do domínio. Representam as funcionalidades e/ou os conceitos do modelo e correspondem a casos de uso e componentes estruturais concretos.	Características de Análise
	**Características de Entidade** – São os atores do modelo. Entidades do mundo real que atuam sobre o domínio. Podem, por exemplo, expor a necessidade de uma interface com o usuário ou de procedimentos de controle.	
	**Características de Ambiente Operacional** - Características que representam atributos de um ambiente que uma aplicação do domínio pode usar e operar. Ex: tipo de terminal, sistemas operacionais, bibliotecas etc.	Características de Projeto (Tecnológicas)
	**Características de Tecnologia de Domínio** - Características que representam tecnologias utilizadas para modelar ou implementar questões específicas de um domínio. Ex: métodos de navegação em um domínio de aviões.	
	**Características de Técnicas de Implementação** – Características que representam tecnologias utilizadas para implementar outras características, podendo ser compartilhadas por diversos domínios. Ex: técnicas de sincronização.	

A notação permite uma disposição de características em forma de grafo, e disponibiliza os relacionamentos-padrão da UML (herança, composição, agregação e associação), além de relacionamentos específicos do modelo (alternativo, ligação de comunicação, "implementado por"), com o objetivo de evidenciar a relação existente entre as características com maior semântica (Tabela 2.2).

Tabela 2.2 - Relacionamentos da notação *Odyssey-FEX* (OLIVEIRA, 2006)

Representação	Descrição	
	**Composição** – Relacionamento em que uma característica é composta de várias outras. Denota relação na qual uma característica é parte fundamental de outra, de forma que a primeira não existe sem a segunda.	Relacionamentos da UML
	**Agregação** – Relacionamento em que uma característica representa o todo, e as outras as partes. Similar à composição, porém as características envolvidas existem independentemente uma da outra.	
	**Generalização** – Relacionamento em que há uma generalização/especialização das características. Este tipo de relacionamento indica que as características mais especializadas (filhas) herdam as propriedades e atributos de características mais generalizadas (antecessores).	
	Associação – Relacionamento simples entre duas características. Denota algum tipo de ligação entre seus membros. Pode ser nomeada, indicando um tipo específico de ligação.	
	**Alternativo** (*Alternative*) - Relacionamento entre um ponto de variação e suas variantes, denota a pertinência de uma variante a um determinado ponto de variação.	Relacionamentos Específicos na Odyssey-FEX
	**Implementado por** *(Implemented By)* - Relacionamento entre Características de Domínio e Características Tecnológicas, ou entre Características Tecnológicas que se encontrem em camadas diferentes.	
	**Ligação de Comunicação** *(Communication Link)* - Relacionamento existente entre Características de Entidade e Características de Domínio. Cumpre o mesmo papel do relacionamento de associação entre atores e casos de uso na UML (OMG, 2004)	

Além disso, apresenta Regras de Composição Complexas para expressar restrições existentes entre características. Essas regras definem relações de dependência entre duas ou mais características (Regra de Composição Inclusiva) ou de mútua exclusividade (Regra de Composição Exclusiva), onde duas ou mais características não devem ser escolhidas em conjunto em um mesmo produto ou aplicação.

Na Figura 2.3, encontra-se um modelo de característica construído utilizando a notação *Odyssey-FEX* e levando em consideração um domínio de Telefonia Móvel, que abrange conceitos e funcionalidades que podem estar presentes em um software desenvolvido para um telefone celular. Vale ressaltar a representação de quatro pontos de variação no modelo: *Jogos*, *Cores no visor, Conexão* e *Recebimento de Toques Musicais*, que indicam pontos de configuração do sistema para a geração de uma aplicação específica. A relação de um ponto de variação com suas variantes é realizada através de um relacionamento alternativo, como podemos notar na relação do ponto de variação *Recebimento de Toques Musicais* e suas variantes *Toque Monofônico, Toque*

Polifônico e *MP3*. As características *Transferência de Dados para PC*, *USB* e *Cabo de Dados* apresentam uma regra de composição inclusiva, significando que "*Transferência de Dados para PC AND USB*) **requer** *Cabo de Dados*", evidenciada pela marcação R_1. As classificações quanto a categorias é realizada através da presença dos estereótipos, como por exemplo, o estereótipo <<*conceptual*>> das características Telefone Celular e Jogos, que representam conceitos do domínio.

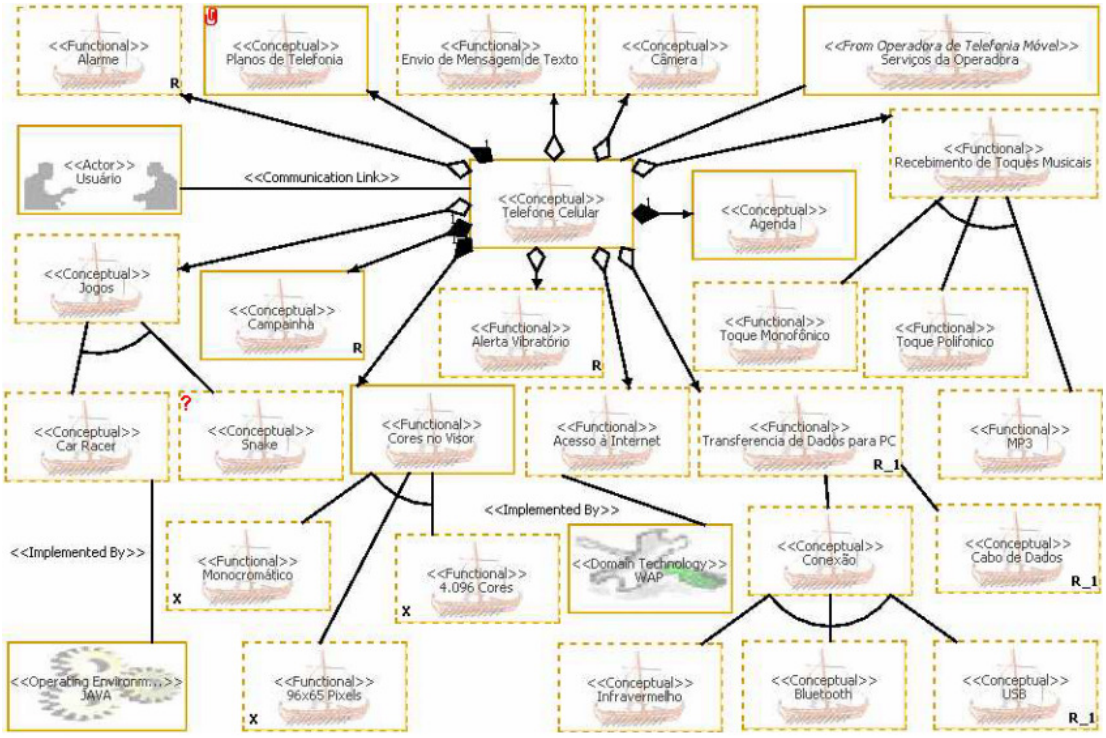


Figura 2.3 - Exemplo de Utilização da Notação *Odyssey-FEX* (OLIVEIRA, 2006)

2.3 – PROCESSO DE SOFTWARE E REUTILIZAÇÃO DE PROCESSOS

Um modelo de desenvolvimento de software é um conjunto de práticas recomendadas para o desenvolvimento de software (SOMMERVILLE, 2004). Estas práticas são organizadas em um processo de software que corresponde ao "conjunto de tarefas de Engenharia de Software necessárias para transformar os requisitos dos usuários em software" (HUMPHREY, 1989). Um processo de software pode ser então 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). Um dos principais motivos para que uma organização de software se preocupe em possuir processos de software bem definidos é o fato da qualidade do produto de software desenvolvido estar

diretamente ligada à qualidade do processo de software (SEGRINI, 2009).

Modelos de processo de software são estruturas complexas, que inter-relacionam diferentes aspectos tecnológicos, organizacionais e sociais para descrever o processo de desenvolvimento de software (REIS, 2002). 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). Essencialmente, um processo é composto por um conjunto de atividades. Estas atividades são partes bem caracterizadas do trabalho, realizadas em certo momento por papéis, de acordo com um conjunto de regras definidas que estabelecem a ordem e as condições em que as atividades devem ser executadas. Cada atividade manipula um conjunto de produtos de trabalho (dados, documentos ou formulários) durante sua execução (ARAUJO e BORGES, 2001). Desta forma, diferentes elementos de um processo, por exemplo, atividades, artefatos, recursos e papéis, podem ser modelados. Sem alcançar um consenso sobre o conjunto de elementos principais que devem ser modelados, propostas de representação têm sido apresentadas como forma de estabelecer um entendimento comum e minimizar o problema da falta de uma padronização dos conceitos envolvidos em um modelo de processo. Como exemplo, temos a criação de ontologias e meta-modelos.

A definição de uma ontologia de processo de software tem por objetivo apoiar a aquisição, organização, reúso e compartilhamento de conhecimento sobre processos de desenvolvimento de software. Para atingir este objetivo, a ontologia deve prover um vocabulário e um conjunto de axiomas, fixando a semântica dos termos neste domínio. Em FALBO (1998), uma ontologia foi desenvolvida. Esta ontologia é descrita em níveis para o domínio de processos de software e foi evoluída mais recentemente (FALBO e BERTOLLO, 2005). Nesta abordagem, processo de software é definido como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software. Esta ontologia é dividida em subontologias de atividade, recurso e procedimento. A Figura 2.4 mostra um modelo parcial da ontologia proposta.

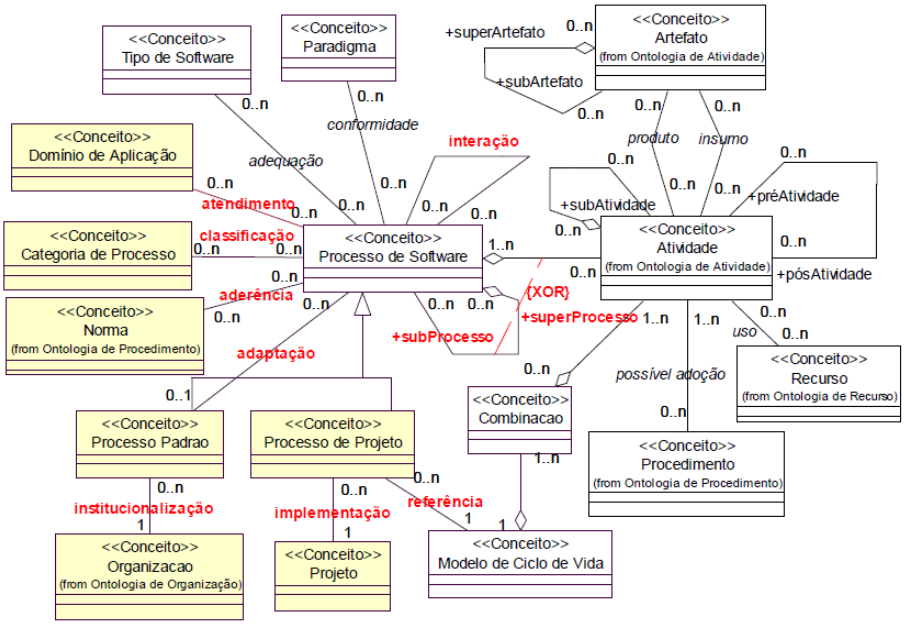


Figura 2.4 - Modelo Parcial da Ontologia de Processos de Software
(Adaptado de BERTOLLO *et al.,* 2006)

Um meta-modelo, denominado 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. O meta-modelo é dividido em sete pacotes:

- *Core:* contém todas as classes e abstrações que fazem parte da base do meta-modelo;
- *Process Structure:* contém os elementos necessários para definição dos modelos de processos;
- *Process Behaviour:* modela o comportamento de processos;
- *Managed Content:* contém os elementos necessários para gerenciar descrições textuais de métodos;
- *Method Content:* contém os conceitos para definir métodos;

- *Process with Methods:* possui os meios para a integração de métodos e processos; e
- *Method Plugin:* contém os conceitos para definir e gerenciar os métodos configuráveis e repositórios de processos reutilizáveis.

Tendo em mente que organizações são diferentes, e ainda que dois projetos dentro da mesma organização também podem ser diferentes (BERGER, 2003, CUGOLA e GHEZZI, 1998), 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).

Um balanceamento é, portanto, necessário entre as necessidades individuais de flexibilidade e a necessidade organizacional de padronização e consistência (HUMPHREY, 1989). Alguns dos fatores a serem considerados são, segundo Humphrey (1989):

- Uma vez que projetos de software possuem diferenças, seus processos de software também as possuem;
- Na falta de um processo de software universal, organizações e projetos devem definir processos que atendam suas próprias necessidades específicas; e
- O processo utilizado para um dado projeto deve considerar o nível de experiência dos membros da equipe, a situação corrente dos produtos e as ferramentas e infraestrutura disponível.

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). A 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).

2.3.1 – Reutilização de Processos de Software

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). A reutilização de processos de software inclui não apenas a reutilização de definições de processos, mas também a reutilização de informações relacionadas à utilização dos processos (conhecimentos e experiências adquiridas) (RU-ZHI *et al.*, 2005). 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). Dessa forma, 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).

Seguindo um breve histórico, a reutilização de processos era, inicialmente, tratada através do armazenamento de componentes de processos que poderiam ser recuperados para posterior uso por meio 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.

Diante da analogia existente entre produto de software e processo de software (OSTERWEIL, 1987), analogias também estão sendo aplicadas no campo da reutilização. KELLNER (1996) destaca que o conhecimento de técnicas de reutilização de produtos de software, tais como: arquiteturas povoadas com componentes reusáveis; gerenciamento de repositórios para armazenar, catalogar, procurar, acessar, etc. ativos reusáveis; e gerência de configuração de ativos reusáveis poderiam ser aplicados a processos de software.

Assim, abordagens que seguem a linha de padrões, arquiteturas e *templates* de processo, em que processos específicos são definidos a partir de processos genéricos, e a componentização de processos, pregando a definição de processos a partir da composição de componentes de processos, são algumas das técnicas existentes para reutilizar 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 padrão de processo pode variar de simples informação adicional sobre como usar um dado componente, até uma estrutura de componentes relacionados para formar, por exemplo, uma fase do processo (BARRETO, 2007).

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 existirem mais de um componente com o mesmo propósito.

2.4 – LINHA DE PROCESSOS DE SOFTWARE E SUA MODELAGEM DE VARIABILIDADES

A abordagem de Linha de Processos de Software surgiu como uma técnica sistemática de reutilização de processos e foi apresentada em abordagens (BARRETO, 2007, JAUFMAN e MÜNCH, 2005, ROMBACH, 2006, WASHIZAKI, 2006) que aplicam a ideia de LPS em processos.

WASHIZAKI (2006) 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).

Uma das questões essenciais que deve ser considerada quando se constrói artefatos reutilizáveis é tratar as variabilidades presentes em uma Linha de Processos de Software, assim como temos a atividade de tratamento de variabilidades fortemente presente em LPS. A modelagem de variabilidades pode estar focada na parte de análise, utilizando-se de modelos de características e seus sucessores, ou focada na fase de projeto e/ou implementação. A notação para a modelagem de variabilidades pode ser gráfica, textual ou um misto das duas formas. No entanto, em uma notação gráfica, os pontos de variação são reconhecidos muito mais facilmente (MASSEN e LICHTER, 2002). Os autores de abordagens de LPS propõem uma lista genérica de requisitos desejáveis a uma notação que tenha por objetivo expressar variabilidade em um artefato de software. Essa lista foi refinada por OLIVEIRA (2006) como uma compilação dos conceitos apresentados nos diversos trabalhos na literatura. No presente trabalho, essa lista foi revisitada e adaptada para tratar variabilidade dentro da área de reutilização de processos:

- ***Representação gráfica de elementos não configuráveis (invariantes):*** uma notação deve ser clara ao representar as partes fixas, não-configuráveis, no domínio de processos de software modelado;

- ***Representação gráfica da opcionalidade e/ou obrigatoriedade de elementos não configuráveis (invariantes)***: uma notação deve ser clara ao representar as partes comuns a todos os diferentes processos de uma família de processos de software, assim como explicitar as partes que podem ser descartadas na instanciação de um processo específico de um projeto. Ou seja, a notação deve deixar claro quais elementos serão, obrigatoriamente, selecionados na instanciação de um processo específico de projeto, e quais são os elementos a serem selecionados apenas de acordo com as características envolvidas no contexto de instanciação;

- ***Representação gráfica de Pontos de Variação***: é desejável a documentação em artefatos dos pontos de configuração dos diferentes processos de software pertencentes à linha de processos, assim como sua fácil visualização em um modelo que reflita claramente essas variabilidades;

- ***Representação gráfica da opcionalidade e/ou obrigatoriedade de Pontos de Variação***: é desejável a fácil identificação da obrigatoriedade de configuração em um determinado ponto da linha de processos de software, assim como a possibilidade de a configuração ser dispensável;

- ***Representação gráfica de elementos Variantes***: as variantes e sua ligação com os seus respectivos pontos de variação devem ser representadas de forma a explicitar as alternativas pertencentes a um ponto de configuração da linha de processos;

- ***Representação gráfica da opcionalidade e/ou obrigatoriedade de elementos Variantes:*** é desejável a fácil identificação da obrigatoriedade de participação de uma alternativa de configuração de um ponto de variação respectivo, assim como a possibilidade de sua participação ser opcional, definida de acordo com o contexto de instanciação;

- ***Representação de relações de Dependência/Exclusividade entre elementos***: relações de dependência ou de mútua exclusividade devem ser explicitamente representadas por uma notação, para quaisquer conjuntos de elementos, independentemente de suas posições no diagrama. Essas relações permitem a definição de conjuntos de restrições que visam garantir a consistência das instâncias de processos de software a serem geradas a partir da linha de processos representada; e

- ***Representação de Relacionamentos entre elementos***: é desejável que uma notação deixe explícita a relação entre elementos de um modelo. A notação deve viabilizar a representação de relacionamentos relevantes e que tenham semântica para a instanciação de elementos na derivação de processos específicos de projeto, a partir da linha de processos de software representada.

A análise aqui proposta foi conduzida em busca de pontos ainda a serem explorados em uma abordagem sistemática de construção, uso e gerenciamento de uma Linha de Processos de Software. Desta forma, dada a importância já destacada da modelagem de variabilidades, o principal ponto analisado foi a questão dessa

modelagem nas diversas abordagens estudadas, conforme discutido a seguir.

2.4.1 – Abordagens de Linha de Processos de Software

SUTTON e OSTERWEIL (1996) discutem o conceito de família de processos baseado na correlação existente entre produtos e os processos pelos quais estes são desenvolvidos. 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. Assim, a noção de uma família de processos está implícita na noção de uma família de produtos. A visão de família enfatiza tanto as similaridades como as diferenças entre os membros que a constituem, e chama atenção para os relacionamentos entre eles. A diferenciação entre membros de uma família de processos pode ser realizada baseada tanto em um conjunto de critérios relacionados a produtos, como tamanho, complexidade, expectativas de evolução, manutenção e reutilização do produto; como em outra variedade de critérios, como a natureza da organização ou do projeto de desenvolvimento. 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, principalmente relacionada à caracterização, utilização e gerenciamento das similaridades e diferenças entre as famílias de processos e os membros dessas famílias.

ROMBACH (2006) trabalha o conceito de Linha de Processos de Software, baseado nos mesmos princípios de LPS. Define como características principais de uma Engenharia de Linha de Processos de Software: (i) Dois processos de desenvolvimento separados: o processo de ED, 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 EA, 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.

JAUFMAN e MÜNCH (2005) propõem um método que usa uma Linha de Processo específica de domínio para adaptar e refinar o processo adaptado, baseado nas atividades de processo executadas em uma primeira iteração. O método consiste em dois passos principais: (1) uma linha de processos específica de um domínio é usada para a adaptação *top-down* realizada no início do projeto, com o propósito de fornecer conhecimento de 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.

WASHIZAKI (2006) 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 Engenharia de Linha de Produtos, a Engenharia de Linha de Processos é dividida em duas fases: (i) ED, que corresponde a 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. O autor também especifica que uma abordagem *top-down* para construção da linha pode gerar perdas e, por isso, propõe uma abordagem *bottom-up* para estabelecer linhas e arquiteturas de linhas de processos a partir de processos existentes. 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. Um diagrama de *features* pode ser utilizado para definir os requisitos da linha de processos; (4) definição de

processos específicos de projetos consistentes. A técnica apresentada inclui algumas extensões ao SPEM v1.1 para expressar claramente as similaridades e variabilidades em fluxos de processos, expressos através de diagramas de atividades. A Figura 2.5 apresenta um exemplo de representação de uma linha de processos derivada para processos de projeto de hardware/software de sistemas embarcados extraída de (WASHIZAKI, 2006).

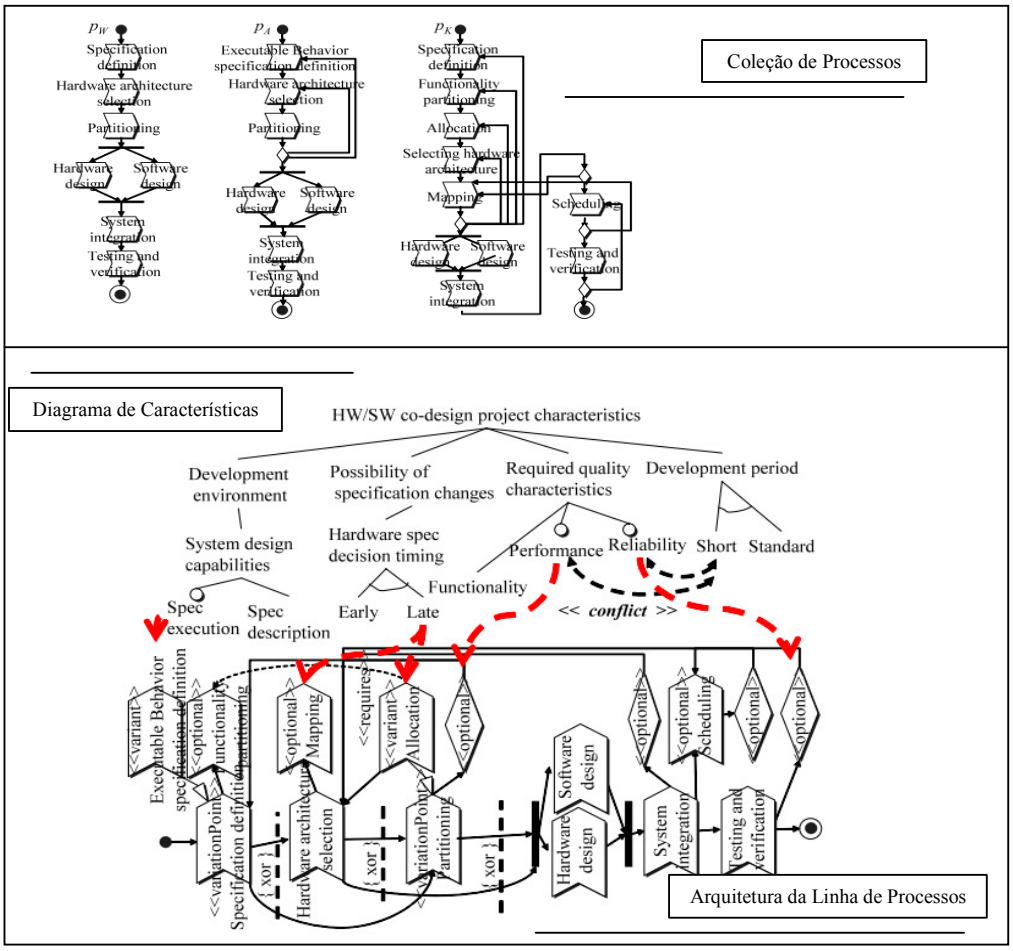


Figura 2.5 - Arquitetura da Linha de Processos e o diagrama de características associado (Adaptado de: WASHIZAKI, 2006)

As variabilidades são representadas por pontos de variação e variantes. Pontos de variação são atividades, artefatos e papéis que podem ser modificados de acordo com características de um projeto específico e são identificados através do estereótipo <<*variationPoint*>>. Variantes são candidatos concretos que podem ser aplicados aos pontos de variação, identificados através do estereótipo <<*variant*>>. As variantes, por serem consideradas como opções a serem atribuídas aos pontos de variação correspondentes, identificados dentro do processo mandatório são, por natureza, sempre opcionais. Elementos opcionais são identificados através do estereótipo <<*optional*>>.

O estabelecimento de relações é limitado às ligações que apresentam o fluxo de atividades; relacionamentos de generalização/especialização entre pontos de variação e variantes; relacionamentos de dependência, representadas pelo estereótipo <<*requires*>>; e relacionamentos de seleção exclusiva, representados pelo estereótipo <<*xor*>>. O último relacionamento apresenta uma aplicação limitada a um conjunto de alternativas de um fluxo de execução de atividades e não é aplicado de forma abrangente a diferentes partes do modelo. Nenhum suporte ferramental é apresentado para a modelagem da linha e verificação da consistência entre os diagramas.

ARMBRUST *et al*. (2009) também trabalham a transferência dos conceitos de LPS para processos de software e definem uma Linha de Processos de Software 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. O suporte a Engenharia de Linha de Processos de Software proposto inclui as atividades de *Definição de Escopo*, que determina os membros da linha de processos; *Engenharia de Domínio de Processos*, que constrói um repositório de processos, contendo todas as partes variáveis e estáveis de processos, assim como um modelo de decisão que governa quando usar determinada variante; e *Instanciação da Linha de Processos*, que extrai do repositório uma instância de processo sem variabilidade para cada projeto, com possibilidade de adaptação durante a customização (ARMBRUST *et al*., 2008). Detalhando a parte de construção da linha, os autores definem três etapas: (1) *Definição do Escopo da Linha de Processos*, estabelecido pela análise dos produtos e projetos da organização para extração das necessidades de processos da organização, ou seja, identificação do conjunto de características que os processos dentro da linha devem envolver; (2) *Modelagem da Linha de Processos*, que corresponde à análise dos processos da linha em termos de similaridades e variabilidades. Baseada nesta análise, uma coleção de artefatos de processos genéricos (blocos de construção) são construídos 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; e (3) *Definição da Arquitetura de Linha de Processos*, que cria uma arquitetura de processos de referência contendo todos os artefatos comuns aos processos e todas as variabilidades, que associados a um modelo de decisão, permitem a derivação de modelos de processos individuais durante o uso da linha construída (ARMBRUST *et al*., 2009).

As variações são modeladas usando a ferramenta gráfica de modelagem de processos de software SPEARMINT™ (ARMBRUST *et al*., 2009). A Figura 2.6 apresenta um exemplo extraído de ARMBRUST *et al*. (2008) de uma modelagem parcial de uma linha de processos construída para o desenvolvimento de software da Agência de Exploração Espacial Japonesa (JAXA).

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. O tratamento de variabilidades nos diferentes elementos de um processo não é especificado.

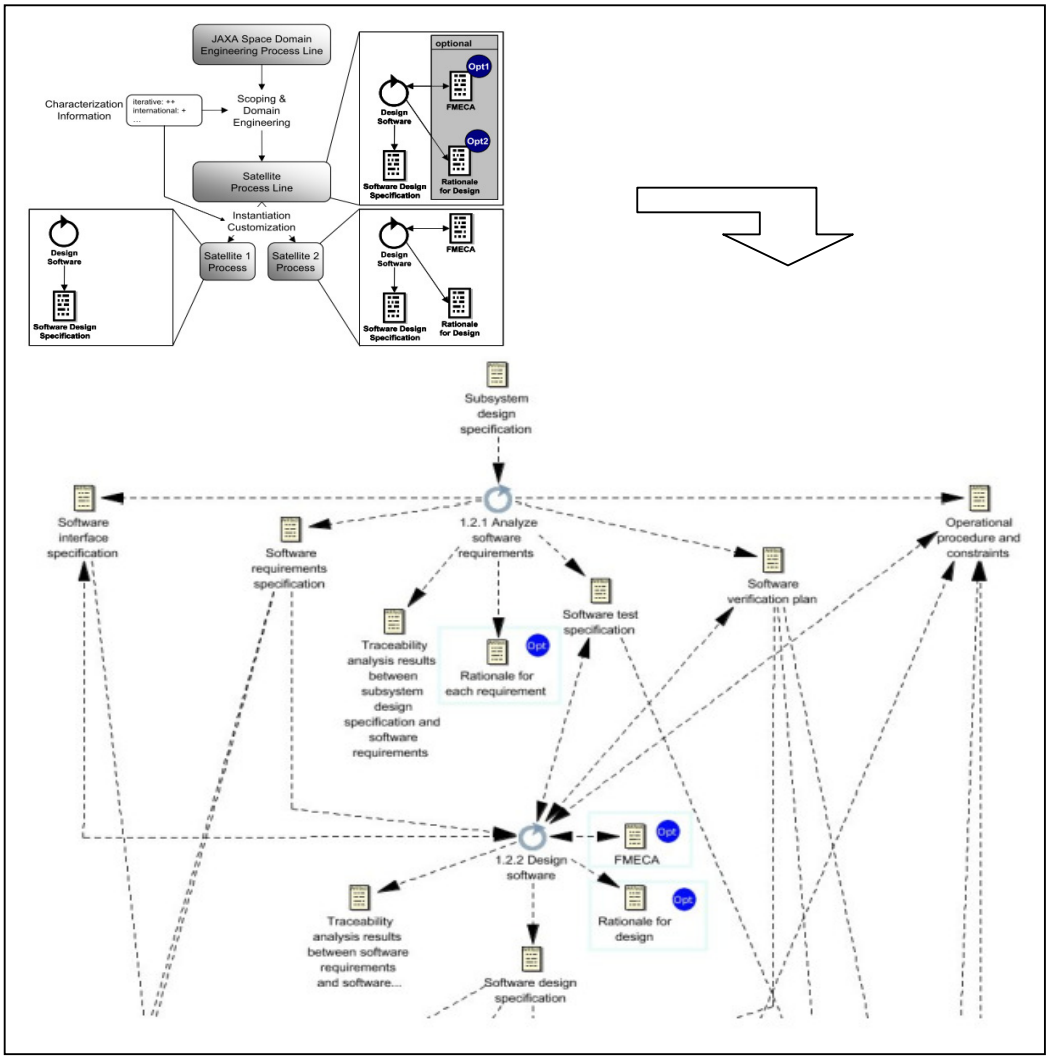


Figura 2.6 - Exemplo parcial de uma Linha de Processos
(Adaptado de ARMBRUST *et al.,* 2008)

SIMIDCHIEVA *et al.* (2007) propõem aplicar a abordagem de família de produtos de software a processos com o intuito de gerenciar a variação de processos. Os autores argumentam que a totalidade das variantes de processos é uma família de processos. Para que vários processos sejam membros da mesma família, precisam ser suficientemente similares, por exemplo, possuir um núcleo comum que é idêntico ou ligeiramente diferente. Adicionalmente, é suposto que todas as instâncias de processos necessárias podem ser geradas a partir de um *framework*, talvez com a ajuda de algum tipo de especificação dos objetivos de processo requeridos, a partir da composição de componentes.

A abordagem proposta em SIMIDCHIEVA *et al.* (2007) usa a linguagem de definição de processos *Little-JIL* (WISE, 2006). A linguagem divide a definição de processos em três dimensões: (i) passos de processos individuais e suas coordenações; (ii) o comportamento dos agentes que executam os passos; e (iii) a estrutura da coleção de artefatos que são produzidos ou consumidos pelos passos. Um exemplo de modelagem de um diagrama de coordenação de passos, associados a seus respectivos agentes, pode ser visto na Figura 2.7.

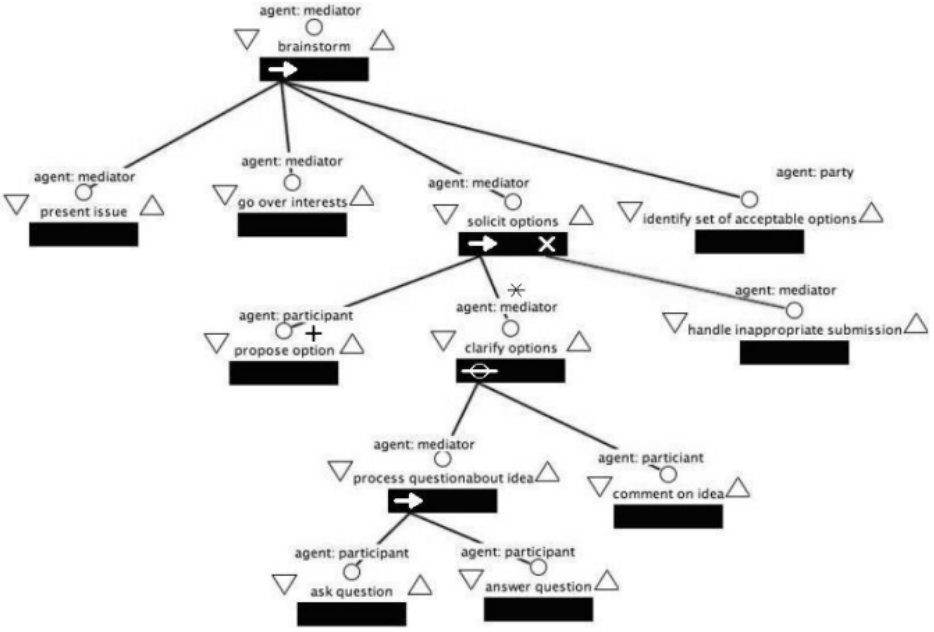


Figura 2.7 - Exemplo de modelagem de um diagrama de coordenação de passos de processos usando Little-JIL (SIMIDCHIEVA *et al.,* 2007)

O diagrama de coordenação especifica os passos envolvidos de acordo com uma estrutura hierárquica. Utiliza conceitos de sequência e iteratividade, à medida que define a ordem de execução e a possibilidade de repetição de instâncias dos passos

representados. Pontos de escolha de alternativas de execução de um passo são representados pela associação de filhos a um passo com o desenho de um círculo cortado associado. É possível notar uma baixa exploração semântica de relacionamentos.

BARRETO *et al*. (2009) propõem uma abordagem para guiar a definição de elementos reutilizáveis de processo de software a partir da adaptação de processos e ativos de processos existentes em um organização. A proposta é que a definição de processos seja realizada a partir de componentes de processos reutilizáveis. 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 analisado; (iii) ser versionada; e (iv) ter vários tipos de rastreabilidade estabelecido; entre outros. A seleção de um conjunto diferente desses três componentes gera um processo diferente, assim a combinação de passos mais elaborados, a seleção de comportamentos distintos de diferentes agentes e diferentes estruturas de artefatos implicam na geração de variações de processos que constituem a família.

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 trabalhos, e pode ser composta por componentes de processos, 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. Desta forma, a modelagem de variabilidades de uma Linha de Processos de Software 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. Um exemplo dessa representação pode ser visualizado na Figura 2.8.

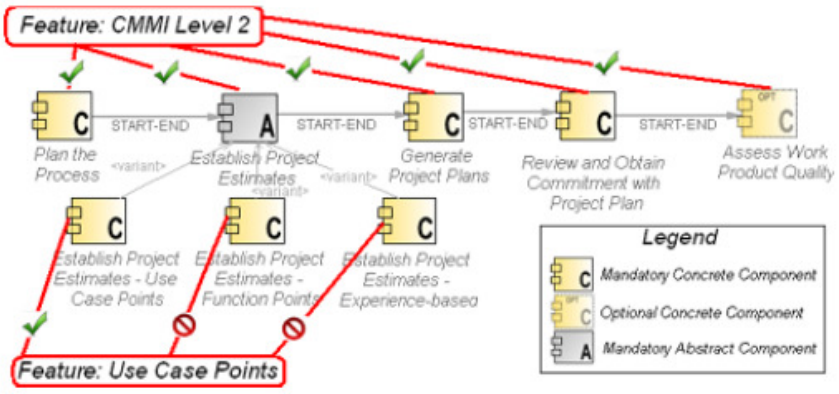


Figura 2.8 - Exemplo de Componentes Abstrato e suas Variantes (BARRETO *et al*., 2010)

O trabalho de MARTÍNEZ-RUIZ *et al*. (2008) estende a modelagem de variabilidades proposta pelo SPEM v2.0 para apoiar a modelagem de Linha de Processos de Software. O SPEM v2.0 trabalha o conceito de variabilidades através de mecanismos que viabilizam a customização dos elementos sem necessariamente modificar suas estruturas originais. Permite a descrição separada das diferenças (adições, mudanças, omissões) em relação ao elemento original. As instâncias de processos a serem geradas podem ser projetadas com a combinação dinâmica e aplicação sob demanda, usando a interpretação das regras definidas pelas variabilidades. O tipo de variabilidade entre duas instâncias de elementos é definido pelos seguintes tipos de relacionamentos:

- 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;
- Substituição (*replaces*): indica que o elemento base é substituído por uma variante do elemento com a definição de outros valores de atributos e associações;
- 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
- 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.

Algumas limitações foram apontadas para a modelagem de variabilidade em Linhas de Processos de Software através do SPEM v2.0 (MARTÍNEZ-RUIZ *et al.*, 2008). Entre estas limitações destacam-se 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. Ainda podemos ressaltar a falta de uma notação específica para representar as variabilidades, que são especificadas por estereótipos acrescentados a relacionamentos de associação ou dependência UML.

Desta forma, o trabalho de MARTÍNEZ-RUIZ *et al.* (2008) propõe extensões que permitem definir pontos de variação e variantes para atividades, tarefas, produtos de trabalho e papéis no nível de modelagem de processos e não de conteúdo de métodos proposto pelo SPEM v2.0. Relações de dependência e restrições também podem ser estabelecidas pela extensão proposta, mas não são graficamente representadas. No entanto, a representação gráfica proposta não trabalha a opcionalidade nem explora diferentes relações semânticas entre elementos. Um exemplo pode ser visto na Figura 2.9, que mostra questões de variabilidade em processos de desenvolvimento de software.

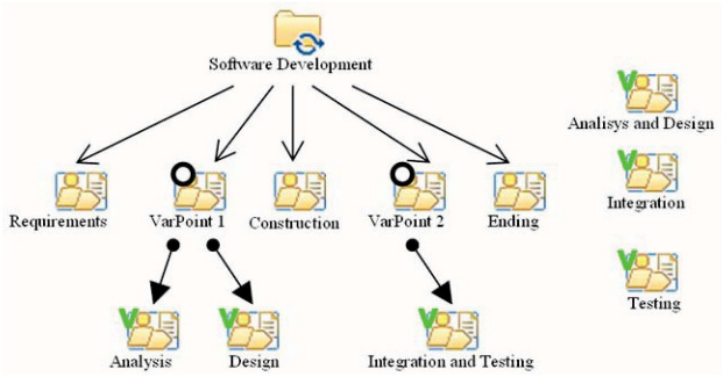


Figura 2.9 - Exemplo de modelagem de variabilidades para processos de desenvolvimento de software (MARTÍNEZ-RUIZ *et al.,* 2008)

Uma abordagem 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, foi proposta por ALEIXO *et al.* (2010). O objetivo central da abordagem é promover o reúso sistemático de componentes de processos de software através da organização de processos fortemente relacionados em torno do conceito de uma linha de processos. A abordagem parte da modelagem e definição de uma linha de processo de software (1), através de

ferramentas existentes de especificação de processos (2). Em seguida, concentra-se na gerência automatizada de variabilidades dos elementos da linha de processo sendo modelada (3), através de especificação dos modelos de variabilidades (modelo de características, modelo de configuração e modelo de processo) (4). Na etapa seguinte, com o auxílio de uma ferramenta de derivação, são selecionadas as características relevantes para um processo (5), possibilitando a geração automática de processos de software customizados que lidem com cenários e necessidades específicas de um dado projeto (6).

A Figura 2.10 apresenta um exemplo de modelagem de variabilidades proposta pela abordagem. A modelagem de características apresenta-se através de uma estrutura de árvore usando a notação proposta por CZARNECKI e EISENECKER (2000), o que limita a visibilidade gráfica. Os diferentes tipos de elementos de processos não apresentam diferenciação gráfica. A diversidade semântica de tipos de relacionamentos é limitada, assim como a não adoção de restrições gráficas de dependência e mútua exclusividade. Além disso, o trabalho aponta a necessidade de extensão do modelo de características em múltiplos níveis, além da implementação da gerência de variabilidades de forma gráfica, através de um modelo de arquitetura de processos (ALEIXO *et al*., 2010).

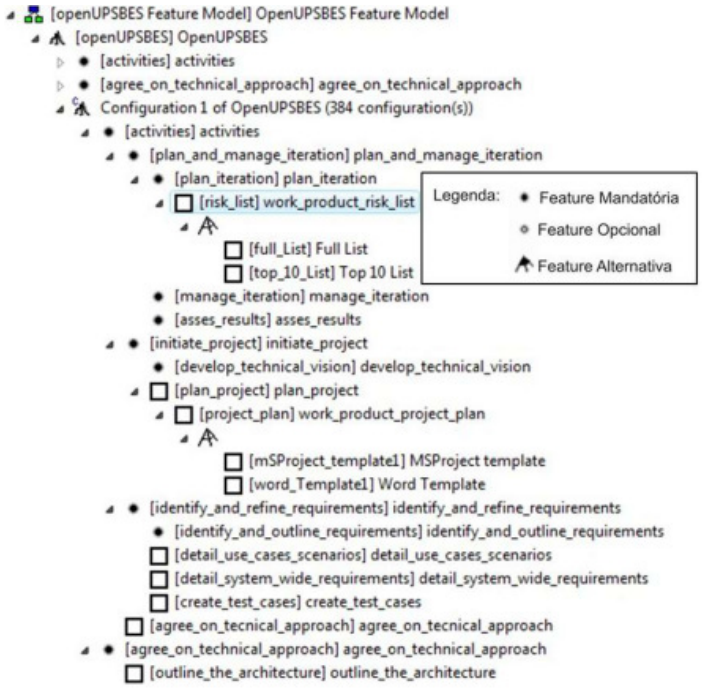


Figura 2.10 - Exemplo parcial de um modelo de características representando variabilidades do processo OpenUP (ALEIXO *et al.,* 2010)

2.4.2 – Considerações sobre as Abordagens e suas Modelagens de Variabilidades

A análise destes trabalhos permitiu observar que o conceito de Linha de Processos é recente e que ainda há poucas publicações sobre o assunto. Em particular, a fase de AD ainda não foi devidamente explorada, pois a maioria das propostas existentes se concentra na arquitetura de componentes de processos. Em geral, os modelos propostos tratam da modelagem no nível de projeto e não discutem a representação em um nível de abstração correspondente ao conceito de característica, fortemente presente em LPS. Essa modelagem é defendida como uma abordagem que trata a complexidade em expressar requisitos em forma de características e da sua estruturação hierárquica em um diagrama de características (MASSEN e LICHTER, 2004).

As representações apresentadas previamente, muitas vezes são voltadas apenas para a modelagem de processos e não possuem por objetivo representar variabilidades inerentes a uma família de processos de software. Apesar dessas limitações, essas representações, já que utilizadas para modelagem em trabalhos que tratam a abordagem de Linha de Processos, foram analisadas segundo o conjunto de requisitos para uma notação gráfica que visa à representação de variabilidades no domínio de processos de software apontados na Seção 2.4. Portanto, as seguintes considerações foram levantadas:

- Notação utilizada por WASHIZAKI (2006): Seu ponto mais forte é a representação explícita do conceito de variabilidade. No entanto, apesar da classificação de opcionalidade ser atribuída a elementos considerados invariantes, não há classificação desse tipo para pontos de variação e variantes. Além disso, o estabelecimento de relações de dependência é realizado entre elementos opcionais e variantes e é expresso apenas entre dois elementos do diagrama. O relacionamento de mútua exclusividade é utilizado para marcar casos em que a seleção de apenas um dos fluxos de transição do processo deve ser realizada quando há alternativas de variabilidade e opcionalidade. Por último, os relacionamentos expressos são limitados a ligações de fluxos de processo, relações de generalização (utilizadas para denotar relações de variabilidade) e marcações de dependência e seleção exclusiva.
- Notação utilizada por ARMBRUST *et al*. (2008, 2009): A modelagem é feita

utilizando uma ferramenta gráfica para modelagem de processos de software, não especificamente para tratamento de variabilidades. A representação de variabilidades é descrita de forma concomitante com questões de opcionalidades. O tratamento desses conceitos é feito através da definição de regras que não estão expressas no modelo. Assim, a representação de variabilidades não está declarada de forma explícita, o que dificulta a interpretação dos pontos de configuração e suas alternativas. Relacionamentos de dependência e mútua exclusividade não são trabalhados, assim como não há especificação declarada de tipos de relacionamentos com maior semântica para o modelo.

- Notação utilizada por SIMIDCHIEVA *et al.* (2007): A modelagem usa uma linguagem de definição de processos. Representa, de forma explícita, variabilidades através da especificação de um símbolo associado ao ponto de variação. A representação de opcionalidade é feita através da aplicação de anotações que especificam o número de execuções possíveis do passo associado (um ou mais "+"/ zero ou mais "*"). A especificação de dependência é parcialmente representada através da sequência de execução e estrutura hierárquica. No entanto, relacionamentos específicos de dependência e mútua exclusividade não foram especificados. Além disso, a semântica dos relacionamentos existentes é limitada e representa fluxos de execução.
- Notação utilizada por BARRETO *et al.* (2010): Apresenta representação explícita dos conceitos de variabilidades e opcionalidades. No entanto, a modelagem proposta trata de componentes de processo, não envolvendo a análise, em termos de opcionalidades e variabilidades, de outros elementos de processos como papéis e artefatos. A semântica dos relacionamentos é também limitada pela atribuição de fluxo de execução dos componentes e estabelecimento de relações entre pontos de variação e variantes. Relações de dependência e mútua exclusividade também não são claramente exploradas, apenas trabalhadas pela ordem de execução dos elementos.
- Notação utilizada por MARTÍNEZ-RUIZ *et al.* (2008): Representa, de forma explícita, classificações de variabilidade. Incluiu definições de relações de dependência e mútua exclusividade que ficaram restritas a elementos pertencentes a relacionamentos de variabilidade. As classificações de

opcionalidades apesar de atribuídas a diversos elementos não possui representação gráfica explícita, a não ser quando atribuídas a relacionamentos.

- Notação utilizada por ALEIXO *et al.* (2010): Apresenta uma representação explícita do conceito de opcionalidade e variabilidade. No entanto, as classificações de opcionalidades não foram especificadas para pontos de variação e variantes. Outro ponto fraco é a não-representação de dependências ou mútua exclusividade no diagrama, o que reduz a expressividade do modelo. Relacionamentos entre características também são parcialmente considerados através de uma estrutura de árvore.

A análise encontra-se sumarizada na Tabela 2.3. Desta forma, é possível perceber com mais clareza as contribuições e limitações de cada representação em relação à lista de requisitos levantados.

Um dos principais problemas encontrados foi a falta de representação explícita dos pontos de configuração denominados pontos de variação. Como consequência, os relacionamentos entre pontos de variação e variantes, por muitas vezes não são explorados nem devidamente representados nos modelos propostos. O estabelecimento de relações de dependência e mútua exclusividade também encontra limitações, pois as representações propostas apenas satisfazem parcialmente aos requisitos, seja por não permitir tal representação no próprio modelo de características, seja por não permitir que tais dependências envolvam mais de duas características, reduzindo a expressividade do modelo.

Ainda é importante destacar que nas abordagens existentes a variabilidade é representada em apenas um nível de abstração seja uma modelagem de características ou uma modelagem de componentes. No entanto, devemos considerar que a construção da Linha de Processos de Software agrega diferentes fases e elementos correspondentes a diferentes níveis de abstração. Desta forma, é importante que a variabilidade seja representada e propagada pelos diferentes modelos que constituem a linha.

Tabela 2.3 - Tabela comparativa entre as representações analisadas (ver legenda[1])

Representação / Requisito	WASHIZAKI (2006)	ARMBRUST *et al.*(2009)	SIMIDCHIEVA *et al.* (2007)	BARRETO *et al.* (2010)	MARTÍNEZ-RUIZ *et al.*(2008)	ALEIXO *et al.*(2010)
Representação gráfica de elemento invariante	✓	✓	✓	✓	✓	✓
Representação gráfica de opcionalidade de elemento invariante	✓	✓	✓	✓	✗	✓
Representação gráfica de ponto de variação	✓	⚠	✓	✓	✓	✓
Representação gráfica de opcionalidade de ponto de variação	✗	✗	✓	✓	✗	✗
Representação gráfica de elemento variante	✓	⚠	✓	✓	✓	✓
Representação gráfica de opcionalidade de elemento variante	✗	✗	✓	✓	✗	✗
Representação de relações de dependência e mútua exclusividade	⚠	✗	⚠	⚠	⚠	✗
Representação de relacionamentos	⚠	⚠	⚠	⚠	⚠	⚠

2.5 – CONSIDERAÇÕES FINAIS

Com o aumento da exigência por parte dos clientes, as organizações de desenvolvimento de software vêm buscando cada vez mais oferecer serviços e produtos com qualidade (MAGDALENO, 2010). Pesquisas (CUGOLA e GHEZZI, 1998, FEI DAI e TONG LI, 2007, FUGGETTA, 2000, OSTERWEIL, 1987, PRESSMAN, 2001, RAMAN, 2000) apontam que a qualidade do produto de software depende do processo de desenvolvimento adotado para construí-lo e que muitos dos problemas associados ao desenvolvimento de produtos de software podem ser resolvidos aperfeiçoando-se este processo.

[1] Legenda: ✓ Sim ✗ Não ⚠ Parcialmente

O tema reutilização de processos de software 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). Dessa forma, as organizações de desenvolvimento de software procuram obter expressivas economias, além de permitir um efetivo aumento na qualidade do software produzido (COSTA *et al.*, 2007). A pesquisa de HOLLENBACH e FRAKES (1996) mostra que é possível diminuir em pelo menos dez vezes o tempo e o esforço necessário para criar a descrição de processo de um projeto quando se instancia um processo reutilizável ao invés de criá-lo desde o início.

Uma das técnicas de reutilização de processos surgiu com base nos conceitos provenientes da abordagem de LPS é denominada de Linha de Processos de Software. Esta área de pesquisa, ainda recente, possui conceitos incipientes e abordagens com limitações. Em particular, a fase de análise de domínio com representação em um alto nível de abstração ainda não foi devidamente explorada, pois a maioria das propostas existentes se concentra no nível de projeto através de arquiteturas de componentes de processos de software. Além disso, o tópico representação de variabilidades é tratado como secundário em diversas abordagens, que apresentam de forma superficial alguma modelagem adotada, sem a devida adaptação para a área de Linha de Processos de Software.

Diante dessas questões, a proposta de uma abordagem sistemática de Engenharia de Linha de Processos de Software é apresentada no próximo capítulo. A abordagem consiste na especificação das etapas que constituem o ciclo de desenvolvimento e aplicação de uma Linha de Processos de Software e seus respectivos produtos. A Engenharia de Linha Processos de Software é constituída de duas grandes fases: Engenharia de Domínio de Processos de Software (EDPS) e Engenharia de Aplicação de Processos de Software (EAPS). Além disso, o foco deste trabalho está na fase de AD e na apresentação de uma notação de modelagem de variabilidades através de um modelo de característica que contemple os conceitos envolvidos no domínio de processos de software. Tal notação tem sua semântica formalizada por meio de um meta-modelo que tem por objetivo servir de referência para a construção desses modelos. A proposta visa aumentar o potencial de reutilização de uma Linha de Processos de Software.

Capítulo III – *OdysseyProcess-FEX*
UM META-MODELO E UMA NOTAÇÃO PARA MODELAGEM DE VARIABILIDADES DE LINHA DE PROCESSOS DE SOFTWARE

3.1 - INTRODUÇÃO

Diante da diversidade existente no universo de projetos de software, a tarefa de definição de processos de software torna-se não trivial. Esta tarefa exige experiência, envolve o conhecimento de muitos aspectos da Engenharia de Software e requer a harmonização de muitos fatores do contexto da equipe, do projeto ou da organização (BARRETO, 2007, MAGDALENO, 2010).

Com o propósito de facilitar a definição de processos, diminuindo o custo e o esforço associado à atividade, além de possivelmente aumentar a qualidade dos processos gerados (BARRETO, 2007), técnicas de reutilização de software têm sido aplicadas no contexto de processos de software, conforme visto no capítulo anterior. Desta forma, a abordagem de Linha de Processos surgiu como uma técnica sistemática de reutilização de processos e foi apresentada em abordagens (BARRETO, 2007, JAUFMAN E MÜNCH, 2005, ROMBACH, 2006, WASHIZAKI, 2006) que aplicam a ideia de Linhas de Produtos de Software (LPS) em processos.

Neste trabalho, definimos uma Linha de Processos de Software 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).

Uma abordagem de Engenharia de Linha de Processos de Software (ELPS) compreende toda a estrutura para construir, utilizar e gerir uma Linha de Processos de Software. Esta abordagem é composta por quatro elementos principais: um método, um meta-modelo, uma notação e um ferramental de apoio (Figura 3.1). O método consiste na especificação das etapas que constituem o ciclo de desenvolvimento e aplicação de uma Linha de Processos de Software. O meta-modelo define uma sintaxe abstrata para a construção de modelos, isto é, uma estrutura que norteia a representação de elementos em um modelo (MATULA, 2003) *apud* (OLIVEIRA, 2006). Um meta-modelo serve de referência para a construção de modelos, através do fornecimento da estruturação de representação de conceitos e das relações semânticas entre eles. A notação representa simbolicamente os conceitos formalizados pelo meta-modelo. Por último, o ferramental

de apoio consiste no provimento de um ambiente para dar suporte às diferentes atividades a serem executadas, definidas pelo método de ELPS.

Figura 3.1 - Elementos da abordagem de Engenharia de Linha de Processos de Software

Neste trabalho, a etapa de Análise do Domínio de Processos de Software foi desenvolvida com a especificação de suas atividades e produtos gerados. Um meta-modelo denominado *OdysseyProcess-FEX* foi desenvolvido, visando explicitar a representação de conceitos de elementos de processos com a representação de variabilidades do domínio. Junto ao meta-modelo proposto é apresentada a notação *OdysseyProcess-FEX.* Como ferramental de apoio, este trabalho propõe a adaptação do ambiente Odyssey (ODYSSEY, 2011). O trabalho de adaptação de técnicas e ferramentas disponíveis no ambiente de apoio à reutilização de software para atender as necessidades de suporte à criação da Linha de Processos de Software encontra-se descrito no Capítulo 4.

Este capítulo está organizado na seguinte forma: a Seção 3.2 descreve, em linhas gerais, a ELPS, com destaque para a etapa de Análise de Domínio de Processos de Software, na Seção 3.2.1. A Seção 3.3 introduz a descrição do meta-modelo e notação propostos. A Seção 3.3.1 apresenta uma descrição do domínio de Engenharia de Requisitos, que será utilizado como exemplo ao longo do capítulo. O meta-modelo é definido na Seção 3.3.2, com detalhamento de cada conceito adotado. A notação *OdysseyProcess-FEX,* cuja simbologia ilustra os conceitos definidos, é apresentada na Seção 3.3.3. A Seção 3.3.4 apresenta um exemplo de utilização da notação *OdysseyProcess-FEX.* Por fim, as considerações finais são discutidas na Seção 3.4.

3.2 - ENGENHARIA DE LINHA DE PROCESSOS DE SOFTWARE (ELPS)

A Engenharia de Linha de Processos de Software (ELPS) é constituída de duas grandes fases: a Engenharia de Domínio de Processos de Software (EDPS) e Engenharia de Aplicação de Processos de Software (EAPS) (Figura 3.2).

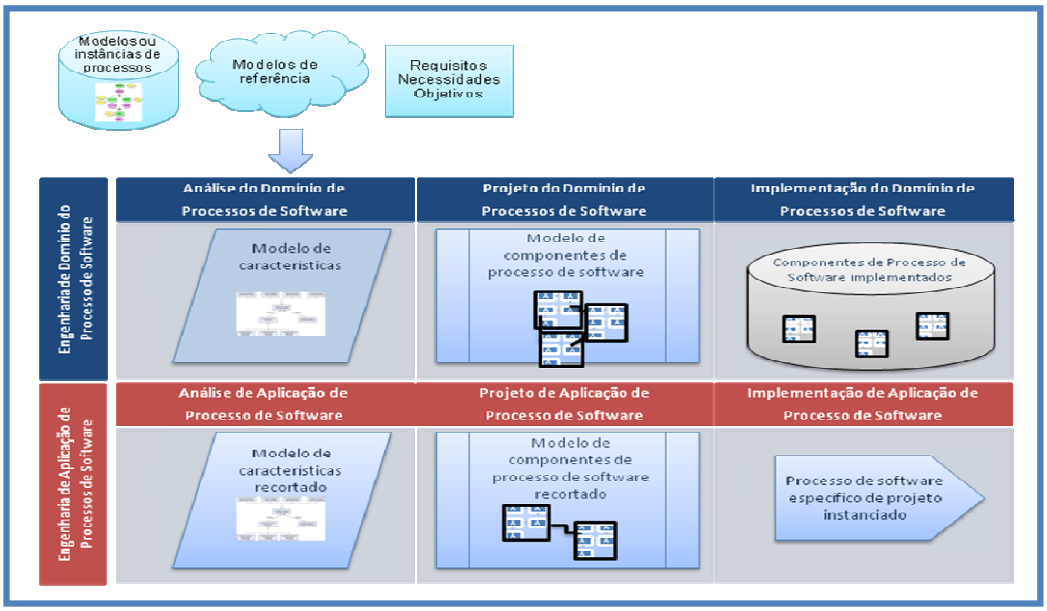


Figura 3.2 - Abordagem de Engenharia de Linha de Processos de Software

A EDPS é composta por três etapas: *Análise, Projeto* e *Implementação do Domínio de Processos de Software.* A etapa de ***Análise do Domínio de Processos de Software*** se preocupa com a identificação e representação do conhecimento do domínio de processos de software, através de uma análise de suas similaridades e diferenças. Como foco deste trabalho, esta etapa encontra-se detalhada na Seção 3.2.1.

Na etapa de ***Projeto do Domínio de Processos de Software*** são detalhados aspectos do domínio, através do refinamento dos modelos de análise na construção de uma Arquitetura do Domínio de Processos de Software. Informações sobre especificação de procedimentos e técnicas, sequencialidade, iteratividade e definição de modelo de ciclo de vida são alguns dos refinamentos que podem ser acrescentados nesta etapa. A arquitetura definida pode ser entendida como uma coleção de ativos do domínio, componentes de processos, estruturados de forma padronizada para promover a reutilização dentro de contextos específicos de aplicação. Componentes de processo podem ser especificados como elementos de processos definidos como grupo de atividades e/ou outros elementos de processo relacionados por dependências lógicas, que quando executados provêem valor ao projeto (BHUTA *et al.*, 2005). Essa

arquitetura admite a definição de variabilidades através de componentes não vinculados a uma única forma de realização, desde que atendam a um conjunto de restrições estabelecidas. A forma pela qual os componentes se relacionam também é descrita, através da especificação de interfaces que designam os artefatos que são requeridos, produzidos e trocados entre eles. Formas de customização disponíveis para os componentes podem ser expressas, assim como a distribuição destes pelos repositórios. Além disso, nesta etapa, é definida a rastreabilidade com o nível de abstração de análise, relacionando os elementos do modelo de componentes gerados com os elementos correspondentes do modelo de características.

A especificação do projeto arquitetural, em conjunto com a semântica dada pelo modelo de domínio, é a base para a etapa de ***Implementação do Domínio de Processos de Software***. O resultado desta fase da EDPS corresponde à implementação de fato da Linha de Processos de Software, através da especificação dos componentes arquiteturais em uma representação executável por uma máquina de processos (*workflow*). A máquina de processos Charon (MURTA, 2002), consiste em um exemplo de ferramenta que viabiliza a modelagem, instanciação, simulação, execução, acompanhamento e evolução de processos de software.

Os componentes gerados devem ser armazenados em uma base para que possam ser resgatados através de uma infraestrutura de reutilização que permita a descrição, catalogação, busca e recuperação de componentes de processos implementados. Esse processo de disponibilização de componentes de processos reutilizáveis deve garantir que esses sejam continuamente evoluídos. Desta forma, a EDPS deve prever a possibilidade de realizar revisões entre suas etapas. A abordagem deve seguir ciclos incrementais e evolutivos em que artefatos vão sendo evoluídos com base em novas informações de domínio adquiridas (Figura 3.3).

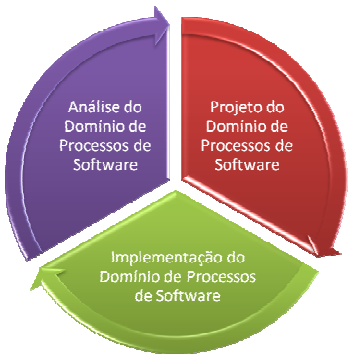


Figura 3.3 - Ciclo de etapas da Engenharia de Domínio de Processos de Software

A abordagem de EAPS consiste em um recorte do domínio para gerar um processo específico para um projeto. A instanciação de um processo se dá através da reutilização dos artefatos gerados na EDPS. A abordagem proposta prevê as seguintes etapas da EAPS: (1) ***Análise da Aplicação de Processo de Software*** (define os requisitos e particularidades do processo a ser instanciado) em que um conjunto de características é selecionado, consistindo em um primeiro nível de recorte ou derivação, baseado na seleção em um alto nível de abstração, que representa elementos de processos em um modelo de características do domínio de processos de software. A verificação das inconsistências geradas pelo recorte e das necessidades de adaptação a particularidades do processo devem ser avaliadas e conduzidas nesta etapa; (2) ***Projeto da Aplicação de Processo de Software*** (soluções arquiteturais), onde a arquitetura do processo é determinada, através de um segundo recorte, que seleciona o conjunto de componentes que deve ser instanciado no processo específico. Desta forma, decisões de variabilidades em nível de projeto devem ser conduzidas e acrescentadas aos componentes mandatórios especificados pelo processo padrão. A adaptação dos componentes selecionados a particularidades não previstas deve ser realizada ainda nesta etapa; e (3) ***Implementação da Aplicação de Processo de Software*** (componentes de processo implementados e processo instanciado) que consiste na seleção dos componentes implementados para o estabelecimento de características dependentes diretamente do suporte ferramental envolvido na execução do processo instanciado. Desta forma, essa atividade é conduzida levando em consideração a máquina de processos selecionada e sua forma de representação do modelo de processo em uma linguagem executável. Assim como na ED, esse processo segue um ciclo espiral, incremental e evolutivo, das etapas definidas.

Vale ressaltar, ainda que não seja o foco deste trabalho, a possibilidade de associar esta abordagem de Engenharia de Linha de Processos de Software a uma gestão de contexto com o intuito de estabelecer relações entre uma determinada situação de contexto e as características e componentes da linha de processos; compreender o contexto do projeto atual através de um conjunto de informações de contexto e identificar mudanças no contexto da organização, do projeto e da equipe, com o intuito de apoiar adaptações dinâmicas do processo instanciado a partir da linha. O contexto pode se entendido como qualquer informação usada para caracterizar a situação de uma entidade (DEY *et al*., 2001). O detalhamento da abordagem de gestão de contexto completa faz parte da pesquisa de doutorado de NUNES (NUNES *et al.,* 2010).

3.2.1 - Análise do Domínio de Processos de Software

A etapa de Análise do Domínio de Processos de Software tem como objetivo a identificação do conhecimento do domínio, seus conceitos e artefatos, e a representação explícita de suas variabilidades e opcionalidades, em um nível de abstração de fácil entendimento. Os conceitos são igualmente importantes para um maior entendimento do domínio como um todo, além de facilitar o entendimento dos elementos que interagem internamente em um componente. Assim, o resultado desta etapa é um modelo que reúne o conhecimento do domínio, denominado Modelo de Domínio de Processos de Software.

O termo domínio pode ser aplicado para especificar uma coleção ou família de processos que compartilham um conjunto de características comuns, ou seja, apresentam um conjunto de artefatos ou elementos de processos similares ou com objetivos similares. Essa coleção pode ser entendida como um conjunto de processos similares, que agrupam os elementos de processos necessários para desenvolver e gerenciar produtos de software ou podem representar apenas processos que apresentam similaridades dentro de uma disciplina como, por exemplo, gerência da qualidade ou engenharia de requisitos, dentre outras. Desta forma, o domínio deve representar todo o conjunto de elementos de processos que podem vir a ser reutilizados, de acordo com o escopo da linha de processos a ser criada, que é determinado pela organização.

Esta etapa é composta de quatro atividades: *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*.

A atividade de ***Identificação dos Escopos do Domínio de Processos de Software*** situa o domínio em relação a seus escopos. Um diagrama de escopo representa subáreas do domínio, com um panorama geral do domínio no contexto da organização. Pode ser evoluído de acordo com informações adquiridas, refletindo mudanças nas subáreas definidas, nos relacionamentos com outros domínios, etc. Uma das formas de organizar internamente o escopo do domínio é a divisão em subáreas que representam diferentes aspectos na instanciação de um processo: *escopo padrão* e *escopos especializados*. No escopo *padrão*, são organizadas as características que representam um processo padrão para a organização, que descreve os elementos de processo que devem estar presentes em qualquer processo definido. Esse escopo trabalha todas as

características consideradas mandatórias para o domínio. Escopos *especializados* devem organizar características específicas de projetos. Características consideradas opcionais ou variáveis podem ser separadas em diferentes escopos especializados, que visam organizar um conjunto de características de acordo com informações específicas de projetos, como o tipo de software a ser desenvolvido, o paradigma de desenvolvimento a ser adotado, o nível de maturidade a ser considerado pela organização, as características da equipe, da qualidade do produto a ser desenvolvido e do próprio projeto, dentre outras.

A atividade ***Captura do Conhecimento do Domínio de Processos de Software*** visa capturar informações e conhecimentos dentro de um domínio de processos de software. Diferentes estratégias de captura visam coletar informações de múltiplas fontes de informação, 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, 1999) e Scrum (SCHWABER, 2004); e (3) Modelos de processos existentes dentro da organização ou no conhecimento recuperado de instâncias executadas dos projetos de desenvolvimento de software através da análise ou mineração de processos. A manipulação de informações depende da capacidade de conhecimento para realizar a análise e combinação das informações.

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. A atividade é definida em três tarefas: *Detecção de Equivalências; Detecção de Opcionalidades* e *Detecção de Variabilidades* (Figura 3.4). A tarefa de Detecção de Equivalências tem por objetivo identificar os elementos que representam similaridades dentro do domínio. Os aspectos comuns identificados representam artefatos base, não variáveis, para a instanciação de novos processos. 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. A etapa de Detecção de Variabilidades determina pontos dos modelos que são pontos configuráveis do domínio através de variantes, alternativas que irão permitir a configuração de processos de acordo com contextos específicos de aplicação. A comparação de elementos de modelos de processos deve levar em consideração que elementos com a mesma semântica podem ter sido especificados através de sintaxes diferentes. Outro critério

importante a ser definido é se a comparação será realizada entre todos os elementos, ou apenas entre elementos que possuem o mesmo "tipo" (e.g., comparação apenas entre papéis, comparação apenas entre unidades de trabalho, comparação apenas entre produtos de trabalhos).

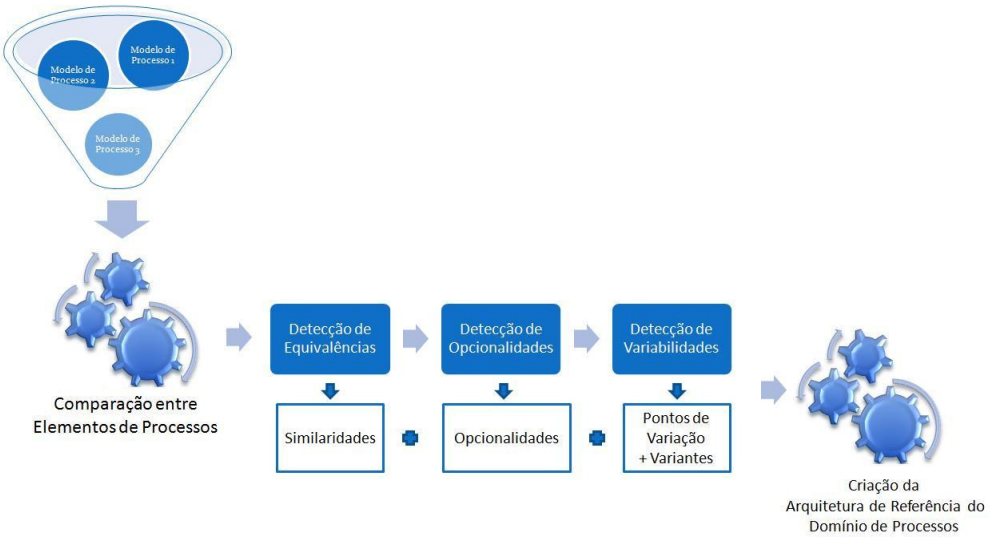


Figura 3.4 - Análise de Similaridades e Variabilidades no Domínio de Processos de Software

A atividade seguinte consiste na ***Modelagem do Conhecimento do Domínio de Processos de Software***, uma especificação do resultado desta análise através do Modelo de Domínio de Processos de Software. Este modelo deve ser expresso de forma a guiar a configuração de novos processos. Dentre os modelos de domínio, o mais conhecido na literatura é o modelo de características, o qual descreve a teoria do domínio (ARANGO e PRIETO-DIAZ, 1991). O modelo de características é um elemento chave para relacionar os conceitos de mais alto nível aos demais artefatos de análise e também para os artefatos de projeto, além de ser considerado o melhor caminho para efetuar o recorte do domínio para a construção de uma aplicação (BLOIS, 2006).

Autores (GRISS *et al.,* 1998, KANG *et al.,* 2002) propõem diferentes tipos de classificações para características, com o intuito de distinguir o tipo de informação que elas representam. No entanto, o foco dos modelos propostos na literatura é na modelagem de características. Dessa forma, são propostas para atender a modelagem em LPS e não apresentam os meios necessários para representar os conceitos envolvidos no domínio de processos de software.

Por outro lado, as abordagens de modelagem de Linhas de Processos de Software existentes (ROMBACH, 2006, WASHIZAKI, 2006, ARMBRUST *et al.*, 2008, BARRETO *et al*., 2009) dão maior ênfase na especificação da estrutura de

arquitetura de componentes de processos, não indicando a representação no nível de características. Além disso, o tópico representação de variabilidades é tratado como secundário, apresentando de forma superficial alguma modelagem adotada, sem a devida adaptação para a área de Linha de Processos de Software.

Desta forma, como contribuição deste trabalho, um meta-modelo, denominado *OdysseyProcess-FEX*, foi construído com o propósito de representar explicitamente os conceitos de características, variabilidade e opcionalidade associados à representação dos elementos que constituem a definição de processos de software, como unidades de trabalho, papéis e produtos de trabalho. A notação *OdysseyProcess-FEX* foi especificada para representar simbolicamente o meta-modelo proposto. Essa estrutura conceitual para a modelagem de variabilidades, através de um modelo de características para o domínio de processos de software é apresentada na próxima seção.

3.3 – *ODYSSEYPROCESS-FEX*

O meta-modelo *OdysseyProcess-FEX* foi especificado com o intuito de prover uma estrutura de representação de conceitos de modelos de características de Linhas de Processos de Software. Este meta-modelo visa explicitar a representação de conceitos de elementos de processos de software com a representação de variabilidades, em concordância com os requisitos levantados na Seção 2.4.

A identificação dos elementos especificados pelo meta-modelo *OdysseyProcess-FEX* foi realizada através da análise de diferentes modelos de processos existentes como o *OpenUP*[2] e o RUP (*Rational Unified Process)* (IBM, 2009); meta-modelos como o SPEM v.2 (OMG, 2008) e ontologias como a ontologia de processos (FALBO E BERTOLLO, 2005); o conjunto de representações analisadas no Capítulo 2 e o meta-modelo de características do domínio de software *Odyssey-FEX* (OLIVEIRA, 2006).

Com relação à identificação do conjunto de informações de processos que deveriam ser modeladas dentro do escopo deste trabalho, a base conceitual mais fortemente utilizada foi o meta-modelo de conceitos de processos de software SPEM 2.0. Promovido no âmbito da OMG (*Object Management Group*), o SPEM utiliza a orientação a objeto e usa a UML como notação.

No entanto, para atender aspectos específicos de uma modelagem de variabilidades, foi necessária a construção de um meta-modelo que unificasse os

[2] Site: http://www.eclipse.org/epf/openup_component/openup_vision.php

principais elementos de processos de software com os conceitos existentes em um modelo de características, e viabilizar a representação do conhecimento adquirido na fase de Análise de Domínio da Engenharia de Domínio de Processos. A base para trabalhar esses conceitos de modelagem de características foi o meta-modelo da notação *Odyssey-FEX* (OLIVEIRA, 2006), conforme mencionado anteriormente. Trata-se de uma notação mais abrangente para representação das variabilidades referente às características de um domínio de software.

Junto ao meta-modelo proposto é apresentada a notação *OdysseyProcess-FEX,* que tem como objetivo representar simbolicamente os conceitos formalizados pelo meta-modelo.

3.3.1 – Domínio de Engenharia de Requisitos de Software: Uma Breve Descrição

Para facilitar o entendimento dos conceitos presentes em um modelo de características do domínio de processos de software, é descrito a seguir parte do domínio de Engenharia de Requisitos de Software, que é utilizado como exemplo ao longo desta dissertação, definido a partir da análise de informações coletadas em uma revisão da literatura sobre a área de requisitos.

A ***Engenharia de Requisitos*** é uma das disciplinas que constituem o processo de desenvolvimento de software e inclui atividades através das quais requisitos de um produto de software são coletados, documentados e gerenciados ao longo de todo o ciclo de vida do software. *Requisito* é um conceito de papel central no desenvolvimento de software, uma vez que uma das principais medidas do sucesso de um software é o grau no qual ele atende aos objetivos para os quais foi construído. Como especializações deste conceito, temos os conceitos referentes a requisitos de sistema classificados como *Requisitos Funcionais* e *Requisitos Não-funcionais*.

Processos de Engenharia de Requisitos podem variar muito em função de diferentes fatores, como diversidade da cultura organizacional, dos domínios de aplicação e dos objetivos de projetos. Ainda assim, é possível estabelecer um conjunto de atividades e tarefas básicas que devem ser consideradas na definição de um processo de Engenharia de Requisitos. Dentre as atividades mandatórias podemos citar: *Levantamento de Requisitos; Análise de Requisitos; Especificação de Requisitos; Verificação & Validação de Requisitos;* e *Gerência de Requisitos*.

A atividade de ***Levantamento de Requisitos*** corresponde à descoberta de requisitos através da execução das tarefas: (1) *Definir Visão*, que permite a identificação

dos diferentes envolvidos e de uma definição inicial do problema a ser resolvido, e (2) *Elicitação de Requisitos*, que corresponde à tarefa de coletar informações de diferentes fontes para especificar as necessidades a serem atendidas. Como alternativas para a realização da tarefa de *Elicitação de Requisitos* podemos citar as tarefas: *Realizar Entrevistas, Aplicar Questionários, Usar Cenários, Construir Protótipos e Realizar Brainstorming*. Essas alternativas podem ser executadas em conjunto, de forma complementar. A tarefa *Definir Glossário*, ainda envolvida nesta atividade, é opcional e tem como função principal definir termos envolvidos no projeto a ser desenvolvido para construir um entendimento comum entre os envolvidos. Os produtos de trabalho *Documento Visão* e *Documento de Requisitos* são artefatos resultantes das duas primeiras tarefas. A tarefa *Definir Glossário* possui como produto de trabalho um documento chamado *Glossário*, que é considerado opcional no domínio como um todo.

A atividade de ***Análise de Requisitos*** corresponde a um trabalho de negociação e priorização dos requisitos a serem desenvolvidos. Possui como tarefas mandatórias: (1) *Priorizar Requisitos* e (2) *Negociar Requisitos*. A tarefa de *Classificar e Organizar Requisitos* depende de características de projeto e de sistemas, sendo, portanto, opcional.

A atividade de ***Especificação de Requisitos*** corresponde à documentação dos requisitos levantados em diferentes graus de formalismo, representando um ponto de múltiplas alternativas, desde descrições básicas das funcionalidades a serem disponibilizadas pelo sistema a ser desenvolvido, como no caso de metodologias ágeis, que usam *estórias de usuários* como forma de representação de requisitos; até especificações mais formais que usam *documentos em linguagens naturais* associados a documentos que usam *modelos como casos de usos*. Duas dessas alternativas são excludentes entre si, já que entram em conflito com a escolha de metodologia escolhida para o desenvolvimento de software. Desta forma, o uso de uma *Documentação usando Estórias de Usuários exclui a Documentação usando Casos de Uso*. A especificação de requisitos através da *Documentação usando Casos de Uso requer o Uso de Cenários* durante a tarefa de Elicitação de Requisitos, para guiar a descrição dos fluxos envolvidos na descrição dos Casos de Uso. Deve-se executar uma tarefa que mantenha a rastreabilidade com os requisitos listados no *Documento de Requisitos*. O conceito de rastreabilidade corresponde à capacidade de rastrear um elemento do projeto a outros elementos correlatos.

A atividade de ***Verificação & Validação de Requisitos*** corresponde a tarefas de verificação e validação dos requisitos com relação a aspectos de consistência, resolução de conflitos, conformidade e atendimentos de interesses. É composta da atividade de *Revisão de Requisitos*, mandatória para realizar verificações nos requisitos levantados, e das tarefas *Elaborar Roteiro de Homologação* e *Homologar,* que constituem a parte de validação. A atividade de *Revisão de Requisitos* inicia-se com a tarefa de Identificação de Inconsistências nas Especificações de Requisitos, que pode ser realizada através de uma ou mais das seguintes alternativas: *Realizar Leitura Ad-Hoc*; *Realizar Leitura usando Checklists* e *Realizar Leitura baseada em Perspectivas*. A seguir, a tarefa *Realizar uma Revisão Técnica Formal* deve ser conduzida através de uma reunião para discutir as considerações dos envolvidos e registrar em uma ata, as decisões de correções dos defeitos e inconsistências identificadas. Por último, as correções necessárias devem ser realizadas visando gerar *Especificações de Requisitos Revisadas*, consistentes com os objetivos do projeto. A tarefa *Elaborar Roteiro de Homologação* possui como produto de trabalho o documento *Roteiro de Homologação.*

Como mudanças nos requisitos ocorrem em todas as fases do processo de desenvolvimento de software, a atividade de ***Gerência de Requisitos*** deve ser conduzida de forma identificar, controlar e rastrear requisitos e mudanças em qualquer momento ao longo do ciclo de vida do software. Como tarefas que compõem essa atividade, podemos citar: *Controlar mudanças, Controlar Versões e Acompanhar Estado dos Requisitos*.

O principal papel presente na disciplina de Engenharia de Requisitos consiste no papel de *Analista de Requisitos,* que é o executante de várias das tarefas descritas. O papel *Stakeholder* é considerado como adicional ao papel do analista e é consultado para a execução de tarefas envolvidas nas atividades de *Levantamento, Análise e Verificação & Validação de Requisitos*. A participação do *Desenvolvedor* é opcional e pode ser adicionada durante atividades de Especificação de Requisitos.

3.3.2 – Meta-Modelo *OdysseyProcess-FEX*

O meta-modelo proposto foi desenvolvido para viabilizar a definição de diretrizes e da estrutura disponível para a construção de modelos de características em uma Linha de Processos de Software. Desta forma, o meta-modelo construído visa representar explicitamente os conceitos de características, variabilidade e opcionalidade associados à representação dos elementos que constituem a definição de processos de

software, como atividades, tarefas, papéis e produtos de trabalho. Além disso, deve definir o conjunto de relacionamentos disponíveis entre os conceitos representados.

O meta-modelo *OdysseyProcess-FEX* possui um conjunto de restrições e propriedades, que juntas constituem as regras de boa formação do modelo. Essas regras direcionam a construção e a verificação de consistência de um modelo de características do domínio de processos de software.

Tal meta-modelo é representado através de um diagrama de classes, onde cada classe representa um elemento ou relacionamento presente no modelo de características. O meta-modelo *OdysseyProcess-FEX* É composto por três pacotes (Figura 3.5):

- ***Principal***: representa a taxonomia das características;
- ***Relacionamentos***: representa as especificações das propriedades dos relacionamentos existentes entre as características; e
- ***Regras de Composição***: representa as especificações das regras de dependência e exclusividade entre as características.

A estrutura de descrição de cada pacote é realizada através de uma explicação detalhada de cada elemento pertencente ao pacote, especificado através da descrição dividida em cinco campos:

- **Descrição**: representa uma descrição do elemento em alto nível. Especifica o significado principal do elemento;
- **Hierarquia**: especifica quais são as classes existentes em uma possível hierarquia, ou seja, determina as subclasses ou superclasses, do elemento descrito;
- **Atributos**: especifica o conjunto de atributos associados ao elemento. Para cada atributo, é definido o seu tipo básico, a sua cardinalidade e, caso exista, o seu valor *default*;
- **Associações**: estabelece a semântica das associações que podem ser estabelecidas entre elementos do meta-modelo. Item de descrição incluído no *Pacote de Relacionamentos* e no *Pacote Regras de Composição*. Descreve o papel dos elementos envolvidos nos relacionamentos estabelecidos; e
- **Restrições**: especifica a descrição de restrições que direcionam a construção e a verificação de consistência do modelo de características.

O conjunto de todas as restrições especificadas para cada elemento constitui as regras de boa formação do meta-modelo.

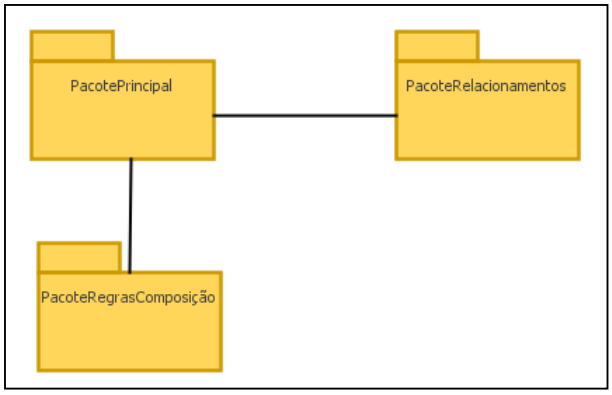


Figura 3.5 - Representação da estrutura em pacotes do meta-modelo *OdysseyProcess-FEX*

3.3.2.1 - Pacote Principal

O pacote Principal (*Core*) do meta-modelo *OdysseyProcess-FEX* é constituído por características, suas diversas categorias e atributos específicos. A definição deste pacote foi fortemente influenciada pelo meta-modelo SPEM v.2 (OMG, 2008). No total, é composto por sete categorias de características: *Característica Disciplina, Característica Atividade, Característica Tarefa, Característica Prática, Característica Conceito, Característica Papel e Característica Produto de Trabalho*. A Figura 3.6 apresenta o detalhamento deste pacote.

As categorias de características Atividade e Tarefa descrevem as unidades de trabalho do domínio. A característica Tarefa corresponde à menor unidade de trabalho considerada elementar. Já a característica Atividade consiste em um agrupamento lógico de unidades menores de trabalho, que podem corresponder a outras Atividades ou a Tarefas. A característica Disciplina representa uma categorização das unidades de trabalho em áreas de interesse maiores no domínio como um todo. A característica Prática indica uma estratégia comprovada para realizar um trabalho, representando uma abordagem ortogonal aos outros elementos do domínio. A característica Conceito representa uma unidade de conhecimento que define um objeto de interesse dentro do domínio. A característica Papel estabelece um conjunto de habilidades e competências disponibilizadas por um indivíduo ou conjunto de indivíduos para a execução do trabalho. Por último, a característica Produto de Trabalho representa os artefatos consumidos, produzidos ou modificados pelas unidades de trabalho do domínio.

A escolha da participação das categorias acima citadas deve ser baseada no tamanho do domínio e dos elementos que o compõem. Por exemplo, as categorias

Disciplina e Atividade têm por objetivo categorizar as unidades de trabalho e, por isso, podem ser omitidas, sendo seu uso opcional, de acordo com a sua necessidade diante da dimensão do domínio a ser representado.

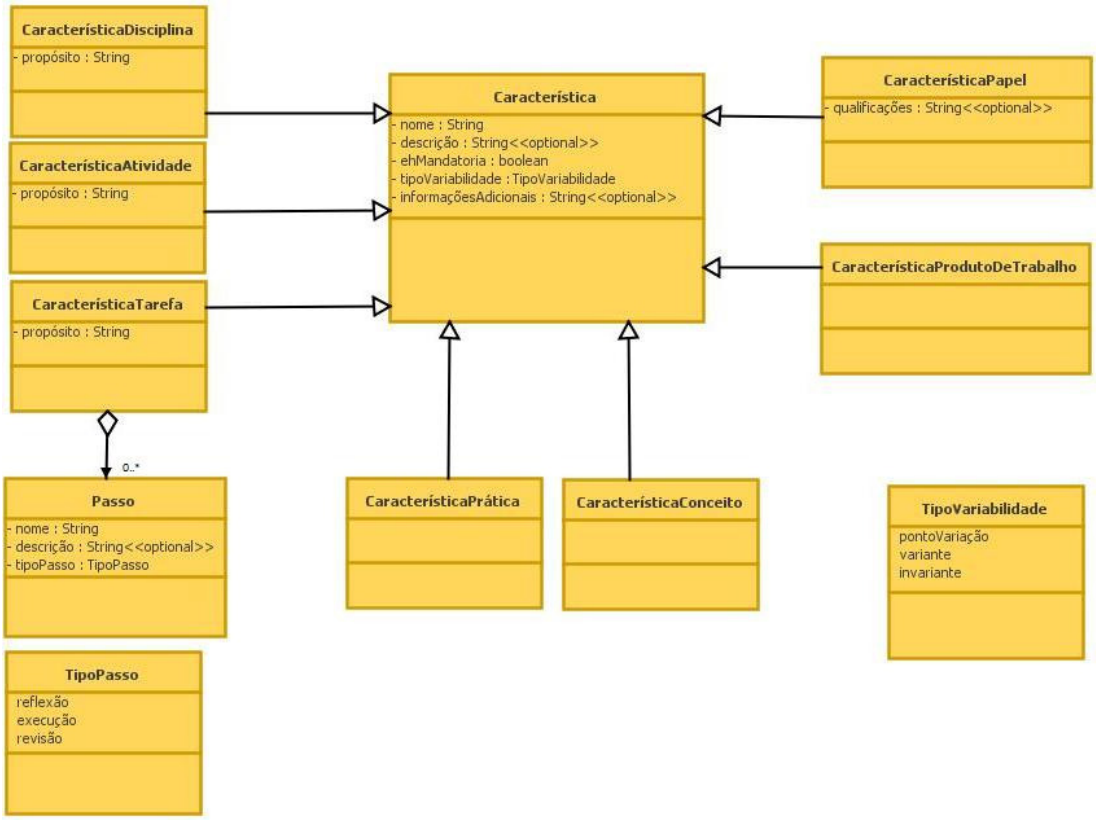


Figura 3.6 - Meta-modelo *OdysseyProcess-FEX*: Pacote Principal

No domínio exemplo de Engenharia de Requisitos, a característica *Engenharia de Requisitos* é um exemplo de característica da categoria Disciplina; as características *Levantamento de Requisitos, Análise de Requisitos* ou *Especificação de Requisitos* são exemplos de características da categoria Atividade; as características *Elicitação de Requisitos*, *Negociar Requisitos* ou *Homologar* são exemplos de características da categoria Tarefa; as características *Documento de Visão, Documento de Requisitos, Modelo de Caso de Uso* ou *Roteiro de Homologação* são exemplos de características da categoria Produto de Trabalho; as características *Analista de Requisitos, Stakeholder* ou *Desenvolvedor* são exemplos de características da categoria Papel; as características *Requisitos* ou *Atributos de Requisitos* são exemplos de características da categoria Conceito; e a característica *Desenvolvimento Dirigido por Caso de Uso* é um exemplo de característica da categoria Prática.

A descrição detalhada de cada elemento participante deste pacote, descrita segundo os cinco campos estabelecidos anteriormente, encontra-se no Anexo I.

3.3.2.2 - Pacote Relacionamentos

O pacote Relacionamentos (*Relationships*) consiste na representação das relações disponibilizadas entre as características. A Figura 3.7 apresenta a classe abstrata *Relacionamento* e suas especializações: *Alternativo, Herança, Associação, Agregação, Composição, LigaçãoPapelTarefa, LigaçãoProdutoDeTrabalhoTarefa e LigaçãoPapelProdutoDeTrabalho*.

O relacionamento *Alternativo* consiste no relacionamento existente entre um ponto de variação e cada uma de suas variantes. Uma *Herança* é um relacionamento entre uma característica mais geral e uma característica mais específica, permitindo uma organização hierárquica entre as características envolvidas. A *Associação* representa a relação, uni ou bidirecional, entre duas características, podendo possuir um estereótipo que especifica o tipo da ligação. O relacionamento de *Agregação* é um tipo de Associação que especifica um relacionamento todo/parte. O relacionamento de *Composição* também especifica um relacionamento todo/parte. No entanto, 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. O relacionamento *LigaçãoPapelTarefa* estabelece um relacionamento de associação entre uma característica Tarefa e uma ou várias características Papel participantes na sua execução. O relacionamento *LigaçãoProdutoDeTrabalhoTarefa* estabelece um relacionamento de associação entre uma característica Tarefa e uma ou várias características ProdutoDeTrabalho que representam produtos de trabalho a serem consumidos, produzidos ou modificados pela execução da unidade de trabalho. *LigaçãoPapelProdutoDeTrabalho* 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.

No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplos de relacionamentos **Alternativos**, as ligações entre as características *Elicitação de Requisitos e Realizar Entrevistas*; *Elicitação de Requisitos e Aplicar Questionário*, *Especificação de Requisitos e Documentar usando Casos de Uso* ou *Especificação de Requisitos e Documentar usando Linguagem Natural*. Como exemplo

de relacionamentos de **Herança**, podemos citar as ligações entre as características *Requisitos e Requisitos Funcionais* ou *Requisitos e Requisitos Não-funcionais*. Como exemplo de relacionamento de **Associação**, podemos citar a ligação entre as características *Engenharia de Requisitos e Requisitos*. Como exemplos de relacionamentos de **Agregação**, podemos citar as ligações entre as características *Levantamento de Requisitos e Definir um Glossário, Análise de Requisitos e Classificar e Organizar Requisitos*. Como exemplos de relacionamentos de **Composição**, podemos citar as ligações entre as características *Engenharia de Requisitos e Levantamento de Requisitos, Engenharia de Requisitos e Análise de Requisitos* ou *Engenharia de Requisitos e Especificação de Requisitos*. Como exemplos de relacionamentos de **LigaçãoPapelTarefa**, podemos citar as ligações entre as características *Analista de Requisitos e Definir Visão, Analista de Requisitos e Negociar Requisitos* ou *Stakeholder e Elicitação de Requisitos*. Como exemplos de relacionamentos de **LigaçãoProdutoDeTrabalhoTarefa**, podemos citar as ligações entre as características *Definir Visão e Documento de Visão, Elicitação de Requisitos e Documento de Requisitos* ou *Elaborar Roteiro de Homologação e Roteiro de Homologação*. Como exemplos de relacionamentos de **LigaçãoPapelProdutoDeTrabalho**, podemos citar as ligações entre as características *Analista de Requisitos e Documento de Visão* ou *Analista de Requisitos e Documento de Requisitos.*

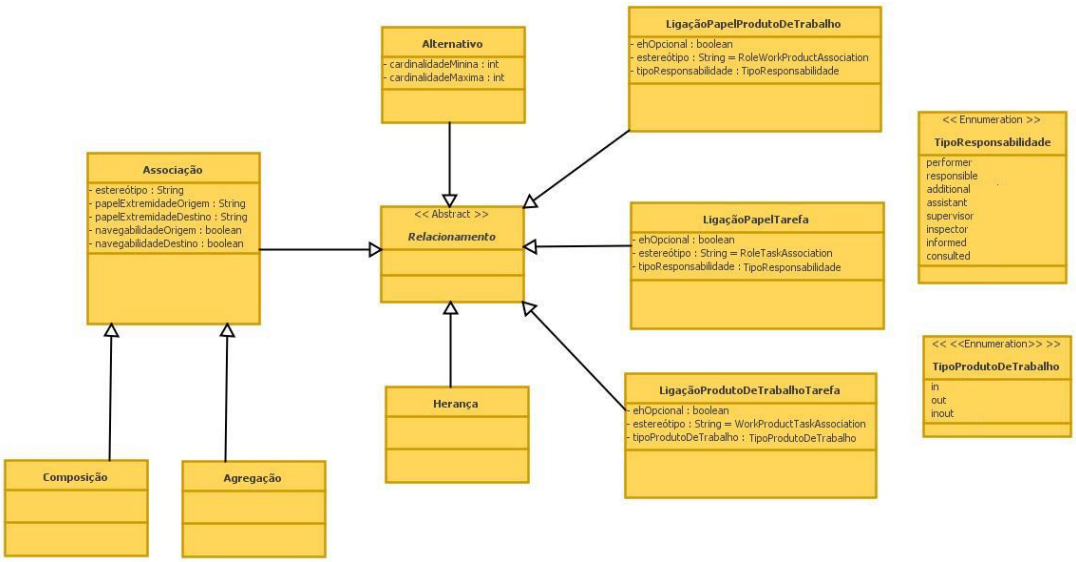


Figura 3.7 - Meta-modelo *OdysseyProcess-FEX*: Pacote Relacionamentos

A descrição detalhada de cada elemento participante deste pacote, descrita segundo os cinco campos estabelecidos anteriormente, encontra-se no Anexo II.

3.3.2.3 - Pacote Regras de Composição

Um dos requisitos para uma notação que represente variabilidade é a necessidade de representação de relações de dependência e mútua exclusividade para quaisquer conjuntos de elementos do modelo (OLIVEIRA, 2006).

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 3.8.

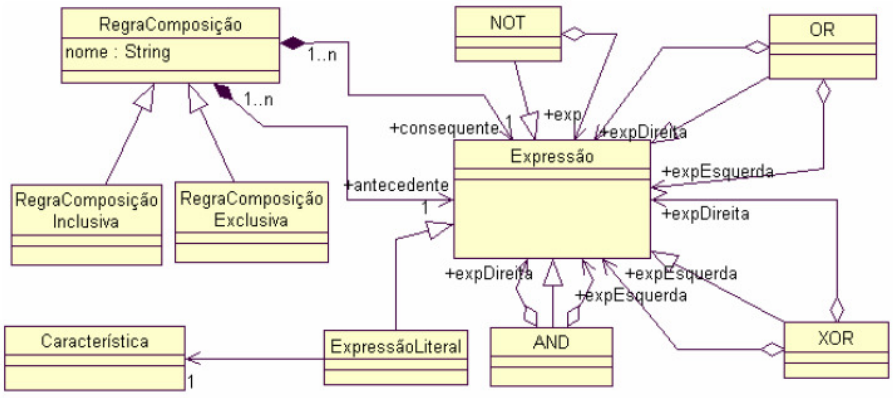


Figura 3.8 - Meta-modelo *OdysseyProcess-FEX*: Pacote Regras de Composição (OLIVEIRA, 2006)

Regras de Composição são representadas pela classe *RegraComposição* que pode ser especializada pelas classes *RegraComposiçãoInclusiva*, que representa uma regra de dependência entre características, e *RegraComposiçãoExclusiva*, que representa uma regra que expressa mútua exclusividade entre características. Uma Regra de Composição pode estabelecer restrições entre duas ou mais características através de expressões combinadas por meio de operadores booleanos. O intuito é permitir uma representação mais expressiva da semântica de um modelo de características.

Uma Regra de Composição tem a seguinte estrutura: **Antecedente + palavra-chave + Consequente**. O antecedente e o consequente são expressões, que podem ser literais ou booleanas, e denotam uma característica do domínio ou uma combinação entre estas. Essas expressões são representadas pela classe abstrata *Expressão,* e suas subclasses *AND, NOT, XOR, OR e ExpressãoLiteral*. A palavra-chave apresentada na estrutura define o tipo de Regras de Composição: Regras Inclusivas são denotadas pela palavra-chave "requer", enquanto que Regras Exclusivas são denotadas pela palavra-chave "exclui".

No domínio exemplo de Engenharia de Requisitos, podemos identificar dois exemplos de Regras de Composição. A regra *Documentar usando Casos de Uso* ***requer*** *Usar Cenários* constitui um exemplo de Regra de Composição Inclusiva. Já a regra *Documentar usando Estórias de Usuários* ***exclui*** *Documentar usando Casos de Uso* constitui um exemplo de Regra de Composição Exclusiva.

Os elementos deste pacote também são detalhados através dos campos anteriormente especificados: descrição, classes relacionadas, atributos, associações e restrições. Essa descrição detalhada encontra-se no Anexo III.

3.3.3 – Notação *OdysseyProcess-FEX*

A notação *OdysseyProcess-FEX* tem como objetivo representar simbolicamente os conceitos formalizados pelo meta-modelo descrito na seção anterior.

As características podem ser classificadas quanto à categoria, variabilidade e opcionalidade. A classificação de opcionalidade é ortogonal às outras classificações, de categorias e variabilidade (Figura 3.9). Isso significa que qualquer tipo de categoria, das sete definidas, com qualquer tipo de variabilidade (invariante, ponto de variação, ou variante) pode ser classificada como mandatória ou opcional.

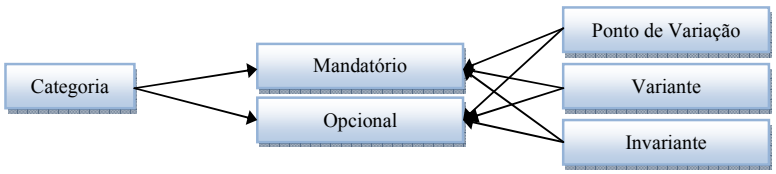


Figura 3.9 - Classificação Ortogonal Opcionalidade X Categoria e Opcionalidade X Variabilidade na notação *OdysseyProcess-FEX*

A classificação de variabilidade possui a seguinte restrição: as categorias Prática e Conceito só podem ser classificadas como invariantes. A combinação dessa classificação com a classificação quanto à categoria está representada na Figura 3.10. Vale ressaltar que os diferentes tipos de categoria são mutuamente exclusivos entre si, assim como os tipos de variabilidade e opcionalidade.

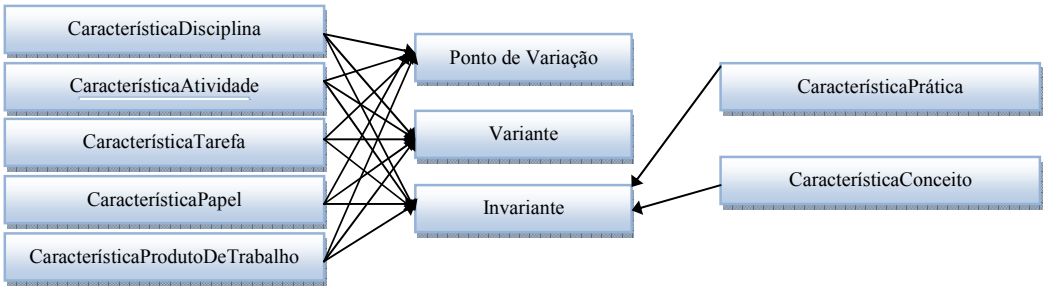


Figura 3.10 - Classificação Categoria X Variabilidade na notação *OdysseyProcess-FEX*

3.3.3.1 - Categorias das características

Cada característica pode ser classificada segundo sete categorias definidas pelo meta-modelo *OdysseyProcess-FEX*, descritas na Seção 3.2.2.1. Para cada categoria definida foi especificado um símbolo e um estereótipo associado à representação visual dos conceitos descritos. A Tabela 3.1 reúne todas as categorias, seus símbolos e apresenta um exemplo gráfico de alguns dos elementos descritos no domínio exemplo de Engenharia de Requisitos de Software, especificado na Seção 3.3.1, que correspondem a cada categoria apresentada.

Tabela 3.1- Categorias e ícones das características na notação *OdysseyProcess-FEX*

Categoria		Ícone / Estereótipo	Exemplo
Disciplina	Característica que representa uma categorização de trabalho, relacionado com uma grande área de interesse no âmbito do domínio como um todo.	***<<discipline>>***	No domínio exemplo de Engenharia de Requisitos, a característica *Engenharia de Requisitos* é um exemplo deste tipo de categoria. 
Atividade	Característica que representa o agrupamento de unidades de trabalhos menores, representadas por outras atividades tarefas.	***<<activity>>***	No domínio exemplo de Engenharia de Requisitos, as características *Levantamento de Requisitos, Análise de Requisitos* ou *Especificação de Requisitos* são exemplos de características da categoria Atividade. 
Tarefa	Característica que representa uma unidade fundamental de trabalho.	***<<task >>***	No domínio exemplo de Engenharia de Requisitos, as características *de Requisitos*, *Negociar Requisitos* ou *Homologar* são exemplos de características da categoria Tarefa. 
Prática	Característica que representa uma forma ou estratégia comprovada de realizar um trabalho.	***<<practice>>***	No domínio exemplo de Engenharia de Requisitos, a característica *Desenvolvimento Dirigido por Caso de Uso* é um exemplo de características da categoria Prática. 
Conceito	Característica que descreve ideias-chave, aborda temas gerais ou princípios básicos.	***<<concept>>***	No domínio exemplo de Engenharia de Requisitos, as características *Requisitos* ou *Atributos de Requisitos* são exemplos de características da categoria Conceito. 
Papel	Característica que representa um conjunto de habilidades, competências e		No domínio exemplo de Engenharia de Requisitos, as características *Analista de Requisitos, Stakeholder* ou *Desenvolvedor* são exemplos de características da categoria Papel.

Categoria		Ícone / Estereótipo	Exemplo
	responsabilidades de indivíduo(s).	***<<role>>***	
Produto de Trabalho	Característica que representa um artefato consumido, modificado ou produzido por uma tarefa.	 ***<<work product>>***	No domínio exemplo de Engenharia de Requisitos, as características *Documento de Visão, Documento de Requisitos, Modelo de Caso de Uso* ou *Roteiro de Homologação* são exemplos de características da categoria Produto de Trabalho. 

3.2.3.2 - Classificação de características quanto à Variabilidade

O conceito de variabilidade será trabalhado através da classificação de uma característica como Ponto de Variação, Variante ou Invariante. Essa classificação quanto à variabilidade é mutuamente excludente, ou seja, cada característica não pode ser classificada com mais de um dos tipos de variabilidade definidos.

A notação permite a identificação das características classificadas como ponto de variação e variantes associadas através do relacionamento Alternativo. No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplo de característica classificada como ponto de variação a característica *Elicitação de Requisitos*, que possui como variantes as características *Realizar Entrevista, Aplicar Questionário, Usar Cenários, Realizar Brainstoming* e *Construir Protótipos* (Figura 3.11).

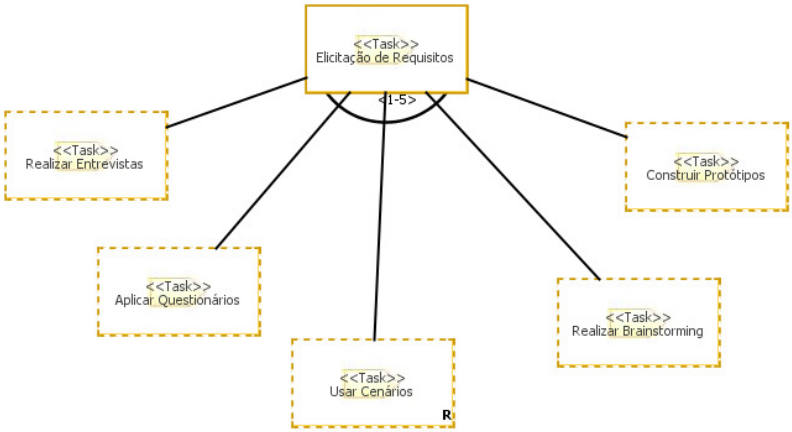


Figura 3.11 - Exemplos de características classificadas como Ponto de Variação e Variantes

3.3.3.3 - Classificação de características quanto à Opcionalidade

A opcionalidade classifica os elementos como mandatórios ou opcionais dentro do domínio de processos de software. Essa classificação determina a obrigatoriedade ou não da presença de determinado elemento nos múltiplos processos a serem instanciados

a partir dos artefatos da linha. Vale ressaltar que a opcionalidade é referente ao domínio como um todo. Características mandatórias são representadas por um retângulo formado por uma linha contínua e as características opcionais são modeladas através de um retângulo formado por uma linha tracejada.

No domínio exemplo de Engenharia de Requisitos, podemos citar as características *Levantamento de Requisitos, Análise de Requisitos* ou *Especificação de Requisitos* como exemplos de características mandatórias, e as características *Desenvolvedor* ou *Modelo de Caso de Uso* como exemplos de características opcionais (Figura 3.12).

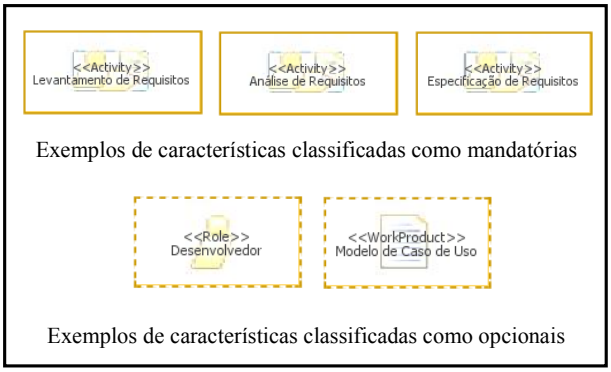


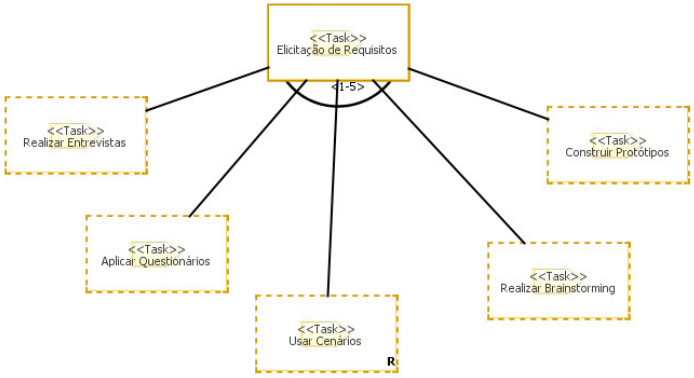
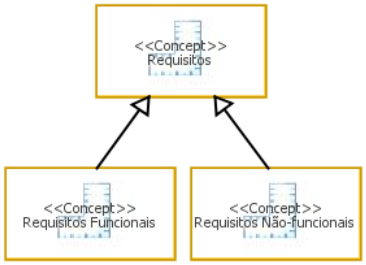
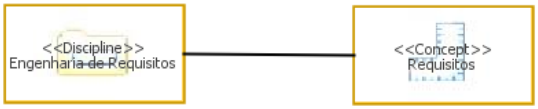
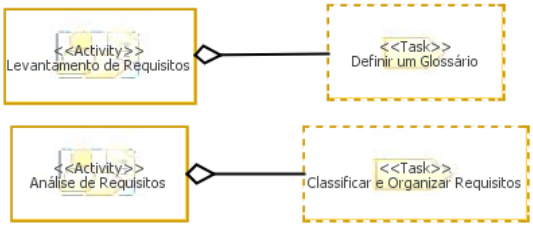
Figura 3.12- Exemplos de representação da classificação de opcionalidade

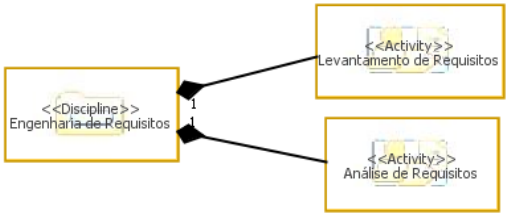
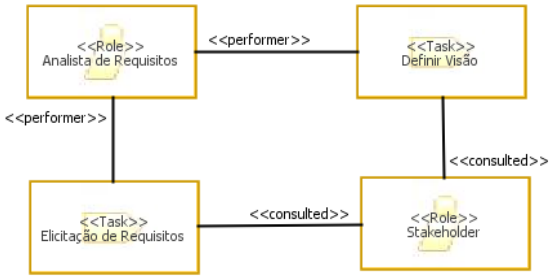
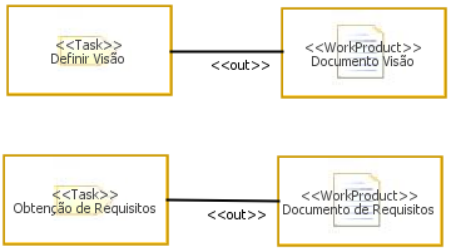
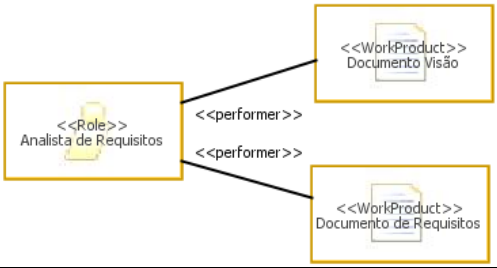
3.3.3.4 – Relacionamentos

Assim como a notação *Odyssey-FEX* (OLIVEIRA, 2006), a notação *OdysseyProcess-FEX* proposta valoriza a semântica dos relacionamentos em um modelo de características, oferecendo uma maior capacidade de representação e expressão. Alguns dos relacionamentos da UML (OMG, 2005), cuja incorporação à notação de modelagem de características está descrita na notação *Odyssey-FEX*, foram também incorporados a essa notação. No entanto, esses relacionamentos foram reavaliados quanto à combinação de categorias de características que podem estar envolvidos em cada relacionamento. O relacionamento que explicita a representação de variabilidade, Relacionamento Alternativo, foi incluído, porém com o estabelecimento de restrições das categorias de características que podem participar em cada extremidade do relacionamento. Nesta notação, os relacionamentos também não são apenas hierárquicos, mas representam um grafo cíclico, permitindo a expansão do modelo em várias direções.

Para cada relacionamento definido no meta-modelo, será especificado o símbolo associado à representação visual e um exemplo dentro do domínio de Engenharia de Requisitos de Software (Tabela 3.2).

Tabela 3.2 - Relacionamentos e representações na notação *OdysseyProcess-FEX*

Relacionamento	Símbolo	Exemplo
Alternativo		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplos de relacionamentos Alternativos as ligações entre as características *Elicitação de Requisitos e as variantes: Realizar Entrevistas*; *Aplicar Questionário; Usar Cenários; Construir Protótipos e Realizar Brainstorming.* 
Herança		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplo de relacionamento de Herança a ligação entre as características *Requisitos e Requisitos Funcionais* ou *Requisitos e Requisitos Não-funcionais.* 
Associação		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplo de relacionamento de Associação a ligação entre as características *Engenharia de Requisitos e Requisito.* 
Agregação		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplos de relacionamentos de Agregação as ligações entre as características *Levantamento de Requisitos e Definir um Glossário, Análise de Requisitos e Classificar e Organizar Requisitos*. 

Relacionamento	Símbolo	Exemplo
Composição		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplos de relacionamentos de Composição as ligações entre as características *Engenharia de Requisitos e Levantamento de Requisitos* ou *Engenharia de Requisitos e Análise de Requisito*. 
Ligação PapelTarefa		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplos de relacionamentos de *LigaçãoPapelTarefa* as ligações entre as características *Analista de Requisitos e Definir Visão* ou *Stakeholder e Elicitação de Requisitos*. 
Ligação ProdutoDeTrabalho Tarefa		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplos de relacionamentos de *LigaçãoProdutoDeTrabalhoTarefa* as ligações entre as características *Definir Visão e Documento de Visão* ou *Elicitação de Requisitos e Documento de Requisito*. 
Ligação Papel ProdutoDeTrabalho		No domínio exemplo de Engenharia de Requisitos, podemos citar como exemplos de relacionamentos de *LigaçãoPapelProdutoDeTrabalho* as ligações entre as características *Analista de Requisitos e Documento de Visão* ou *Analista de Requisitos e Documento de Requisitos*. 

3.3.3.5 – Regras de Composição

As Regras de Composição expressam restrições de dependência ou mútua exclusividade existentes entre características. A semântica dessas regras influencia fortemente o recorte para a instanciação de um processo de software.

a. RegrasComposição

Regras de Composição são expressas pela seguinte estrutura:

Antecedente + *palavra-chave* + Consequente

Os componentes *Antecedente* e *Consequente* são expressões, que podem ser literais ou *booleanas* e representam uma característica ou combinação de características do domínio. A formação de expressões que representem um conjunto de características é realizada através da combinação de características com os seguintes operadores *booleanos*: *AND, OR, XOR, NOT*.

A Tabela 3.3 apresenta a representação gráfica de cada tipo de regra definida pelo meta-modelo associada a um exemplo dentro do domínio de Engenharia de Requisitos de Software.

Tabela 3.3 - Tipos de Regras de Composição e sua representação na notação *OdysseyProcess-FEX*

Tipo de Regra de Composição	Estrutura de Representação	Exemplo
Regra de Composição Inclusiva	Antecedente + ***requer*** + Consequente Graficamente uma Regra de Composição Inclusiva é representada por uma marcação "R", no canto inferior direito do retângulo, associada a. uma numeração (n), que representa a ordem sequencial de criação das regras: "R_n".	No domínio exemplo de Engenharia de Requisitos, podemos identificar a regra: ***Documentação usando Casos de Uso requer o Uso de Cenários***, expressa graficamente pelo símbolo R nas características pertencentes à regra. 
Regra de Composição Exclusiva	Antecedente + ***exclui*** + Consequente Graficamente uma Regra de Composição Inclusiva é representada por uma marcação "X", no canto inferior esquerdo do retângulo, associada a. uma numeração (n), que representa a ordem sequencial de criação das regras: "X_n".	No domínio exemplo de Engenharia de Requisitos, podemos identificar a regra: ***Documentação usando Estórias de usuários exclui Documentação usando Casos de Uso,*** expressa graficamente pelo símbolo X nas características pertencentes à regra. 

3.3.4 – Exemplo de Utilização da Notação *OdysseyProcess-FEX*

A Figura 3.13 mostra a modelagem de característica construída utilizando-se a notação *OdysseyProcess-FEX*. Este exemplo foi construído considerando o domínio de Engenharia de Requisitos descrito na Seção 3.3.1. O modelo foi parcialmente apresentado nas seções anteriores com o intuito de auxiliar o entendimento dos elementos definidos no meta-modelo e notação.

3.4 – CONSIDERAÇÕES FINAIS

A abordagem de Linha de Processos de Software consiste em uma técnica de sistematização da reutilização de processos de software que aplica os conceitos de LPS, se utilizando da analogia entre software e processos de software, já argumentada por OSTERWEIL (1987). Com o uso da Linha de Processos, espera-se que, ao derivar um processo, o esforço requerido para a definição deste processo seja reduzido. A pesquisa de HOLLENBACH e FRAKES (1996) mostra que é possível diminuir em pelo menos dez vezes o tempo e o esforço necessário para criar a descrição de processo de um projeto quando se instancia um processo reutilizável, ao invés de criá-lo desde o início.

Uma definição para Linha de Processos de Software, dentro do contexto deste trabalho, foi apresentada neste capítulo. Além disso, uma abordagem de Engenharia de Linha de Processos de Software foi especificada com foco na fase de Análise do Domínio de Processos de Software, ainda pouco explorada por outros trabalhos. Devido à sua importância, o estudo desta fase é percebido pelo grupo de pesquisa como uma oportunidade e um diferencial desta abordagem. Desta forma, o suporte à derivação do resultado desta atividade da EDPS é provido pelo desenvolvimento de um meta-modelo, denominado *OdysseyProcess-FEX,* e sua notação correspondente.

O meta-modelo e a notação *OdysseyProcess-FEX* foram desenvolvidos para atender aos requisitos para representação de variabilidades levantados na Seção 2.4. Este fato pode ser interpretado como uma contribuição para uma modelagem mais abrangente das características e suas variabilidades, dentro do domínio de processos de software.

O suporte ferramental para a construção de modelos de características do domínio de processos de software é provido através da adaptação do ambiente Odyssey (ODYSSEY, 2011), descrita no Capítulo 5.

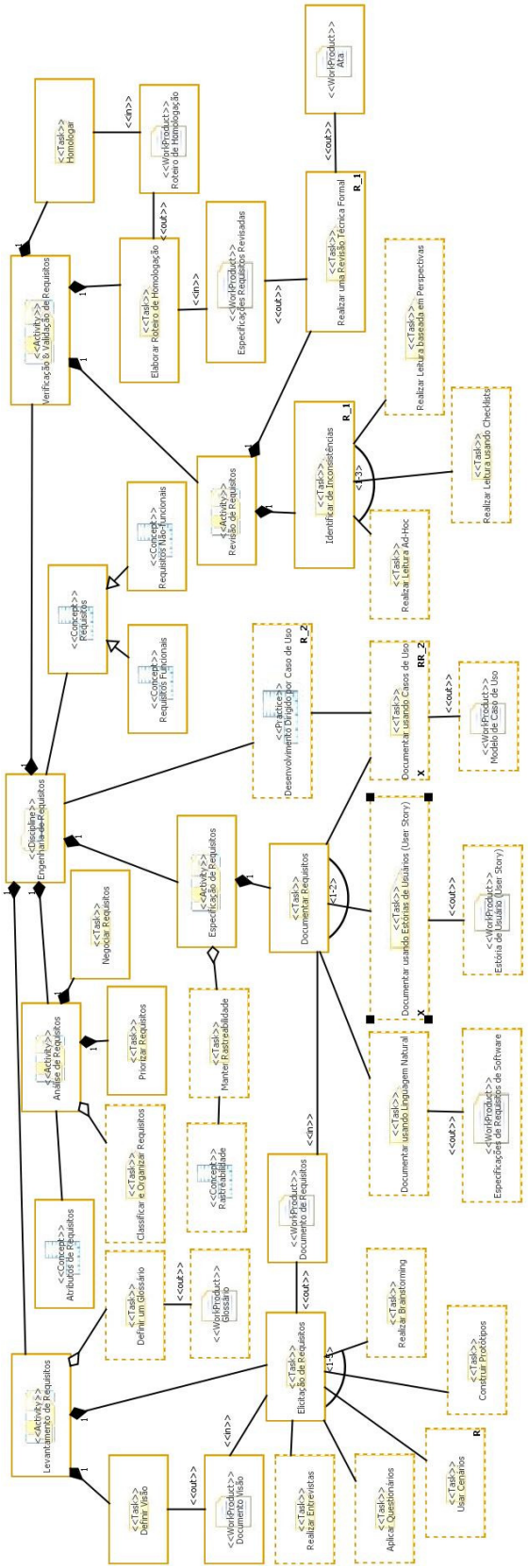


Figura 3.13 – Modelo de Características do domínio de Engenharia de Requisitos usando a notação *OdysseyProcess-FEX*

Capítulo IV
ESTUDO DE OBSERVAÇÃO

4.1 - INTRODUÇÃO

O meta-modelo e a notação *OdysseyProcess-FEX* foram apresentados como mecanismo para representação do conhecimento adquirido na etapa de Análise de Domínio da abordagem de Engenharia de Linha de Processos de Software proposta. Com o objetivo de caracterizar a viabilidade da aplicação de ambos na atividade de construção de Linhas de Processos de Software, um estudo preliminar foi realizado e encontra-se descrito neste capítulo.

O estudo realizado pode ser classificado como um estudo de observação, onde o participante realiza alguma tarefa enquanto é observado por um experimentador (SHULL *et al.,* 2001). Este tipo de estudo tem como finalidade coletar dados sobre como determinada tarefa é realizada. Através desse conjunto de informações, pode-se obter uma melhor compreensão de como um novo processo é utilizado.

O objetivo do estudo é apresentado na Tabela 4.1, seguindo o modelo descrito por WOLHIN *et al.* (2000).

Tabela 4.1 – Descrição do Objetivo do Estudo de Observação

Analisar o meta-modelo e a notação *OdysseyProcess-FEX*
Com o propósito de caracterizar a viabilidade de sua aplicação
Em relação à representação explícita de variabilidades na modelagem de Linhas de Processos de Software
Do ponto de vista do Engenheiro de Processos de Software
No contexto de Definição de Linhas de Processos de Software

Neste estudo, os indivíduos selecionados utilizaram o meta-modelo e a notação propostos no intuito de expressar, graficamente e de forma explícita, as variabilidades presentes no domínio de processos de software.

Os passos definidos para este estudo de observação foram: (a) definição dos participantes; (b) apresentação de um resumo do meta-modelo e da notação *OdysseyProcess-FEX* aos participantes; (c) utilização do meta-modelo e da notação; (d) avaliação do estudo. Estes passos são descritos a seguir.

Um estudo piloto foi executado com apenas um participante. O intuito deste piloto consistiu na verificação da aplicabilidade do estudo em termos de tempo e entendimento e qualidade do material utilizado.

4.2 – DEFINIÇÃO DOS PARTICIPANTES

Para a realização deste estudo, foram convidados quatro participantes com ampla experiência em modelagem de processos e com conhecimento do domínio de requisitos de software (domínio escolhido para realização deste estudo). No entanto, não tinham, ou tinham pouca, experiência em modelagem de características e na abordagem de Linha de Processos de Software.

Cada participante deveria, inicialmente, preencher: o Formulário de Consentimento (Anexo IV), que descreve dados informativos sobre o estudo e, ao ser assinado, confirma a concordância do participante com o mesmo; e o Formulário de Caracterização do Participante (Anexo V), que avalia o nível de conhecimento e experiência do participante em relação a diferentes tópicos relacionados ao estudo, como, por exemplo, definição de processos de software, linha de produtos de software, modelagem de características e linha de processos de software. Estas informações foram utilizadas como forma de auxiliar na identificação da aptidão do participante na execução do estudo e interpretação dos resultados obtidos, através da identificação de pontos que exercem ou podem exercer alguma interferência sobre os resultados.

4.3 – APRESENTAÇÃO DO RESUMO DO META-MODELO E DA NOTAÇÃO *ODYSSEYPROCESS-FEX*

Um resumo do meta-modelo e da notação *OdysseyProcess-FEX* foi distribuído previamente por e-mail aos participantes do estudo. A disponibilização do material foi importante para promover o conhecimento a respeito desta representação e permitir um melhor entendimento dos participantes.

Juntamente com as descrições, foi enviado um exemplo de utilização desta modelagem para a disciplina de Gerência de Configuração de Processos de Software, visando auxiliar os participantes na compreensão das propriedades definidas pela taxonomia utilizada no estudo.

O material provido aos participantes nesta etapa encontra-se nos Anexos VI e VII.

4.4 – UTILIZAÇÃO DO META-MODELO E DA NOTAÇÃO *ODYSSEYPROCESS-FEX*

Neste estudo, cada participante deveria exercer o papel de um Engenheiro de Processos de Software contratado por uma empresa de desenvolvimento de software. Essa empresa possuía uma Linha de Processos de Software definida para a disciplina de

Engenharia de Requisitos. Para atender exigências de mercado, a empresa passou a adotar a abordagem de LPS como paradigma de desenvolvimento de software baseado em reutilização. Para isso, precisou levar em consideração que a disciplina de Engenharia de Requisitos apresenta particularidades quando realizada dentro da LPS. A abordagem de LPS se aplica no desenvolvimento de uma família de sistemas de software e desempenha tarefas específicas dentro da disciplina de Engenharia de Requisitos, como a identificação de similaridades e variabilidades desses sistemas e a modelagem desse conhecimento através de modelos de características. Originalmente, a linha de processos definida não contemplava tais atividades. Assim, para atender o novo processo de desenvolvimento de software, novos elementos de processos devem ser incluídos na modelagem da linha de processos original, com o propósito de incluir variabilidades ou opcionalidades que representem particularidades da disciplina de Engenharia de Requisitos em LPS. Desta forma, a tarefa do participante era modelar os refinamentos do domínio necessários, a fim de contemplar tais variabilidades, utilizando a notação proposta.

Para a realização do estudo, o seguinte conjunto de atividades foi realizado: (1) Apresentação para o participante abordando os tópicos relevantes: Linha de Processos de Software; Modelagem de Características; o meta-modelo e a notação *OdysseyProcess-FEX* e o contexto de realização da tarefa; (2) Disponibilização do documento referente a tarefa a ser executada, contendo a descrição do exemplo de parte do domínio da disciplina de Engenharia de Requisitos de Software e uma especificação de detalhamento, que representa o refinamento do domínio para a inclusão da definição de processos cuja disciplina de Engenharia de Requisitos contemple Linha de Produtos de Software (Anexo VIII); (3) Disponibilização do modelo de características do domínio exemplo da disciplina de Engenharia de Requisitos de Software descrito anteriormente (Anexo IX). Após a apresentação, foi reservado um tempo para o participante tirar possíveis dúvidas sobre o conteúdo da apresentação e do modelo exemplo com o experimentador.

De posse destes documentos, os participantes deveriam modelar o domínio refinado. Todo material fornecido ao participante poderia ser utilizado durante a execução do estudo.

O resultado esperado desta atividade é um modelo de características do domínio da Disciplina de Engenharia de Requisitos que contemplasse o conjunto de refinamentos descritos no Anexo VIII.

Anotações foram registradas durante a execução da tarefa proposta. Ao final, um questionário de avaliação (Anexo XI) foi entregue, com o objetivo de confirmar os dados observados e coletar a opinião dos participantes sobre outros aspectos investigados neste estudo.

A principal questão investigada neste estudo é: "*A aplicação do meta-modelo e da notação OdysseyProcess-FEX é viável para a modelagem de Linha de Processos de Software?*". Planeja-se uma análise qualitativa dos dados a serem observados durante a execução da tarefa. Essa análise envolve o acompanhamento dos elementos criados; das inconsistências geradas; das dúvidas expressas; e das dificuldades enfrentadas pelos participantes para completar a tarefa.

4.5 – AVALIAÇÃO DO ESTUDO

O estudo de observação foi executado em quatro sessões, duas por dia, totalizando dois dias para a realização do estudo. Cada sessão foi realizada com o experimentador e um dos participantes, individualmente, e durou aproximadamente 02 horas.

A Tabela 4.2 apresenta um resumo da caracterização dos participantes deste estudo. Nos itens da questão 2, os níveis de experiência variam de 0 (nenhuma experiência) a 4 (usei em projetos na indústria), e, no item da questão 3, variam de 0 (nenhuma familiaridade) a 2 (muito familiar).

Tabela 4.2 - Resumo da caracterização dos participantes do estudo

Pergunta			**P1**	**P2**	**P3**	**P4**
1	Formação Acadêmica		Mestrado	Doutorado	Pós-Doutorado	Pós-Doutorado
2	2.1	Definição de Processos de Software (0-4)	4	4	4	4
	2.2	Linha de Produtos de Software (0-4)	0	0	1	1
	2.3	Modelagem de Características (0-4)	0	4	1	1
	2.4	Linha de Processos de Software (0-4)	0	0	1	1
3	Experiência no domínio de Engenharia de Requisitos de Software (0-2)		2	1	2	2

Durante o estudo, o tempo de realização da tarefa proposta foi medido para cada participante. Esses números são apresentados na Tabela 4.3 e levam em consideração apenas o intervalo entre o início de leitura dos refinamentos a serem executados no modelo base até o término do preenchimento das tabelas de apoio à modelagem (Anexo X) e encerramento do esboço do modelo final.

Tabela 4.3 - Tempo de execução da tarefa de cada participante do estudo

Participante	Tempo da tarefa (em minutos)
P1	66
P2	50
P3	52
P4	28

Nas seções seguintes, encontram-se descritos os resultados obtidos em cada sessão realizada neste estudo.

4.5.1 - Participante P1

O modelo final gerado seguiu os refinamentos especificados, sem acréscimo de conhecimento adicional da abordagem de Linha de Produtos de Software à modelagem. A tarefa foi conduzida utilizando como material de apoio o documento de descrição do meta-modelo e notação (Anexo VI). No entanto, o documento que contemplava as regras de boa formação (Anexo VII) foi pouco consultado, fato que gerou um conjunto de inconsistências no modelo gerado.

As características especificadas foram corretamente classificadas quanto à categoria existente na notação e sua aplicação dentro da linha de processos. Todas as características foram classificadas como opcionais, diante do correto entendimento que seriam utilizadas apenas no caso da instanciação de um processo dentro da abordagem de LPS, não sendo mandatórias no domínio como um todo. Já a classificação quanto à variabilidade não foi devidamente analisada, sendo todas as características classificadas como invariantes, mesmo em casos onde a modelagem indica uma classificação como ponto de variação e outros casos de classificação como variante. O relacionamento *Alternativo* não foi utilizado. No entanto, os relacionamentos *Associação* e *LigaçãoProdutoDeTrabalhoTarefa* foram utilizados incorretamente, como relacionamentos que agregassem o conceito de variabilidade. O conceito de cardinalidade foi atribuído a esses relacionamentos, mas sua utilização não tinha sido prevista para esses casos pela notação. Uma Regra de Composição Inclusiva foi corretamente criada relacionando três características.

Com relação à representação gráfica do modelo foi sugerida a utilização dos estereótipos em português; uso de cores e símbolos específicos para cada elemento. No entanto, a visualização pode ter sido prejudicada pela impressão do modelo em preto e branco e pelo seu tamanho reduzido. Em termos semânticos, o participante indicou a falta da possibilidade de especificação da ordem de execução das tarefas, conceito de seqüencialidade fortemente presente em modelagem de processos.

O participante ressaltou ainda, durante a execução da tarefa, a dificuldade de modelagem manual e que o uso de um suporte computacional poderia auxiliar a construção de modelos.

4.5.2 - Participante P2

Este participante apresentou dificuldades para a realização da tarefa. Apresentou diversos questionamentos sobre o modelo base descrito no Anexo IX, indicando a necessidade de inclusão de outros elementos e de mais semântica aos relacionamentos representados. Esta discordância em relação aos refinamentos propostos inviabilizou a finalização da realização da tarefa. No entanto, esta análise do modelo, associada ao vasto conhecimento do participante com modelagem conceitual, principalmente com ontologias, permitiu que diversas observações conceituais sobre a notação e sobre o meta-modelo proposto fossem expressas.

Uma das análises que foi verbalizada consistiu na identificação de representação de dois níveis de informação em um mesmo modelo: representação estrutural (visão estática), com a identificação de conceitos e entidades; e representação comportamental (visão dinâmica), com a identificação de elementos de um modelo de processo como disciplinas, atividades e tarefas. A granularidade dos elementos a serem representados no modelo foi também observada, demonstrando certo questionamento da fronteira entre tarefas e passos. O conjunto de relacionamentos proposto foi comparado com os relacionamentos da UML, com questionamento da necessidade de inclusão de outros a esse conjunto. Ressaltou a necessidade de suporte notacional para representar precedência entre alguns elementos representados no modelo, conceito de sequencialidade fortemente presente em modelagem de processos. Houve a tentativa de estabelecimento de alguma relação semântica entre as regras de composição propostas e regras de negócio, mas sem uma especificação detalhada de qual seria essa relação. E, ainda, um questionamento sobre a possibilidade de representação de opcionalidade através do recurso de cardinalidade, indicando a não necessidade de aplicação de um recurso notacional próprio para essa classificação.

4.5.3 - Participante P3

O modelo final gerado seguiu os refinamentos especificados, sem acréscimo de conhecimento adicional da abordagem de Linha de Produtos de Software à modelagem. A tarefa foi conduzida utilizando como material de apoio o documento de descrição do meta-modelo e notação (Anexo VI). No entanto, o documento que contemplava as

regras de boa formação (Anexo VII) foi pouco consultado, fato que gerou um conjunto de inconsistências no modelo gerado.

As características especificadas foram corretamente classificadas quanto à categoria existente na notação e sua aplicação dentro da linha de processos. A classificação quanto à opcionalidade foi utilizada, mas o participante apresentou dúvidas de sua aplicação. A representação de opções dentro do domínio não ficou clara para o participante, que por diferentes vezes apresentou a dúvida de representá-las usando o conceito de opcionalidade ou variabilidade. Isso acarretou certas decisões de modelagem, que acabaram sendo influenciadas pelos exemplos modelados e representados nos documentos entregues ao participante. Assim, o entendimento semântico desses conceitos pareceu não ter sido contemplado de forma completa e seu uso poderia ser questionado em alguns pontos da modelagem. O conceito de cardinalidade também foi questionado e uma breve explicação da definição do conceito foi realizada durante a execução da tarefa.

Observou-se a necessidade de especificar neste modelo quando determinados elementos deveriam ser utilizados em um processo instanciado. Desta forma, a abordagem de Linha de Produtos de Software foi modelada como uma prática, que quando adotada acionava a dependência com outros elementos do modelo, inseridos durante a condução da tarefa proposta. Assim, foram incluídas Regras de Composição Inclusivas para atender esse propósito de relacionamentos de dependência entre elementos.

Adicionalmente, o participante considerou que características classificadas como conceitos não deveriam possuir a classificação de opcionalidade, devendo ser sempre mandatórias. Ressaltou ainda, durante a execução da tarefa, a dificuldade de modelagem manual, indicando que o uso de um suporte computacional poderia auxiliar a construção de modelos.

4.5.4 - Participante P4

A modelagem realizada seguiu, de forma objetiva, a descrição dos refinamentos propostos. Devido a pouca análise dos elementos e descrições disponibilizadas, o tempo de realização da tarefa foi relativamente pequeno, quando comparado aos outros estudos. A classificação das características quanto à categoria foi facilmente aplicada. No entanto, os conceitos de opcionalidade e variabilidade não foram completamente entendidos e tiveram a aplicação prejudicada. Isso gerou algumas inconsistências na

modelagem em que a variabilidade foi aplicada de forma equivocada. Pode-se observar que a opcionalidade foi aplicada seguindo os exemplos aplicados no modelo base. Os relacionamentos foram corretamente aplicados e nenhuma regra de composição foi gerada.

Em termos semânticos, o participante indicou a falta da possibilidade de especificação da ordem de execução das tarefas, conceito de sequencialidade fortemente presente em modelagem de processos.

4.6 - ANÁLISE DOS RESULTADOS DO ESTUDO

O questionário apresentado no Anexo VIII foi utilizado para a obtenção da opinião dos participantes. Uma sumarização do resultado obtido através das respostas do questionário é apresentada na Tabela 4.4. Cada item do questionário foi definido como uma afirmação. O participante deveria expressar sua concordância através da escolha entre opções. A escala utilizada variou em 05 níveis: DT – Discordo totalmente; DP – Discordo parcialmente; I – Indiferente; CP – Concordo parcialmente e CT – Concordo totalmente.

Tabela 4.4 - Avaliação do questionário preenchido pelos participantes do estudo

Questão	DT	DP	I	CP	CT
1. O material enviado foi suficiente claro e permitiu a absorção de conhecimento necessário para utilização da notação na execução da tarefa.				2	2
2. O tempo para execução da tarefa foi suficiente.					4
3. O meta-modelo e a notação *OdysseyProcess-FEX* são de fácil entendimento.				1	3
4. Taxonomia e representação gráfica de elementos de processos de software					
4a. Em algum momento existiu a necessidade de representar algum elemento de processo que não foi previsto pela notação.	2	1			1
4b. Algum dos elementos de processos identificados não deveria estar representado neste modelo.	3				1
4c.Os símbolos utilizados para representar os elementos associados aos estereótipos são suficientemente claros, ou seja, permitem uma clara distinção entre os elementos.		1		1	2
4d. A representação de opcionalidade e variabilidade é suficiente explícita.				1	3
4e. O modelo final gerado foi suficientemente satisfatório.		1		1	2
5.Relacionamentos					
5a. Em algum momento você sentiu necessidade de incluir um relacionamento não contemplado.	3				1
5b. Em algum momento as restrições de participação de categorias de elementos em um relacionamento prejudicaram o estabelecimento de algum relacionamento desejado.	4				
6.Suporte Computacional					
6a. Você acredita ser viável a modelagem manual de variabilidades de domínios de processos mais extensos e complexos.	1	2			1
6b. O uso de um suporte computacional para guiar a modelagem, usando notificações que identificam e inviabilizem a ocorrência de violação de alguma restrição imposta pelo meta-modelo seria considerado invasivo.	2	1		1	

De acordo com as respostas obtidas, o meta-modelo e a notação *OdysseyProcess-FEX* foram considerados de fácil entendimento, apesar do aparecimento de dúvidas que foram consideradas pontuais pelos participantes.

Apesar de extensiva, a tarefa não apresentou alto grau de dificuldade para ser executada, com exceção de um dos participantes que não conseguiu restringir as modificações do modelo a um escopo reduzido, indicando a necessidade de grandes mudanças no modelo base fornecido.

Com relação ao acréscimo de algum elemento não contemplado pela notação, foi indicada a necessidade de representação da ordem de precedência entre alguns dos elementos de processo. Uma possibilidade, não abordada durante a execução do estudo, para representar essa informação seria a definição de Regras de Composição que indicassem dependências e mútua exclusividades entre as características precedentes e conseqüentes. No entanto, uma análise mais detalhada de como representar o conceito de temporalidade está sendo realizada em um trabalho de mestrado do Grupo de Reutilização de Software da COPPE/UFRJ. O trabalho tem como finalidade o desenvolvimento de uma abordagem para a Análise de Domínio que incorpora o aspecto temporal à análise de variabilidades na definição de famílias de aplicações cujos leiautes temporais possuem importância substancial em conjunto com suas características (SCHAU, 2010). Esta abordagem poderá ser evoluída para atender aspectos de temporalidade em Linhas de Processos de Software.

O único participante que ressaltou a não concordância com representação de um elemento identificado pela notação, indicou a questão da opcionalidade, enfatizando que o recurso de cardinalidade poderia ser utilizado para representar este conceito. A distinção entre os elementos foi criticada, mesmo que de forma branda, por mais de um participante. No entanto, acredita-se que a visualização do modelo foi prejudicada pela ausência de cores e tamanho reduzido de impressão.

Apesar de graficamente bem aceitos, os conceitos de opcionalidade e variabilidade não foram semanticamente bem entendidos pelos participantes. Através dos resultados das modelagens, observou-se que o propósito e importância da existência e representação desses conceitos para a modelagem de características dentro da abordagem de Linha de Processos de Software não foram capturados.

De modo geral, o entendimento do meta-modelo e da notação e uso de seus elementos foi fortemente realizado baseado nos exemplos fornecidos. Por esse motivo, a

necessidade de mais exemplos na leitura introdutória foi mencionada por mais de um participante, para diferentes conceitos.

Observou-se que as regras de boa formação não foram utilizadas, em consequência da baixa consulta ao documento onde estas estavam listadas. Desta forma, algumas dessas regras foram violadas e acabaram gerando inconsistências no modelo que não foram nem notadas pelos participantes, mas que geram problemas para a reutilização dos elementos de processos de software.

O suporte computacional foi apontado como um recurso auxiliar para contemplar as restrições impostas pelas regras de boa formação e verificação da modelagem. Outro ponto levantado foi como auxílio na modelagem gráfica. Com relação ao mecanismo de suporte à modelagem de notificações, observou-se que houve certa precaução. Sugestões do controle do nível de interferência e formas de sinalização de erros através de cores foram relatadas, assim como a possibilidade de dar ao usuário a opção de trabalhar livremente associada à geração de um relatório final com uma lista das inconsistências geradas durante a modelagem.

4.7 – VALIDADE

Este estudo preliminar de avaliação foi executado com o objetivo de caracterizar a viabilidade de aplicação do meta-modelo e da notação *OdysseyProcess-FEX* na modelagem de Linha de Processos de Software. A análise dos resultados obtidos foi útil para identificar algumas oportunidades de melhoria, benefícios e entraves de aplicação da *OdysseyProcess-FEX*. No entanto, devido às restrições deste estudo, estes resultados não podem ser generalizados.

Apenas quatro participantes realizaram o estudo. Desta forma, o pequeno número de participantes pode ter influenciado os resultados obtidos. A não familiaridade dos participantes com alguns dos principais temas envolvidos no estudo também acaba por contribuir para a impossibilidade de generalização dos resultados. Este fato dificulta o entendimento do objetivo e do contexto em que a *OdysseyProcess-FEX* deve ser utilizada.

Outra limitação é o pouco recurso disponível para exposição do conjunto de informações que constam no meta-modelo e na notação, além do conteúdo envolvido na abordagem de Linha de Processos de Software. Apesar da entrega prévia de uma descrição dos principais elementos da *OdysseyProcess-FEX* (Anexo VI e Anexo VII), esse conjunto de informações foi limitado e não foi devidamente absorvido pelos

participantes. Em alguns casos a leitura antecipada desse conteúdo não foi realizada e em outros, mesmo tendo dedicado certo tempo a esta leitura introdutória, os participantes comunicaram no início do estudo não terem tido condições de absorver todo o conteúdo. Assim, os resultados também podem ter sido influenciados pela falta de tempo de maturação do conhecimento necessário para a realização da tarefa proposta. Uma outra forma de introdução deste conhecimento poderia ter sido aplicada, como por exemplo, uma apresentação mais detalhada e presencial do conteúdo envolvido no estudo.

4.8 – CONSIDERAÇÕES FINAIS

Neste capítulo, foi apresentado um estudo de observação do meta-modelo e da notação *OdysseyProcess-FEX,* que teve como objetivo a modelagem de características de uma Linha de Processos de Software, na etapa de Análise de Domínio. Com os resultados obtidos, alguns indícios puderam ser apresentados para a questão de viabilidade de aplicação da *OdysseyProcess-FEX* para atender este tipo de modelagem. Sua aplicação para modelagem foi bem aceita pela facilidade de entendimento dos principais conceitos e certas semelhanças com outras modelagens conceituais e de processos de software. No entanto, o maior entrave para avaliação foi o pouco conhecimento da abordagem de Linha de Produtos de Software, modelagem de características e, consequentemente, da abordagem de Linha de Processos de Software. Esta limitação acaba por não deixar claro o contexto em que se aplica a modelagem proposta e dificulta o raciocínio necessário para a construção de um modelo que tem por objetivo refletir uma família de processos de software, suas similaridades e variabilidades. Assim, todo o potencial da modelagem não foi explorado nem devidamente aplicado para a realização da tarefa.

O estudo descrito neste capítulo foi um primeiro passo para a avaliação do meta-modelo e da notação *OdysseyProcess-FEX,* propostos neste trabalho. O conjunto de limitações exposto, associado a outros pontos a serem investigados, apresentam a perspectiva de realização de outros estudos, inclusive focando outros aspectos da abordagem de Engenharia de Linha de Processos de Software.

Capítulo V

IMPLEMENTAÇÃO DO META-MODELO E DA NOTAÇÃO *ODYSSEYPROCESS-FEX* NO AMBIENTE ODYSSEY

5.1 – INTRODUÇÃO

A proposta de um meta-modelo e de uma notação para a modelagem de características do domínio de processos de software foi apresentada no Capítulo 3. O objetivo da notação *OdysseyProcess-FEX* é oferecer uma maior gama de recursos para a representação das variabilidades nos artefatos de uma estrutura de reutilização de processos, como uma Linha de Processos de Software. No entanto, para tornar a sua utilização viável, é necessário um ferramental automatizado que apóie a construção de modelos de características e outros artefatos, de diferentes níveis de abstração, a eles relacionados. Os resultados do estudo, descrito no Capítulo 4, reforçam o desenvolvimento de um protótipo para apoiar a aplicação da abordagem proposta.

Uma vez que, na construção de uma Linha de Processos de Software, o modelo de características não é um artefato isolado, é interessante construí-lo e utilizá-lo em um ambiente de reutilização que forneça subsídios necessários à construção e integração dos artefatos envolvidos. Nesse sentido, é apresentado neste capítulo o ambiente de reutilização de software Odyssey (ODYSSEY, 2011), que foi estendido para apoiar a reutilização de processos através de uma Linha de Processos de Software. O ambiente foi adaptado para viabilizar a utilização da notação *OdysseyProcess-FEX*. De forma complementar, um mecanismo de acompanhamento da aplicação correta das regras de boa formação, definidas no meta-modelo, foi implementado com o intuito de verificar a consistência do modelo de características e evitar que erros sejam eventualmente introduzidos e propagados para as próximas etapas existentes na construção e utilização da linha de processos.

Este capítulo está organizado da seguinte maneira: a Seção 5.2 apresenta uma descrição do ambiente Odyssey, como contexto de utilização da notação *OdysseyProcess-FEX*. A Seção 5.3 descreve a implementação e adaptações realizadas, a fim de estruturar o ambiente como suporte inicial a construção de uma Linha de Processos de Software, através da modelagem de características do domínio de processos de software, de acordo com o meta-modelo e a notação *OdysseyProcess-FEX*. A Seção5.4 abrange as considerações finais sobre a implementação realizada nesse trabalho.

5.2 – CONTEXTO DE UTILIZAÇÃO - O AMBIENTE ODYSSEY

O ambiente Odyssey (ODYSSEY, 2011) é uma infraestrutura de reutilização de software baseada em modelos de domínio, que oferece ferramentas de apoio à construção de aplicações de software seguindo abordagens sistemáticas como LPS. Contempla atividades do desenvolvimento de software *para* reutilização, processo de Engenharia de Domínio (ED), e atividades do desenvolvimento *com* reutilização, processo de Engenharia de Aplicação (EA).

Essa infraestrutura foi implementada utilizando-se a linguagem Java (SUN, 2005) e vem sendo evoluída desde 1997 (WERNER *et al.*, 1999). Desde então, várias ferramentas foram introduzidas com o objetivo de apoiar os processos de ED e EA e tornar o ambiente mais completo. Através de uma análise focada na identificação das funcionalidades básicas de um ambiente de reutilização e as funcionalidades que só seriam utilizadas de acordo com a necessidade do usuário, uma versão reestruturada passou a ser disponibilizada como forma de tratar problemas de escalabilidade e desempenho (Tabela 5.1).

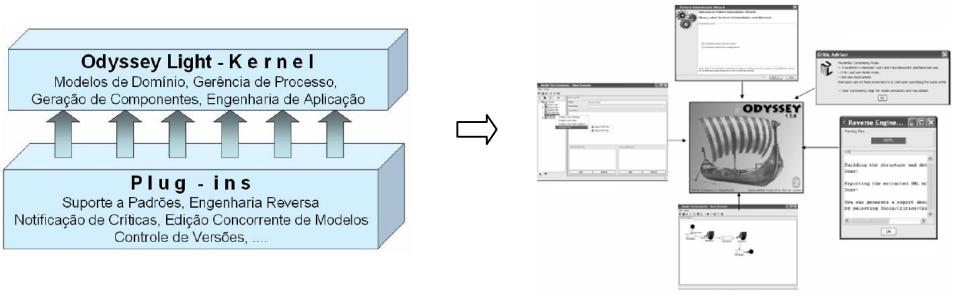


Figura 5.1 - Infraestrutura ambiente Odyssey

Essa versão denominada *Odyssey Light* (MURTA *et al.*, 2004) disponibiliza as funcionalidades básicas através do *kernel* do ambiente e as funcionalidades secundárias através de *plug-ins*, os quais podemos citar: ferramentas para a documentação de componentes (MURTA *et al.*, 2001), especificação e instanciação de arquiteturas específicas de domínios (XAVIER, 2001), apoio à engenharia reversa (VERONESE *et al.*, 2002), de notificação de críticas em modelos UML (DANTAS *et al.*, 2001), de modelagem e execução de processos (MURTA, 2002), para controle de versões de modelos (OLIVEIRA *et al.*, 2004), de transformação de modelos (MAIA *et al.*, 2005), de representação de variabilidades em componentes no contexto de ED (BLOIS, 2006), de apoio à criação de arquiteturas de referência de domínio (VASCONCELOS, 2007), de modelagem de características flexível (TEIXEIRA *et al.*, 2009a), de modelagem de

características para LPS sensíveis ao contexto (FERNANDES, 2009), de apoio à reengenharia de sistemas orientados a objetos para componentes (MOURA, 2009), entre outras.

A estrutura interna do Odyssey apresenta seis níveis de abstração: escopo, características, classes, tipos de negócios, casos de uso e componentes (Figura 5.2). O recorte do domínio, i.e., a seleção dos componentes reutilizáveis, delimita o escopo de informações do domínio que corresponde às especificações da aplicação. Esse recorte é feito selecionando-se escopos, que têm por objetivo situar o domínio em relação ao seu escopo, limites, relacionamentos com outros domínios de aplicação e principais atores envolvidos (BRAGA, 2000), representando sub-domínios, e características, ambos representam níveis de modelo mais abstratos, de acordo com BRAGA (2000) e MILER (2000). Outros artefatos relacionados, como classes, casos de uso e componentes, são selecionados através de rastros, que são ligações entre os elementos dos diferentes níveis de abstração. Esses elementos são apresentados através de uma árvore semântica disponibilizada na parte esquerda do diagramador do ambiente.

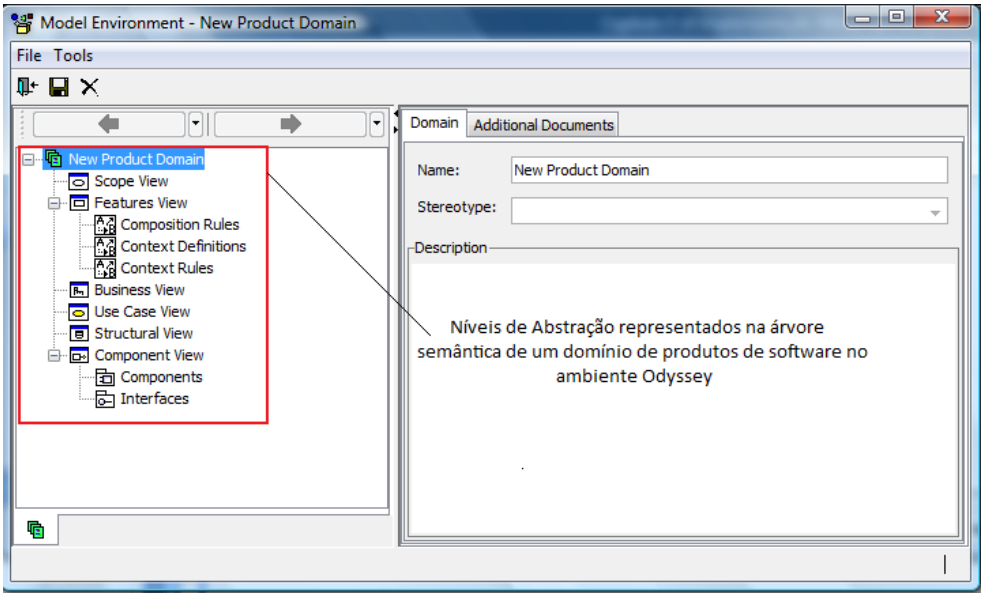


Figura 5.2 - Ambiente de Modelagem de Linhas de Produtos de Software e seus níveis de abstração

Parte da estrutura interna do ambiente Odyssey pode ser visualizada através do diagrama de classes representado na Figura 5.3. A representação interna do Odyssey é organizada hierarquicamente por meio de uma árvore semântica de objetos, onde cada objeto corresponde a um elemento de modelagem chamado *ModeloAbstrato*.

A organização da árvore semântica é feita através de categorias de modelos expressas pelas classes *Modelo* e *Categoria*. Cada modelo é composto por diferentes itens de modelagem, instâncias da classe *ItemModelo*, que representam pacotes, diagramas, nós e ligações específicos à categoria do modelo, que são elementos das subclasses *Diagrama*, *Ligacao* e *No*, todas identificadas na Figura 5.3. A partir de um destes elementos, é possível percorrer a árvore hierarquicamente através das relações entre os objetos, com o intuito de obter informações relativas ao domínio.

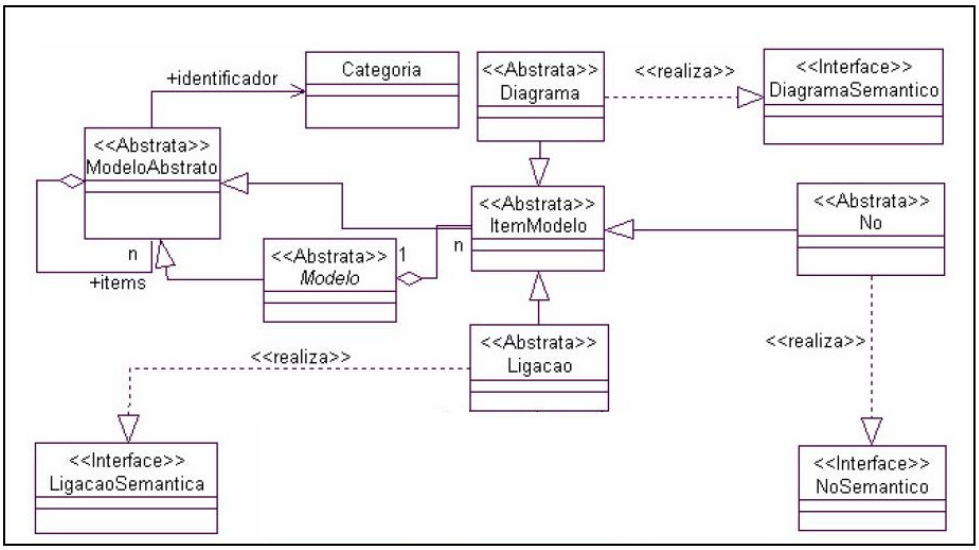


Figura 5.3 - Parte da estrutura interna do ambiente Odyssey (Adaptado de OLIVEIRA, 2006)

O ambiente foi construído utilizando uma notação para modelagem de características, no contexto do processo de Engenharia de Aplicação de Software do ambiente, denominado Odyssey-EA, que foi proposta por MILER (2000). Esta notação foi estendida através do trabalho desenvolvido por OLIVEIRA (2006), que integrou ao ambiente a notação *Odyssey-FEX*. Mais recentemente, o ambiente foi adaptado de forma a disponibilizar diferentes opções de notações para a modelagem de características. Desta forma, em (TEIXEIRA, 2008), foram incorporadas as notações de Czarnecki (CZARNECKI *et al.,* 2004, 2005) e de GOMAA (2004) ao ambiente Odyssey, e também foi possível disponibilizar a transição entre as notações. A estrutura adotada visou criar condições para extensões futuras com a incorporação de novas notações. O elemento que representa características no ambiente corresponde a subclasse de *No* denominada *FeatureBase* e as classes relacionadas ao estabelecimento de perfis (Figura 5.4). A classe abstrata *No* é uma superclasse para elementos como classes, casos de uso, características, componentes, etc. Já a classe *FeatureBase* representa uma base conceitual de compartilhamento de similaridades entre as

diferentes possibilidades de notações para a modelagem de características disponibilizadas pelo ambiente Odyssey.

Cada perfil agrega as particularidades de cada notação. Assim, a classe *FeatureBase* agrega os conceitos comuns às notações, como o conceito de variabilidade e o conceito de opcionalidade. O *PerfilOdysseyFEX* guarda informações relativas apenas a notação *Odyssey-FEX*, como as suas categorias de classificação e propriedades específicas. O *PerfilCzarnecki* guarda informações como valores de cardinalidade, tipo e valor de atributo, particularidades da notação definida em CZARNECKI *et al.* (2004, 2005). Por último, o *PerfilGomaa* guarda as informações específicas referentes à notação definida por GOMAA (2004). Cada perfil possui como extensão os diferentes tipos de características disponíveis na determinada notação.

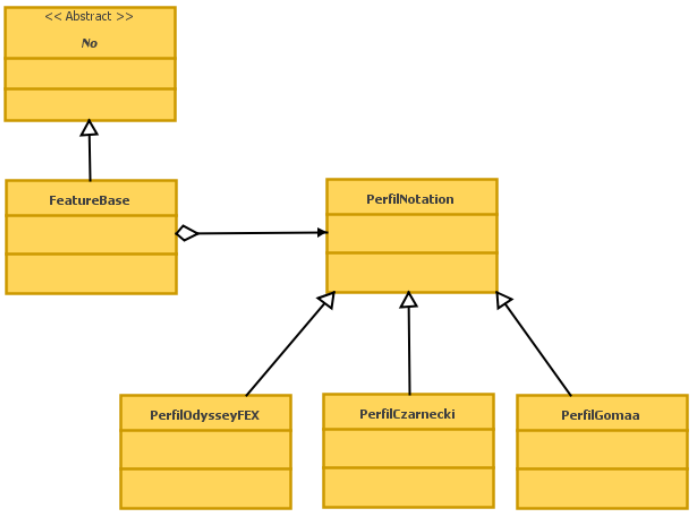


Figura 5.4 - Estrutura de representação do nível de características e do suporte do ambiente às múltiplas representações de diferentes notações de características (TEIXEIRA, 2008)

5.3 – IMPLEMENTAÇÃO DA NOTAÇÃO *ODYSSEYPROCESS-FEX*

O desenvolvimento deste trabalho visou estender o ambiente Odyssey para apoiar a abordagem de Engenharia de Linha de Processos de Software proposta. Desta forma, foram necessárias adaptações para flexibilizar o ambiente a trabalhar tanto com LPS quanto com Linhas de Processos de Software. O maior volume de trabalho se concentrou na disponibilização da notação *OdysseyProcess-FEX* para viabilizar a modelagem de características do domínio de processos de software. Essas adaptações foram realizadas no *kernel* do ambiente.

No ambiente Odyssey, a estrutura de representação de notações de modelagem de características se divide em estrutura Semântica, Léxica e de Apresentação. A estrutura Semântica é responsável pela representação conceitual das notações. As

estruturas Léxica e de Apresentação ficam responsáveis pela representação gráfica, que consiste na visualização das notações. A parte Léxica refere-se à representação gráfica de cada elemento e a de Apresentação se destina às interfaces gráficas de configuração dos elementos semânticos. Desta forma, cada uma dessas partes teve de ser adaptada para atender a modelagem proposta pela notação *OdysseyProcess-FEX*.

Esta seção se dedica a apresentar funcionalidades que foram adaptadas ou incluídas no ambiente Odyssey para apoiar a abordagem de Engenharia de Linha de Processos de Software proposta. Assim, são descritas as possibilidades de configurações das características, de acordo com os conceitos presentes na notação; a verificação da consistência dos modelos criados segundo o meta-modelo *OdysseyProcess-FEX*; e a representação gráfica dos elementos e seus relacionamentos em diagramas.

5.3.1 - Descrição da parte semântica alterada do Odyssey

A estrutura semântica do ambiente Odyssey, apresentada anteriormente, foi refatorada com o objetivo de criar uma infraestrutura flexível com suporte à modelagem de características de LPS e de Linhas de Processos de Software. Desta forma, o maior volume de trabalho concentrou-se no nível de características, denominado "*Features View*".

Para incorporar a notação *OdysseyProcess-FEX* ao ambiente, foi criado um novo perfil, denominado *PerfilOdysseyProcessFEX* como forma de descrever as particularidades desta notação (Figura 5.5).

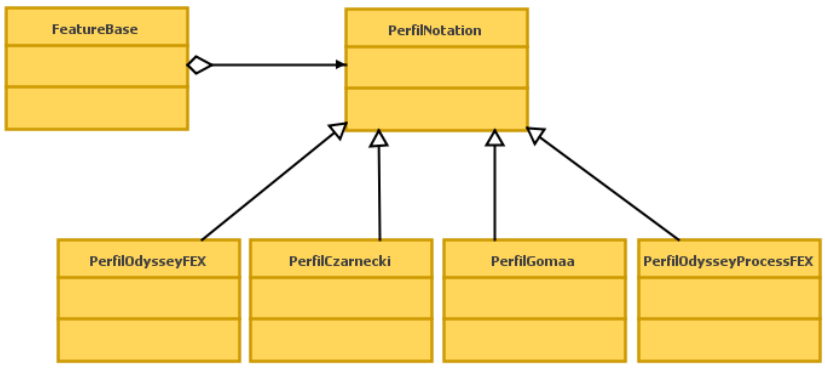


Figura 5.5 - Alteração em parte da estrutura semântica do ambiente Odyssey para incorporação da notação *OdysseyProcess-FEX*

Além disso, os tipos específicos de características definidos pelo meta-modelo foram acrescentados como extensões do perfil criado. Os tipos de características acrescentados correspondem as setes categorias especificadas pelo meta-modelo *OdysseyProcess-FEX*: *Característica Disciplina, Característica Atividade,*

Característica Tarefa, Característica Prática, Característica Conceito, Característica Papel e Característica Produto de Trabalho. Particularidades de cada tipo de categoria são ainda especificados neste nível da estrutura, como podemos observar o atributo propósito das classes *NoProcessFeatureDiscipline, NoProcessFeatureActivity* e *NoProcessFeatureTask*, e o atributo qualificações da classe *NoProcessFeatureRole*. Essa nova estrutura pode ser visualizada na Figura 5.6.

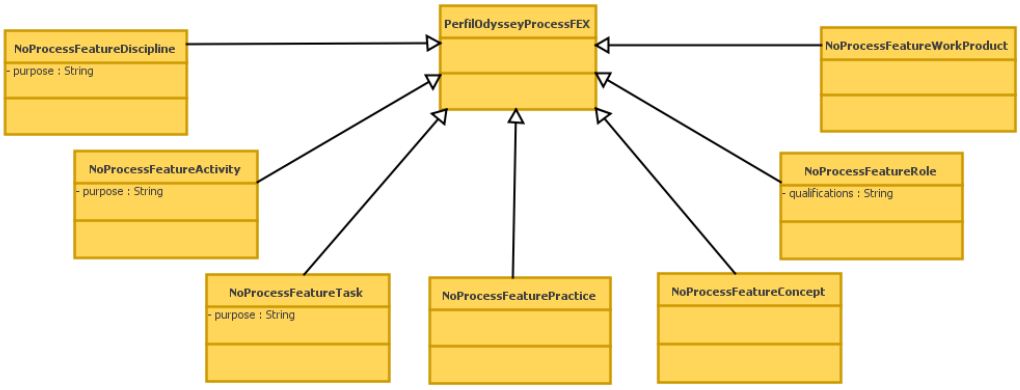


Figura 5.6 - Estrutura semântica alterada do ambiente Odyssey

5.3.2 - Descrição da parte funcional alterada do Odyssey

O ambiente antes estruturado apenas para viabilizar a construção de uma LPS passa a apoiar também a construção de uma Linha de Processos de Software. Desta forma, são disponibilizadas as opções para a criação de domínios de processos de software e domínios de produtos de software, como pode ser visto na Figura 5.7.

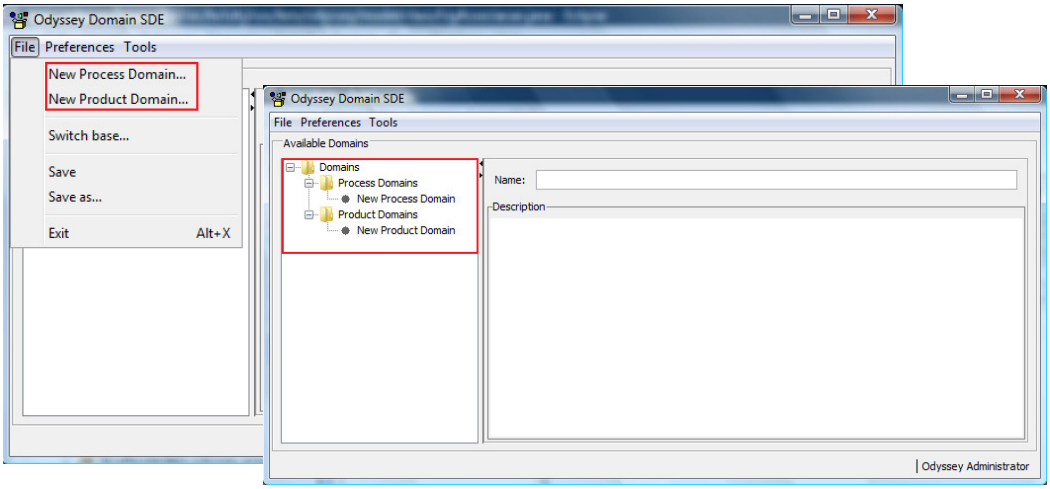


Figura 5.7 - Tela Principal do ambiente Odyssey adaptado

No nível de características, denominado no Odyssey de "*Features View*", encontra-se o maior volume de modificações deste trabalho. Podemos observar que

novas categorias de características foram incluídas e passaram a ser disponibilizadas no ambiente (Figura 5.8).

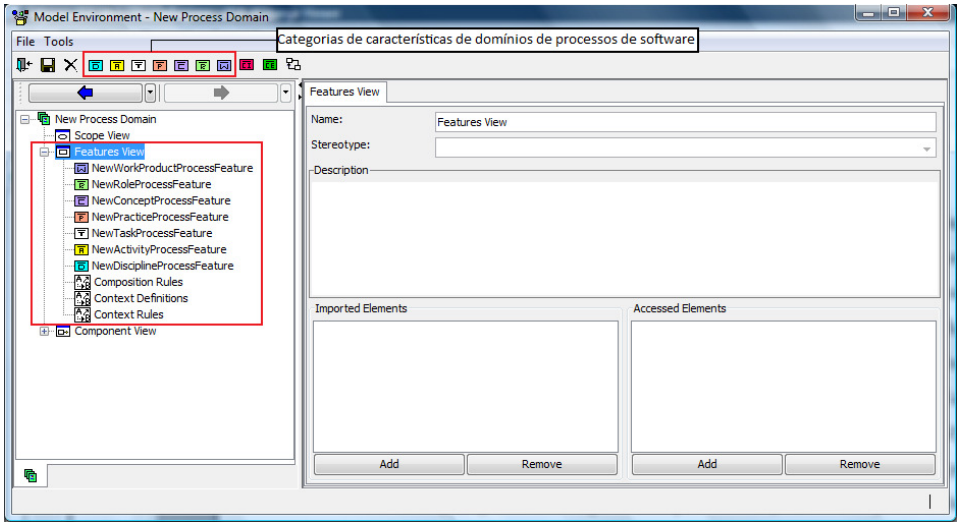


Figura 5.8 - Nível de características de Linha de Processos de Software no ambiente Odyssey

Cada característica apresenta um conjunto de dados que são configuráveis por um painel associado (Figura 5.9). Os campos nome, categoria, descrição, tipo de variabilidade, tipo de opcionalidade e informações adicionais são comuns a todas as categorias de características. O campo de categoria não permite edição, estando de acordo com o tipo de característica previamente criada.

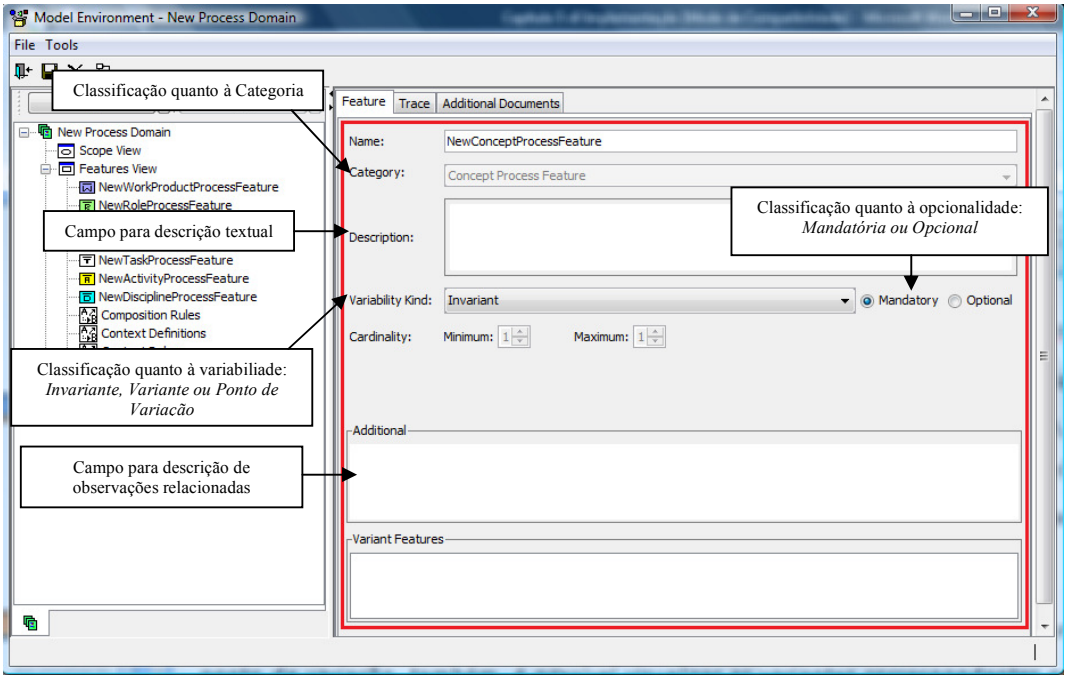


Figura 5.9 - Painel de configuração de característica de processos de software no ambiente Odyssey

Caso a característica seja classificada com tipo de variabilidade correspondente a ponto de variação, também, é possível visualizar as variantes correspondentes no campo "*Variant Features*" (Figura 5.10).

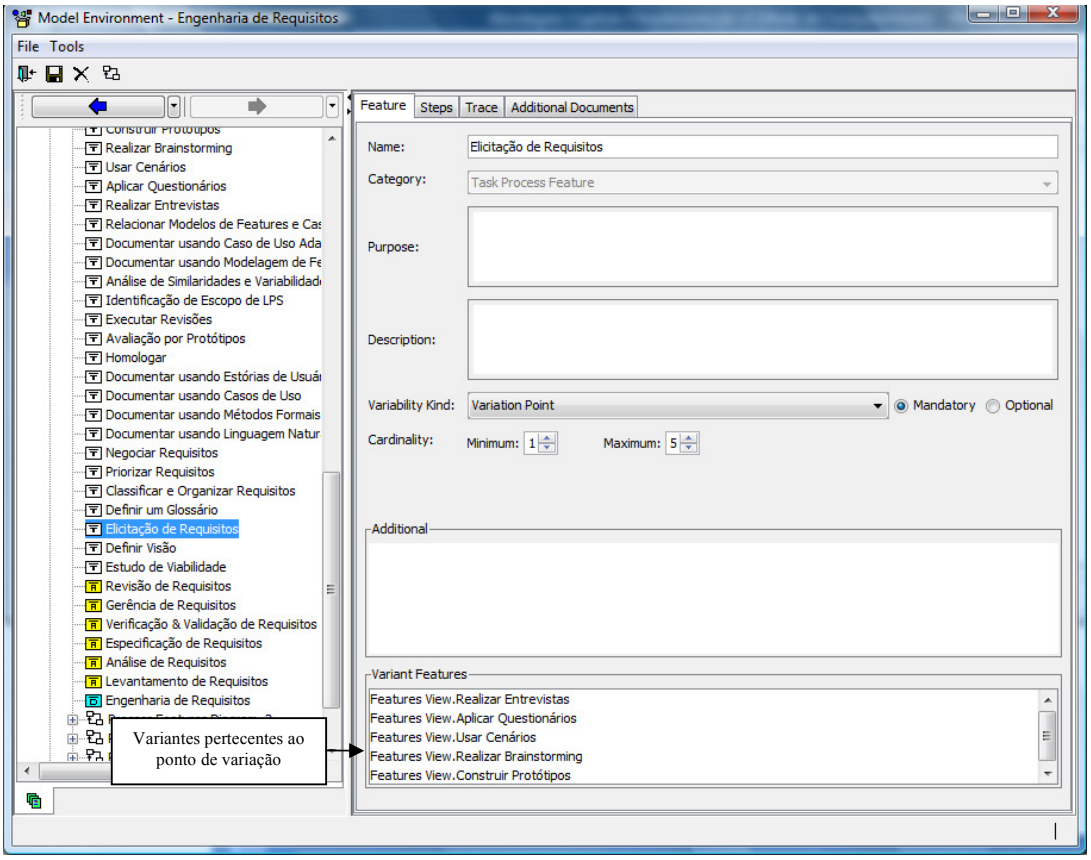


Figura 5.10 - Exemplo de um conjunto de variantes pertecentes a um ponto de variação

Ainda no caso do tipo de variabilidade ser igual a ponto de variação, o campo de cardinalidade também fica disponível para configuração. Neste caso, a classificação quanto à opcionalidade também será impactada pelo valor atribuído à cardinalidade. Desta forma, o tipo de opcionalidade deve estar em conformidade com o valor da cardinalidade mínima, sendo automaticamente atualizado de acordo com a classificação quanto à opcionalidade, mantendo uma coerência entre as informações, como apresentada na Figura 5.11. O valor da cardinalidade máxima igual a um determina a mútua exclusividade das variantes definidas para o ponto de variação.

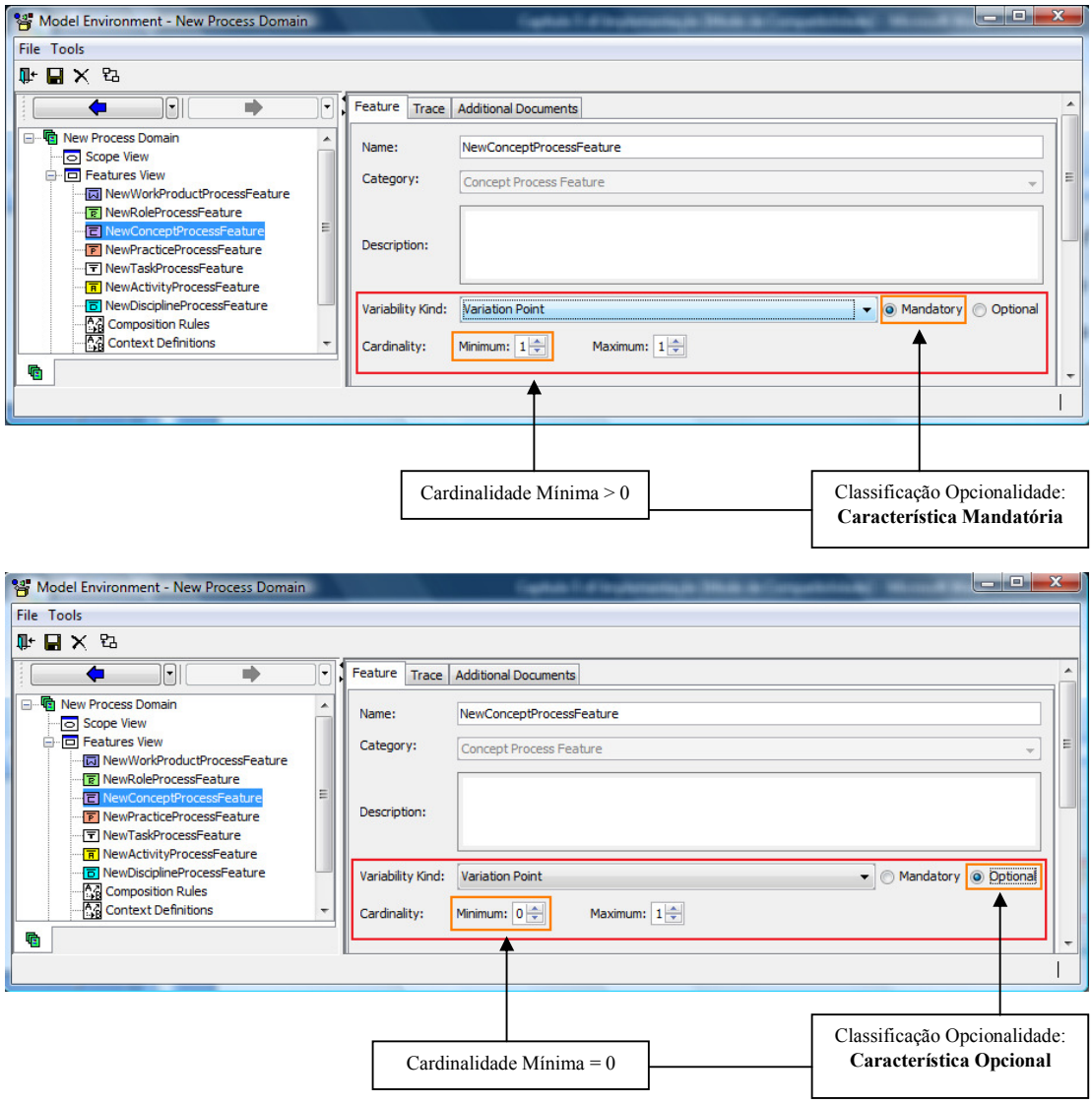


Figura 5.11 - Cardinalidade e classificação de opcionalidade em características classificadas como ponto de variação

Esse painel para configuração de informações de determinada característica apresenta alguns campos específicos a certas categorias de características (Figura 5.12). Por exemplo, podemos citar o campo propósito disponível para edição nas categorias Disciplina, Atividade e Tarefa. O campo qualificações foi disponibilizado apenas para características da categoria Papel.

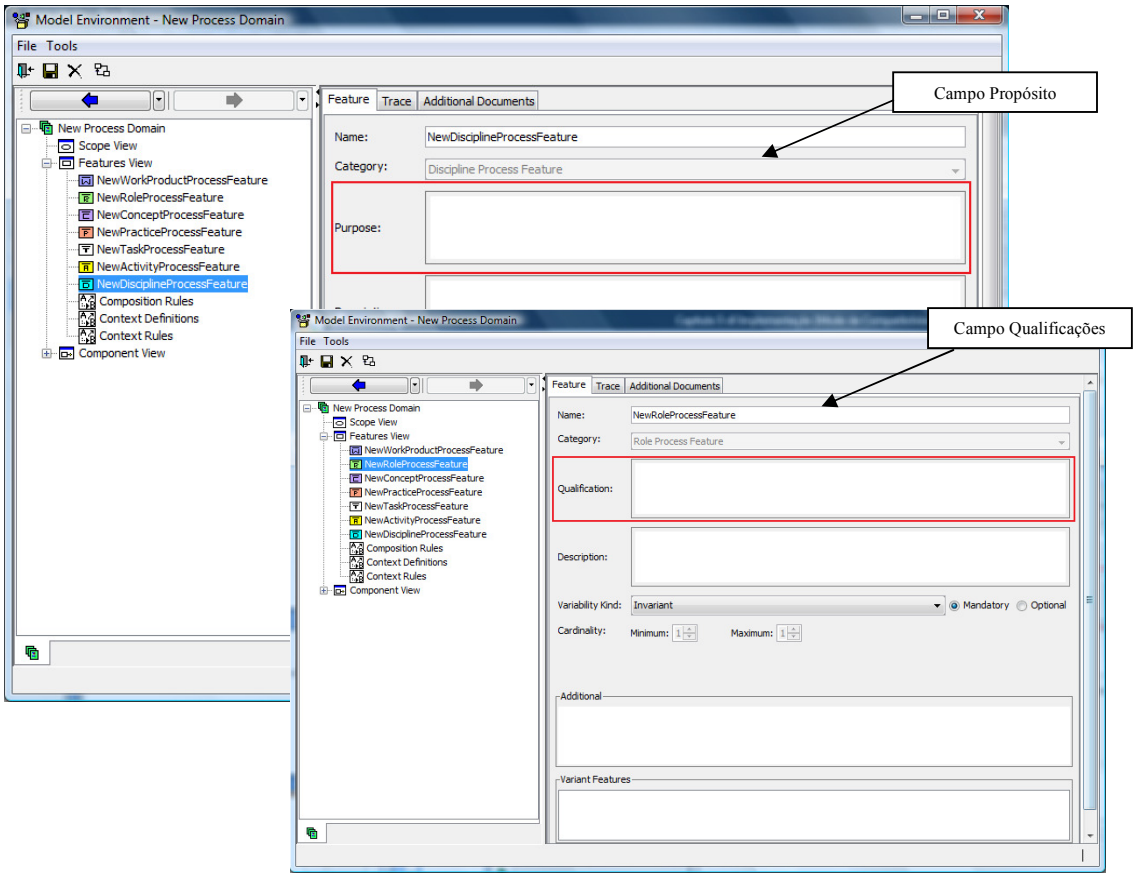


Figura 5.12 - Campos propósito e qualificações no painel de configuração de características de categorias específicas

5.3.2.1 – Diagramador de Características

Além da estrutura semântica já apresentada em seções anteriores, o ambiente possui uma estrutura léxica formada por diagramas específicos a cada modelo representado. Esses modelos são construídos através de uma ferramenta de diagramação, denominada Editor de Diagramas. Para todo item do modelo semântico, existe zero ou mais itens léxicos correspondentes, nós e arestas que formam os desenhos dos diagramas. São os elementos léxicos que determinam a concordância com a representação de cada notação e linguagem adotada no ambiente.

Para cada categoria de diagrama identificada no ambiente Odyssey (e.g., Diagrama de Classes, Diagrama de Componentes, Diagrama de Características, etc.) existe uma representação através de um painel específico. Cada painel apresenta os elementos léxicos pertencentes aquele diagrama. Neste painel, os elementos léxicos sabem se os desenhos estão de acordo com a notação determinada pela especificação associada ao painel. A janela do diagramador de características pode ser visualizada na

Figura 5.13. À esquerda, a árvore de itens semânticos instanciados encontra-se representada. À direita, identificamos o diagrama com o desenho que representa as características (nós) e seus relacionamentos (arestas). A exibição desse diagrama corresponde ao item semântico *Process Features Diagram*, que está selecionado na árvore.

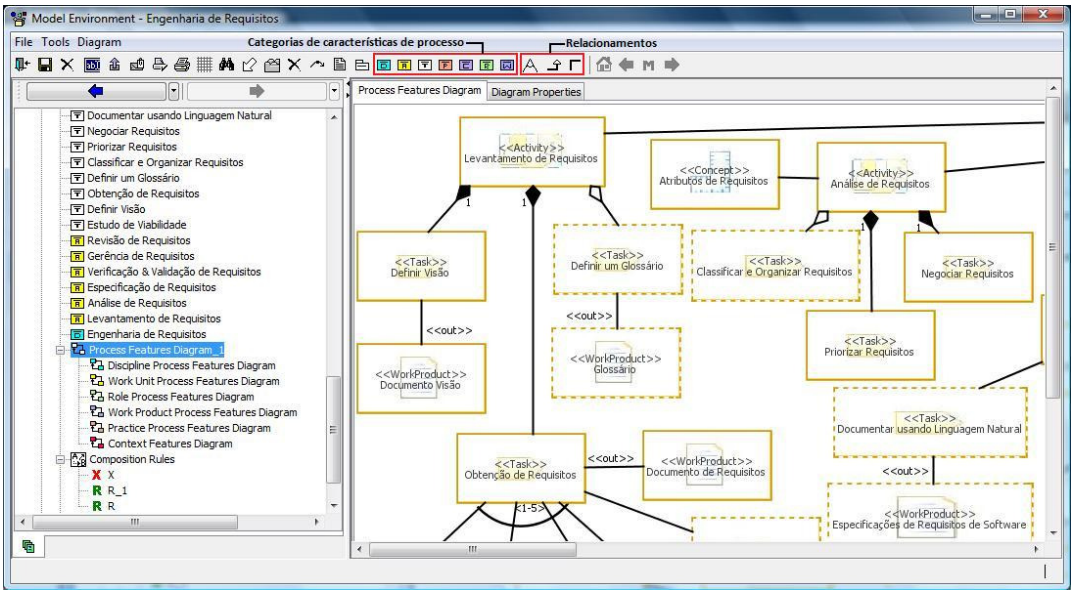


Figura 5.13 - Janela de Diagramação do ambiente Odyssey – visão *Process Feature Diagram*

Na parte superior da janela, uma barra de ferramentas auxilia o usuário no processo de modelagem. Nessa barra, encontram-se as ações necessárias à construção do modelo, tais como as ações relativas à inclusão e remoção de características e das ligações entre elas. Essas ligações, tanto semânticas quanto léxicas, só podem ser manipuladas através do diagramador, já que não possuem representação na árvore semântica do Odyssey. A fim de contemplar a notação *OdysseyProcess-FEX*, foram acrescentadas no diagramador as seguintes ações: criação das diferentes categorias de características contempladas pela notação e dos relacionamentos identificados. Esses itens encontram-se destacados na Figura 5.13.

Vale ainda ressaltar que as informações desse diagrama podem ser manipuladas através de cinco visões diferentes, que correspondem aos sub-diagramas associados (i.e., *Discipline, Work Unit, Role, Work Product* e *Practice Process Feature Diagram*). Para cada um deles, a barra é dinamicamente atualizada segundo o conjunto de elementos que fazem parte da visão selecionada.

5.3.2.2 – Implementação do Mecanismo de Verificação de Regras de Boa Formação do meta-modelo *OdysseyProcess-FEX*

No intuito de viabilizar uma verificação da correta aplicação dos elementos definidos no meta-modelo *OdysseyProcess-FEX,* para a construção de modelos de características dentro do domínio de processos de software, foi implementado neste trabalho um mecanismo que acompanha a atividade de modelagem e intervém na medida que inconsistências são introduzidas no modelo. O mecanismo monitora a inclusão, remoção e edição de elementos no ambiente Odyssey, enviando notificações aos desenvolvedores a cada vez que as regras de boa formação de um modelo são desrespeitadas.

A interação do mecanismo criado ocorre com a notificação ao usuário através de mensagens enviadas no momento da ação que gerou a inconsistência e também de algumas restrições de modelagem diretamente implementadas no diagramador. Na Figura 5.14, podemos ver uma situação de atuação do mecanismo. Neste caso, o usuário é notificado da inconsistência gerada ao tentar associar uma variante mandatória a um ponto de variação opcional. A inconsistência é decorrente do fato que a ausência do ponto de variação em um processo instanciado deve implicar na ausência das variantes a eles associadas. Assim, nenhuma de suas variantes pode ser mandatória.

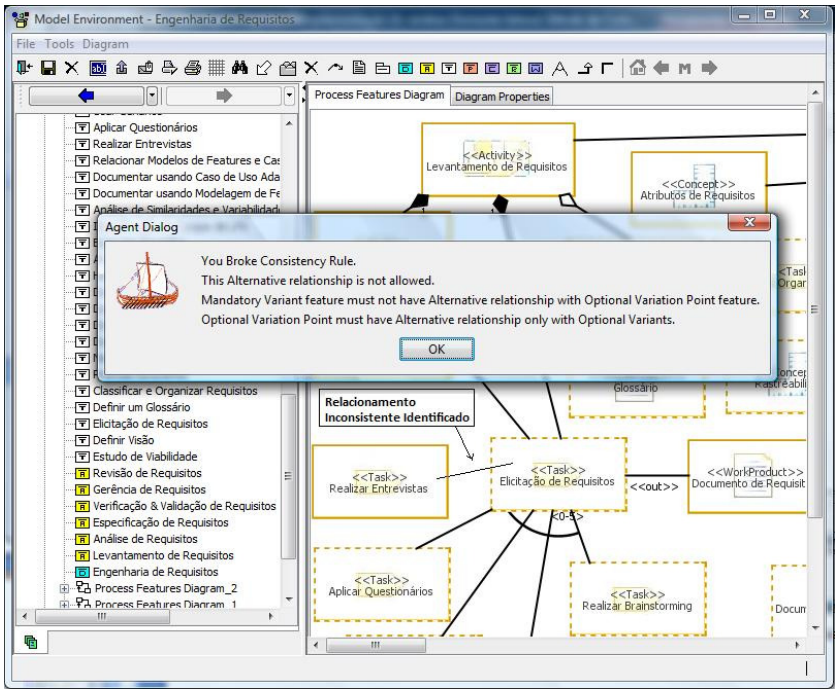


Figura 5.14 - Exemplo de notificação de inconsistência gerada pelo mecanismo de verificação

Neste trabalho, a definição das críticas e regras que são verificadas está amarrada ao código. Como trabalho futuro, a intenção é modificar o mecanismo para que seja incluído em um sistema de críticas que utiliza um arquivo descritor para armazenar as informações sobre as críticas e regras. O Oráculo (DANTAS *et al.,* 2001), mecanismo de críticas, foi desenvolvido dentro do contexto do ambiente Odyssey, e é uma ferramenta auxiliar à ferramenta de diagramação do ambiente para realizar a verificação de consistências dos modelos implementados no Odyssey de forma opcional, a cargo da ativação pelo usuário. O objetivo é realizar uma extensão nesta ferramenta para também contemplar as regras do meta-modelo *OdysseyProcess-FEX*.

O objetivo desta implementação foi oferecer o apoio necessário à detecção de inconsistências em modelos de características construídos com a notação *OdysseyProcess-FEX*. Com isso, pode-se evitar a propagação de erros de modelagem paras as diferentes fases de modelagem e entre os diferentes níveis de abstração. Espera-se alcançar uma modelagem de artefatos do domínio de maneira coerente, aumentando assim a produtividade e a qualidade da definição de processos com reutilização.

5.4 – CONSIDERAÇÕES FINAIS

Neste capítulo, foi descrita a implementação realizada no contexto do ambiente Odyssey, com o intuito de viabilizar a adaptação do ambiente para apoiar a construção inicial de Linhas de Processos de Software. Foram realizadas adaptações em nível de modelo conceitual, bem como no funcionamento do sistema, para o apoio automatizado à utilização da notação *OdysseyProcess-FEX*. Desta forma, o ambiente foi flexibilizado para trabalhar tanto com LPS quanto com Linha de Processos de Software.

A notação *OdysseyProcess-FEX* foi acrescentada ao ambiente de forma a possibilitar o usuário uma modelagem de características de processo de software consistente com o meta-modelo proposto.

Capítulo VI
CONCLUSÃO

6.1 - EPÍLOGO

A abordagem de Linha de Processos de Software surgiu como uma técnica sistemática de reutilização de processos que aplica os conceitos de LPS em processos de software. Entre os benefícios almejados podemos citar o aumento da produtividade da atividade de definição de processos; o aumento da qualidade e da adequação dos processos gerados; o aumento do potencial de reutilização através da representação de variabilidades e redução dos riscos de uma definição inadequada de processo. Uma das questões essenciais que deve ser considerada quando se constrói artefatos de processos de software reutilizáveis é tratar as variabilidades presentes em uma Linha de Processos de Software, assim como temos a atividade de tratamento de variabilidades fortemente presente em LPS. Para tanto, existe a necessidade de representação dessas variabilidades em modelos que representam uma família de processos ao longo de todo o ciclo de desenvolvimento de software. A modelagem de características, fortemente utilizada em LPS, busca capturar as similaridades e variabilidades através de um modelo de alto nível de abstração, uniformizando o entendimento entre todos os envolvidos no processo de desenvolvimento.

A partir da análise de representações de variabilidades de uma Linha de Processos existentes na literatura, percebeu-se algumas deficiências nessas propostas, no que diz respeito à representação de conceitos relacionados à variabilidade. A maioria das representações propostas foca no nível de projeto e apresenta limitações que podem ocasionar uma representação inadequada da variabilidade, gerando uma modelagem incorreta que será posteriormente reutilizada. São representações incompletas e, por vezes, não explícitas ou ambíguas.

Neste trabalho de pesquisa foi apresentada uma abordagem sistemática de Engenharia de Linha de Processos de Software, com foco na fase de Análise do Domínio de Processos de Software e na representação adequada de variabilidades em artefatos reutilizáveis inerentes ao domínio de processos de software. Com o objetivo de minimizar os problemas ocasionados pela representação incompleta de variabilidades de uma família de processos de software, um meta-modelo e uma notação, denominados *OdysseyProcess-FEX* foram propostos. A *OdysseyProcess-FEX* visa apoiar a modelagem de variabilidades de uma Linha de Processos de Software através de uma

modelagem de características voltada ao domínio de processos de software.

Neste último capítulo, são destacadas as principais contribuições deste trabalho, na Seção 6.2. A Seção 6.3 apresenta algumas limitações identificadas e por fim, na Seção 6.4, são discutidas algumas possibilidades de trabalhos futuros.

6.2 - CONTRIBUIÇÕES

Este trabalho teve como objetivo principal propor uma abordagem para definição de Linhas de Processos de Software através da representação de variabilidades de famílias de processos em um modelo de características. Entre as principais contribuições podemos destacar:

- Definição de uma representação de variabilidades através de um modelo de características inerente ao domínio de processos de software, que envolve: 1) um meta-modelo, que formaliza a semântica dos conceitos envolvidos nessa modelagem; e 2) uma notação para a representação gráfica, em um modelo de características, dos conceitos definidos pelo meta-modelo;
- Avaliação preliminar da viabilidade de aplicação do meta-modelo e da notação *OdysseyProcess-FEX* para atividade de construção de Linhas de Processos de Software. Os resultados obtidos apontaram pontos de possíveis melhorias e poderão servir como base para o planejamento de uma avaliação mais completa; e
- Desenvolvimento de um protótipo que implementa a abordagem proposta no contexto do ambiente Odyssey. Um ferramental automatizado foi disponibilizado para apoiar a construção de modelos de características de processo de software consistente com o meta-modelo proposto.

Podemos ainda ressaltar as seguintes contribuições secundárias:

- Estudo das áreas de Reutilização de Processos de Software, Linha de Processos de Software e Modelagem de Variabilidades, buscando identificar oportunidades de pesquisa existentes;
- Identificação de um conjunto de requisitos para a representação de variabilidades em modelos de características dentro do domínio de processo de software;
- Identificação dos elementos necessários para possibilitar a representação em modelos de características de linhas de processos de software;

- Definição de um conjunto de regras para a verificação de consistência intra-modelo, evitando que erros sejam propagados; e
- Participação na definição de uma abordagem sistemática de Engenharia de Linha de Processos de Software, com foco na fase de Análise do Domínio de Processos de Software. Essa abordagem consiste na especificação das etapas que constituem o ciclo de desenvolvimento e aplicação de uma Linha de Processos de Software e seus respectivos produtos.

6.3 – LIMITAÇÕES

Algumas limitações foram identificadas a partir de uma análise crítica da abordagem proposta e do protótipo desenvolvido:

- O meta-modelo e a notação *OdysseyProcess-FEX* propostos tratam apenas da representação de elementos de processos de software reutilizáveis em modelos de características, que é um modelo de alto nível de abstração. No entanto, é necessário que esse conhecimento seja representado nos demais artefatos que pertencem a uma Linha de Processos de Software, como o modelo de componentes de processos;
- Apesar de ter sido realizado um estudo de observação para avaliar a aplicabilidade do meta-modelo e da notação *OdysseyProcess-FEX* na construção de Linha de Processos de Software, este pode ser considerado apenas uma avaliação inicial devido a seu escopo reduzido e parâmetros de avaliação apenas de caráter observatório. Desta forma, não é possível confirmar se a representação proposta é satisfatória para a modelagem de características no domínio de processos de software. Com os resultados obtidos há indicações que melhorias podem ser acrescentadas com a inclusão de alguns outros elementos à notação; e
- O suporte computacional construído não foi avaliado. Desta forma, não há indicações da sua viabilidade de uso e necessidades de melhorias. Além disso, a parte de verificação de regras de boa formação deve ser adaptada para permitir a sua aplicação através do mecanismo de críticas existente no ambiente Odyssey, denominado Oráculo (DANTAS *et al*., 2001). Dessa maneira, o mecanismo se tornaria menos intrusivo e mais flexível a alterações no conjunto de regras proposto.

6.4 – TRABALHOS FUTUROS

Um conjunto de oportunidades de melhoria, tanto na abordagem proposta quanto do protótipo desenvolvido, foi identificado, assim como novas oportunidades de pesquisa. Como possibilidades de trabalhos futuros, podemos enumerar:

- Aplicação da notação *OdysseyProcess-FEX* e do protótipo no desenvolvimento de Linhas de Processos de Software reais e de maior porte. Dessa forma, será possível identificar oportunidades de evolução e melhoria da abordagem e das ferramentas propostas;
- Planejamento e execução de estudos de avaliação mais completos, que envolvam a abordagem proposta como um todo e a utilização do protótipo;
- Representação e mapeamento dos elementos definidos pelo meta-modelo *OdysseyProcess-FEX* para níveis de abstração mais baixos como, por exemplo, o modelo de componentes de processos de software. A identificação de todos os artefatos envolvidos na construção e aplicação de uma Linha de Processos de Software deve ser confirmada, com estudos mais detalhados de todo o ciclo de vida do domínio de processos de software, para que representações coerentes de variabilidades nos diversos artefatos sejam propostas e heurísticas de mapeamentos entre estes sejam estabelecidas, de forma a manter a consistência entre as diferentes representações da linha;
- Construção de uma arquitetura de Linha de Processos de Software que suporte o conceito de adaptabilidade dinâmica contextual. Desta forma, uma arquitetura de componentes de processo deve ser gerada e organizada de forma a viabilizar a adaptação, em tempo de execução, do processo instanciado a partir da linha. Trabalhos realizados na área de LPS (TEIXEIRA, 2009, TEIXEIRA *et al*., 2009b) e na área de sensibilidade ao contexto em LPS (FERNANDES *et al*., 2010, MARINHO *et al.,* 2010) podem ser utilizados como base para a realização deste trabalho. Além disso, a abordagem de Engenharia de Linha de Processos de Software proposta precisa ser associada a uma gestão de contexto, trabalho de pesquisa de doutorado de NUNES (NUNES *et al.,* 2010). Desta forma, essa estrutura deve fornecer mecanismos apropriados para guiar adaptações e promover a evolução da Linha de Processos de Software;

- Analisar a possibilidade de acoplamento de uma máquina de processos que permita a execução de processos de software a serem instanciados a partir do recorte de uma Linha de Processos de Software definida. Para isso, especificações dos componentes arquiteturais de processos de software em representações executáveis devem ser estabelecidas; e
- Estabelecimento das etapas envolvidas no processo de Engenharia de Aplicação de Processos de Software, com o estabelecimento de mecanismos que direcionem os sucessivos recortes envolvidos na instanciação de um processo de software específico de projeto, com garantias de consistência e de validade.

Referências Bibliográficas

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.

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.

ARAUJO, R.; BORGES, M. R. S., 2001, "Sistemas de Workflow". In: *Jornadas de Atualização em Informática (JAI) – Congresso da Sociedade Brasileira de Computação (SBC)*, pp. 1-17, Fortaleza, Ceará, Brasil.

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". *Software Process: Improvement and Practice*, v. 14, Issue 3, pp. 181-197, New York, NY, USA.

ATKINSON, C., BAYER, J., BUNSE, C., *et al*., 2002, *Component-based product line engineering with UML*, Boston, Addison-Wesley Longman Publishing Co., Inc.

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", *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.

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.

BECK, K., 1999, *Extreme Programming Explained: Embrace Change*. Boston, MA, USA, Addison-Wesley.

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. D. A., 2006, "Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias". *Simpósio Brasileiro de Qualidade de Software (SBQS)*, pp. 72-86, Vila Velha, Espírito Santo, Brasil.

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.

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.

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

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, RJ, Brasil.

CECHTICKY, V., PASETTI, A., ROHLIK, O., *et al*., 2004, "XML-based feature modelling". In: *Proceedings of the 8th International Conference on Software Reuse: Methods, Techniques and Tools – ICSR8*, v. 3107, pp. 101–114, Madrid, Spain, July.

CHEN, L.; BABAR, M. A.; ALI, N., 2009, "Variability management in software product lines: A systematic review". In: *Proceedings of the 13th International Software Product Lines Conference (SPLC)*, pp. 81-90, San Francisco, CA, USA.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S., 2006, *CMMI: Guidelines for Process Integration and Product Improvement.* 2 ed. Boston, MA, 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.

CUGOLA, G.; GHEZZI, C., 1998, "Software processes: A retrospective and a path to the future", *Journal of Software Process Improvement and Practice (SPIP),* v. 4, pp. 101-123.

CZARNECKI, K.; EISENECKER, U., 2000, *Generative Programming: Methods, Tools, and Applications.*Addison-Wesley, New York, NY, USA.

CZARNECKI, K., HELSEN, S., EISENECKER, U., 2004, "Staged configuration using feature models". In: *Software Product Lines: Third International Conference, SPLC 2004, Proceedings*, v. 3154, pp. 266–283, Boston, MA, USA.

CZARNECKI, K., HELSEN, S., EISENECKER, U.W., 2005, "Formalizing cardinality-based feature models and their specialization", *Software Process: Improvement and Practice*, v. 10, n. 1 (March), pp. 7-29.

DANTAS, A.R., CORREA, A.L., WERNER, C.M.L., 2001, "Oráculo: Um Sistema de Críticas para a UML". In: *XV Simposio Brasileiro de Engenharia de Software - SBES, Caderno de Ferramentas*, pp. 398-403, Rio de Janeiro, RJ , Brasil.

DANTAS, A.R., VERONESE, G.O., CORREA, A.L., *et al*., 2002, "Suporte a Padrões no Projeto de Software". In: *XVI Simpósio Brasileiro de Engenharia de Software*, pp. 450-455, Gramado, RS, Brasil, Outubro.

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

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.

FERNANDES, P., 2009, *UbiFEX: Uma Abordagem para Modelagem de Características de Linha de Produtos de Software Sensíveis ao Contexto*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

FERNANDES, P., TEIXEIRA, E.N., WERNER, C., 2010, "An Approach for Feature Modeling of Context-Aware Software Product Line", *Journal of Universal Computer Science (Print)*.

FEI DAI; TONG LI, 2007, "Tailoring Software Evolution Process". In: *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD)*, pp. 782-787, Nagoya, Japan.

FRAKES, W.B., KYO KANG, K., 2005, "Software Reuse Research: Status and Future", *Journal of IEEE Transactions on Software Engineering*, v. 31, n.7, July.

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

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.

GARG, A., CRITCHLOW, M., CHEN, P., *et al*., 2003, "An Environment for Managing Evolving Product Line Architectures". In: *Proceedings of International Conference on Software Maintenance*, pp. 358-369, Amsterdam, The Netherlands.

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

GOMAA, H., SHIN, M.E., 2004, "A Multiple-View Meta-modeling Approach for Variability Management in Software Product Lines". In *Proceedings of the 8th International Conference on Software Reuse: Methods, Techniques and Tools – ICSR8*, v. 3107, pp. 274-285, Madrid, Spain, July.

GRISS, M.L., FAVARO, J., D'ALESSANDRO, M., 1998, "Integrating Feature Modeling with the RSEB". In: *Proceedings of 5th International Conference on Softwre Reuse - ICSR5*, pp. 76-85, Victoria, British Columbia, Canada.

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.

IBM, 2009. *Rational Unified Process (RUP)*. Disponível em: http://www-01.ibm.com/software/awdtools/rup/. Acesso em: 12 Abr 2010.

ISO/IEC, 2007, *ISO/IEC 15939: Software Engineering – Software Measurement Process.* Geneve.

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.

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 the 10th International Software Process Workshop,* pp. 8-11, Dijon, France, June.

KRUEGER, C.W., 1992, "Software Reuse", *Journal of ACM Computing Surveys*, v.24, n.2 (June), pp. 131-183.

LEE, K., KANG, K.C., LEE, J., 2002, "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering". In: *Proceedings of the 7th International Conference Software Reuse: Methods, Techniques, and Tools, ICSR-7*, pp. 62 - 77, Austin, TX, USA, April.

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.

LOPES, M.A.M., MANGAN, M.A.S., WERNER, C.M.L., 2004, "MAIS: Uma Ferramenta de Percepção para apoiar a Edição Concorrente de Modelos de Análise e Projeto". In: *XVIII Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, pp. 61-66, Brasília, DF, Brasil, Outubro.

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.

MAIA, N.E.N., BLOIS, A.P.B., WERNER, C.M., 2005, "Odyssey-MDA: Uma Ferramenta para Transformação de Modelos UML". In: *XIX Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas*, Uberlândia, MG, Brasil, Outubro.

MARINHO, F., LIMA, F., FERREIRA FILHO, J., ROCHA, L., MAIA, M., VIANA, W., ANDRADE, R., TEIXEIRA, E. N., WERNER, C., 2010, "Software Product Line for the Mobile and Context-Aware Applications Domain". In: *14th*

International Software Product Line Conference, v. 1, p. 346-360, Ilha Jeju, South Korea.

MARTÍNEZ-RUIZ, T., GARCÍA, F. AND PIATTINI, M., 2008, "Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms". In: *Software Engineering Research, Management & Applications*, Vol. SCI 150 (Ed, Lee, R.) Springer Verlag. Praga, Czech Republic, pp. 115-130.

MASSEN, T., LICHTER, H., 2002, "Modeling Variability by UML Use Case Diagrams". In: *Proceedings REPL02 - International Workshop on Requirements Engineering for Product Lines*, pp. 19-31, Essen, Germany, September.

MASSEN, T.; LICHTER, H., 2004, "Deficiencies in Feature Models". In: *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, pp. 59-72, Boston, MA, USA.

MATULA, M., 2003, "NetBeans Metadata Repository". In: http://mdr.netbeans.org/MDR-whitepaper.pdf.

MILER, N., 2000, *A Engenharia de Aplicações no Contexto da Reutilização baseada em Modelos de Domínio*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

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

MOURA, A.M., 2009, *ROOSC: Uma Abordagem de Reengenharia de Sistemas Orientados a Objetos para Componentes Baseada em Métricas.* Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

MURTA, L.G.P., BARROS, M.O., WERNER, C.M.L., 2001, "FrameDoc: Um Framework para a Documentação de Componentes Reutilizáveis". In: *IV International Symposium on Knowledge Management/Document Management (ISKM/DM'2001)*, pp. 241-259, Curitiba, PR, Brasil, Agosto.

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

MURTA, L.G.P., VASCONCELOS, A.P.V., BLOIS, A.P.T.B.*, et al.*, 2004, "Run-time Variability through Component Dynamic Loading". In: *XVIII Simpósio Brasileiro de Engenharia de Software. Caderno de Ferramentas*, pp. 67-72, Brasília, DF, Brasil, Outubro.

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.

ODYSSEY, 2011, "Projeto Odyssey", *In: http://reuse.cos.ufrj.br/odyssey*

OLIVEIRA, H., MURTA, L.G.P., WERNER, C.M.L., 2004, "Odyssey-VCS: Um Sistema de Controle de Versões Para Modelos Baseados no MOF". In: *XVIII*

Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, pp. 85-90, Brasília, DF, Brasil, Outubro.

OLIVEIRA, R.F., BLOIS, A.P.T.B., VASCONCELOS, A.P.V., *et al.*, 2005, "Metamodelo de Características da Notação Odyssey-FEX - Descrição de Classes", COPPE/UFRJ, Projeto Reuse - Relatório Técnico 2/2005. Disponível em: 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.

OMG, 2008, "Software Process Engineering Metamodel". Disponível em: http://www.omg.org/technology/documents/formal/spem.htm. Acessado em: Mar 2010.

OSTERWEIL, L., 1987, "Software Processes Are Software Too". In: *Proceedings of the 9th International Conference on Software Engineering,* pp. 2-13, Monterey, USA.

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.

PRESSMAN, R. S., 2001, *Software Engineering: A Practitioner's Approach*. 5 ed., McGraw-Hill.

RAMAN, S., 2000, "It is software process, stupid: next millennium software quality key", Aerospace and Electronic Systems Magazine, IEEE, v. 15, n. 6, pp. 33-37.

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.

RIEBISCH, M., BÖLLERT, K., STREITFERDT, D., *et al*., 2002, "Extending Feature Diagrams with UML Multiplicities". In: *Proceedings of 6th Conference on Integrated Design & Process Technology*, pp. 1-7, Pasadena, California, USA, June.

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

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.

SCHAU, W., 2010, "Uma Abordagem para Análise de Variabilidade de Software com Características Distribuídas no Tempo". In: 15º Workshop de Teses e Dissertações em Engenharia de Software (WTES'2010), pp. 13-18, Salvador, Brasil.

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 SIGSOFT Software Engineering Notes, v. 26, n. 5 (September), pp. 288-296.

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.

SIMOS, M., ANTHONY, J., 1998, "Weaving the Model Web: A Multi-Modeling Approach to Concepts and Features in Domain Engineering". In: *Proceedings of 5th International Conference of Software Reuse (ICSR-5)*, pp. 94-102, Victoria, British Columbia, June.

SOFTEX, 2009, *Melhoria de Processo do Software Brasileiro – Guia Geral*, Modelo de Qualidade Versão 2.0 Disponível em*: http://www.softex.br.*

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

SUN, 2005, "*Java Technology*". In: *http://www.java.sun.com*.

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., 2008, *Flexibilização para Representação de Características no Ambiente Odyssey*. Projeto Final, UFRJ/IM.

TEIXEIRA, E. N., 2009, "Uma Abordagem para Geração de uma Arquitetura de Linha de Produtos de Software Dinâmica". In: *14º Workshop de Teses e Dissertações em Engenharia de Software,* Fortaleza, Brasil.

TEIXEIRA, E. N., VASCONCELOS, A., WERNER, C., 2009a, "An Approach to Support a Flexible Feature Modeling". In: *III Simpósio Brasileiros de Componentes, Arquiteturas e Reutilização de Software (SBCARS),* p. 81-94, Natal, RN.

TEIXEIRA, E. N., FERNANDES, P., WERNER, C., 2009b, "Uma Proposta para Geração de uma Arquitetura de Linha de Produtos de Software Dinâmica". In: *I Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)*, p. 1139-1144, Bento Gonçalves, RS.

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.

VERONESE, G.O., NETTO, F.J., CORREA, A., *et al*., 2002, "ARES – Uma Ferramenta de Engenharia Reversa Java-UML". In: *XVI Simpósio Brasileiro de Engenharia de Software - Caderno de Ferramentas*, pp. 347-352, Gramado, RS, Outubro.

WASHIZAKI, H., 2006, "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.

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.

WISE, A., 2006, *Little-JIL 1.5 Language Report*. Department of Computer Science, University of Massachusetts, Amherst, MA.

WOHLIN, C., RUNESON, P., HÖST, M., OHLSSON, M. C., REGNELL, B.,WESSLÉN, A., 2000, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers.

XAVIER, J.R., 2001, *Criação e Instanciação de Arquiteturas de Software Específicas de Domínio no Contexto de uma Infra-Estrutura de Reutilização*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

Anexo I

ODYSSEYPROCESS-FEX: PACOTE PRINCIPAL

O pacote Principal (*Core*) do meta-modelo *OdysseyProcess-FEX* (Figura 1) é constituído por características, suas diversas categorias e atributos específicos. No total, é composto por sete categorias de características: *Característica Disciplina, Característica Atividade, Característica Tarefa, Característica Prática, Característica Conceito, Característica Papel e Característica Produto de Trabalho*.

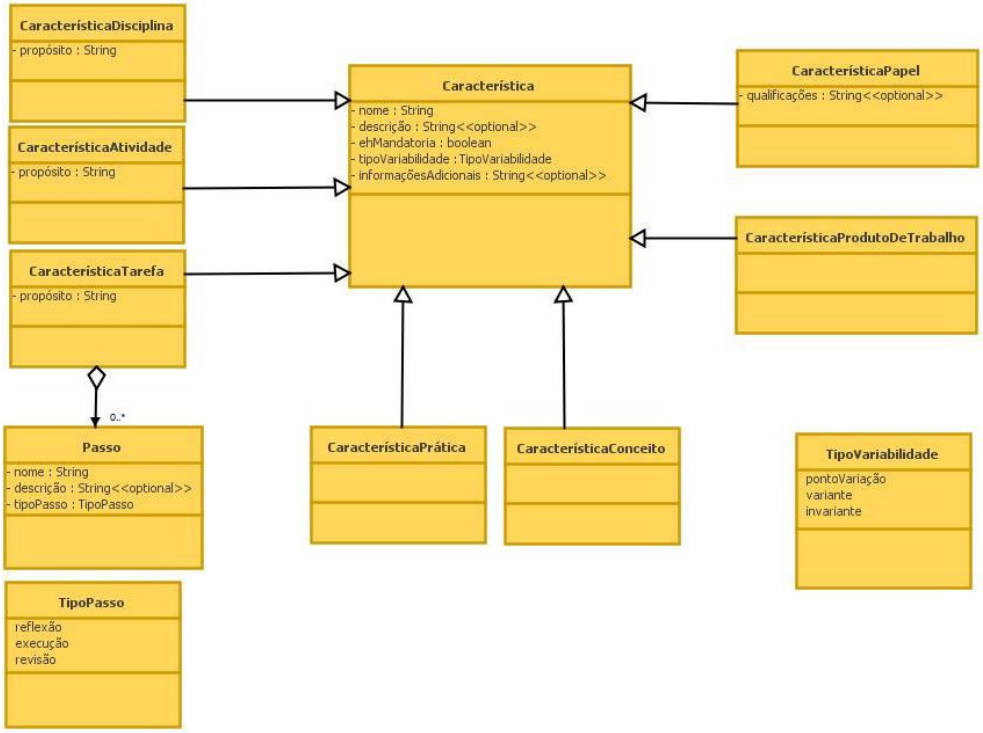


Figura 1 - Meta-modelo *OdysseyProcess-FEX*: Pacote Principal

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. Característica

Descrição

Representa elementos do domínio de processos de software.

Hierarquia

Subclasses: *CaracterísticaDisciplina, CaracterísticaAtividade, CaracterísticaTarefa, CaracterísticaPrática, CaracterísticaConceito, CaracterísticaPapel, CaracterísticaProdutodeTrabalho.*

Atributos

- **nome**: String[1] – atributo que define o nome da característica.
- **descrição**: String[0..1] – atributo que permite uma descrição textual sobre a característica.
- **ehMandatoria**: 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: *true*.
- **tipoVariabilidade**: TipoVariabilidade[1] – atributo que indica o tipo de variabilidade que a característica apresenta. Pode assumir os valores "Invariante", "Variante" e "Ponto de Variação". O tipo de variabilidade é representado pela lista enumerada *TipoVariabilidade*. Default: *Invariante*.
- **informaçõesAdicionais**: String[0..1] – atributo que permite a descrição de observações relacionadas à característica.

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 (Figura2). Desta forma, uma característica pode receber uma classificação quanto à opcionalidade e outra classificação complementar quanto à variabilidade.

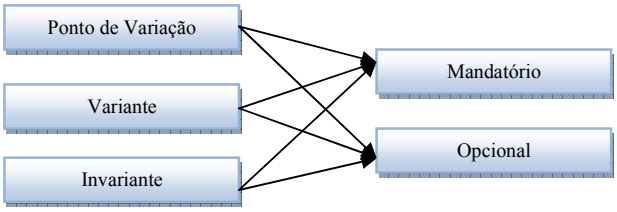


Figura2 - Classificação ortogonal das características

No domínio exemplo de Engenharia de Requisitos, as características *Engenharia de Requisitos, Levantamento de Requisitos, Análise de Requisitos* ou *Especificação de Requisitos* são exemplos de características classificadas como mandatórias. As características *Definir um Glossário, Classificar e Organizar Requisitos, Glossário* ou *Modelo de Caso de Uso* são exemplos de características opcionais.

b. TipoVariabilidade

Descrição

TipoVariabilidade é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo de variabilidade presente em uma Característica. Pode assumir os seguintes valores: "ponto de variação", "variante" e "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 se configurar aquele 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.

No domínio exemplo de Engenharia de Requisitos, as características *Obtenção de Requisitos ou Especificação de Requisitos* são exemplos de características classificadas como ponto de variação. As características *Realizar Entrevista, Usar Cenários ou Documentar usando Casos de Uso* são exemplos de características classificadas como variantes. As características *Levantamento de Requisitos, Análise de Requisitos, Elaborar Roteiro de Homologação* ou *Homologar* são exemplos de características classificadas como invariantes.

c. CaracterísticaDisciplina

Descrição

Característica que representa uma categorização de trabalho baseado em uma similaridade de interesses e cooperação de esforços. 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

Superclasse: *Característica*

Atributos

- **propósito**: String [0..1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela característica.

Associações

Sem associações específicas.

Restrições

[1] Uma *CaracterísticaDisciplina* representa o agrupamento lógico de características da categoria *CaracterísticaTarefa* ou de características da categoria *CaracterísticaAtividade* (Figura 3).

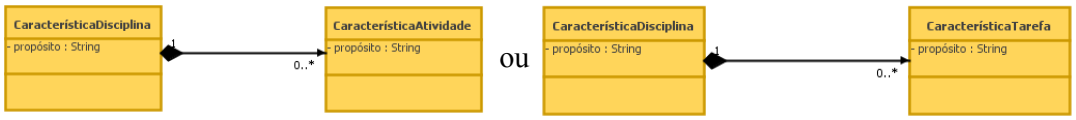


Figura 3 - Restrição entre as categorias de características Disciplina e Atividade

d. 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: *Característica*

Atributos

- **propósito**: String [0..1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela característica.

Associações

Sem associações específicas.

Restrições

[1] Uma *CaracterísticaAtividade* representa o agrupamento de trabalho expresso por outras características da categoria *CaracterísticaAtividade* ou por unidades elementares representadas por características da categoria *CaracterísticaTarefa*. Esses relacionamentos associados à restrição [1] da *Característica Disciplina* descrevem a possibilidade de construção de uma estrutura hierárquica, de um ou dois níveis, dos elementos de trabalho presentes em um domínio de processos de software modelado através de características e podem ser visualizados de forma resumida na Figura 4.

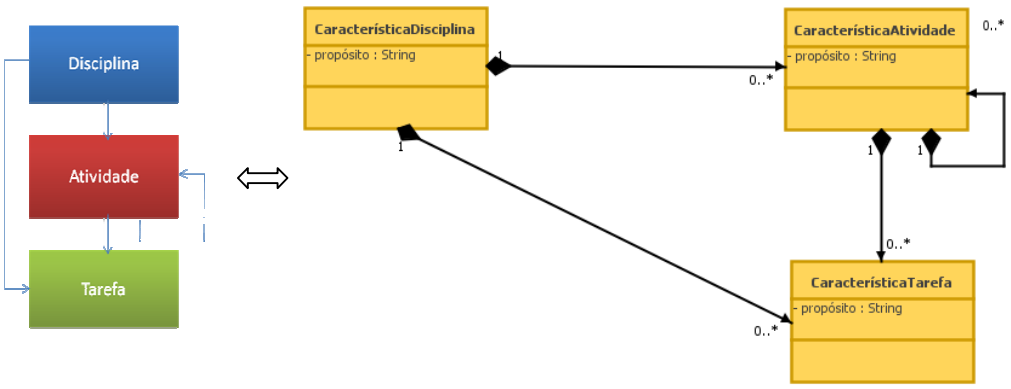


Figura 4 - Estrutura hierárquica das unidades de trabalho de um domínio de processos de software

e. CaracterísticaTarefa

Descrição

Característica 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, uma *CaracterísticaTarefa* pode ser detalhada através de instâncias da classe *Passo*, que representa a especificação das partes que compõem uma tarefa (Figura 5). Assim, uma tarefa representa uma unidade de trabalho elementar.

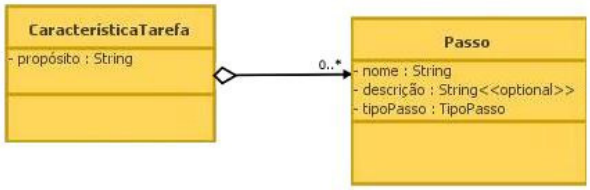


Figura 5 - Relação entre *CaracterísticaTarefa* e Passo

Hierarquia

Superclasse: *Característica*

Atributos

- **propósito**: String [0..1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela característica.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

f. Passo

Descrição

Corresponde a uma das partes que compõem o trabalho a ser executado durante a execução de uma tarefa. Corresponde a uma especificação detalhada de como atingir o propósito definido pela tarefa a qual pertence.

Hierarquia

Sem hierarquia específica.

Atributos

- **nome**: String [1] – atributo que define o nome do passo.

- **descrição**: String [0..1] – atributo que permite uma descrição textual sobre a execução do passo.
- **tipoPasso**: TipoPasso[1] – atributo que define o tipo do passo sendo descrito. Pode assumir os seguintes valores: *"Reflexão", "Execução"* e *"Revisão*". Default: "*Execução*".

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

g. TipoPasso

Descrição

TipoPasso é uma classe do tipo "*ennumeration*", que define literais que determinam o tipo de passo presente em uma Característica de Tarefa. Pode assumir os valores: "reflexão", "execução" e "revisão", onde:

1. *Reflexão*: passo em que o indivíduo que executa o papel compreende a natureza da tarefa, reúne e examina o(s) artefato(s) de entrada e formula a(s) saída(s);
2. *Execução*: passo em que o indivíduo que executa o papel cria ou atualiza algum(ns) artefato(s); e
3. *Revisão*: passo em que o indivíduo que executa o papel analisa o(s) resultado(s) em relação a alguns critérios.

Hierarquia

Sem hierarquia específica.

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

h. CaracterísticaPrática

Descrição

Característica que representa uma 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.

Hierarquia

Superclasse: *Característica*

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

[1] Para a característica Prática, o tipo de variabilidade só poderá ser classificado como Invariante, uma vez que essa característica não participa do relacionamento *Alternativo* (*Característica.tipoVariabilidade = invariante*).

i. CaracterísticaConceito

Descrição

Característica que representa um conceito dentro do domínio de processos de software. Descreve uma ideia-chave, aborda temas gerais ou princípios básicos que permeiam os diferentes elementos que constituem um processo.

Hierarquia

Superclasse: *Característica*

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

[1] Para a característica Conceito o tipo de variabilidade só poderá ser classificado como Invariante, uma vez que essa característica não participa do relacionamento *Alternativo* (*Característica.tipoVariabilidade = invariante*).

j. 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: *Característica*

Atributos

- **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.

k. CaracterísticaProdutoDeTrabalho

Descrição

Característica que representa um artefato consumido, modificado ou produzido por uma tarefa ou atividade.

Hierarquia

Superclasse: *Característica*

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Anexo II

ODYSSEYPROCESS-FEX: PACOTE RELACIONAMENTOS

O pacote Relacionamentos (*Relationships*) consiste na representação das relações disponibilizadas entre as características. Está estrutura na classe abstrata *Relacionamento* e suas especializações: *Alternativo, Herança, Associação, Agregação, Composição, LigaçãoPapelTarefa, LigaçãoProdutoDeTrabalhoTarefa e LigaçãoPapelProdutoDeTrabalho* (Figura 1).

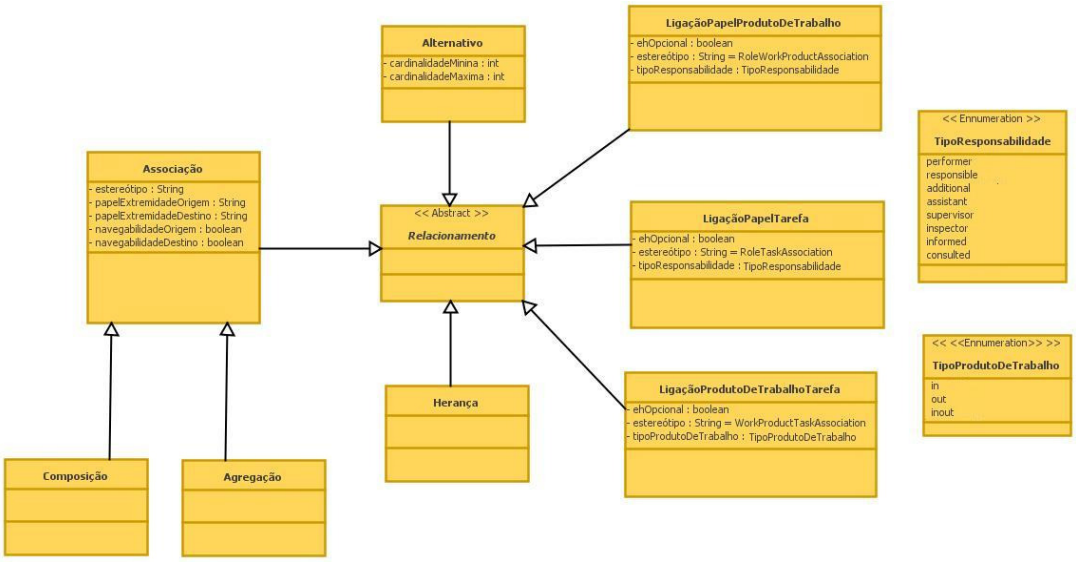


Figura 1 - Meta-modelo *OdysseyProcess-FEX*: Pacote Relacionamentos

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. Relacionamentos

Descrição

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

Hierarquia

SubClasses*: Alternativo, Herança, Associação, Agregação, Composição, LigaçãoPapelTarefa, LigaçãoProdutoDeTrabalhoTarefa, LigaçãoPapelProdutoDeTrabalho.*

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

b. 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 2 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.

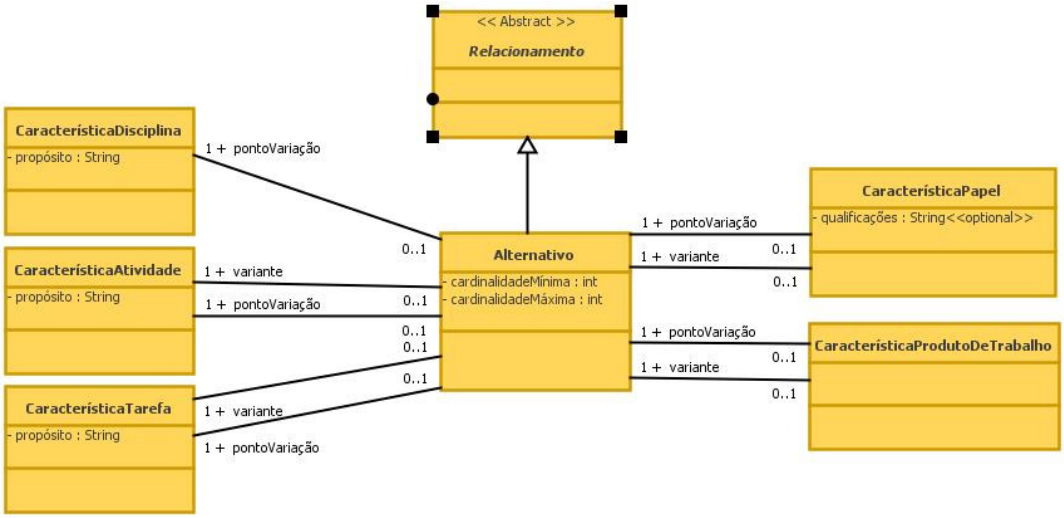


Figura 2 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento Alternativo

Hierarquia

Superclasse: **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:

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

Associações

- **variante**: Característica[1..*] – Referencia as características do tipo Variante que fazem parte do relacionamento Alternativo.
- **pontoVariação**: Característica[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 (*Característica.tipoVariabilidade = pontoVariação*) e do tipo Variante (*Característica.tipoVariabilidade=variante*).

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

[3] Características classificadas como Variantes (*Característica.tipoVariabilidade = variante*) são caracterizadas como destino do relacionamento do tipo *Alternativo*.

[4] Características classificadas como Variante (*Característica.tipoVariabilidade = variante*) e Mandatória (*Característica.ehMandatoria = true*) devem ter um relacionamento do tipo *Alternativo* com outra característica classificada como Ponto de Variação (*Característica.tipoVariabilidade = pontoVariação*) NECESSARIAMENTE Mandatória (*Característica.ehMandatoria = true*).

[5] Características classificadas como Variante (*Característica.tipoVariabilidade = variante*) e Opcionais (*Característica.ehMandatoria = false*) podem ter um relacionamento do tipo Alternativo com outra característica classificada como Ponto de Variação (*Característica.tipoVariabilidade = pontoVariação*) que seja Mandatória (*Característica.ehMandatoria = true*). 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 (*Característica.tipoVariabilidade = pontoVariação*) e Opcionais (*Característica.ehMandatoria = false*) devem ter um relacionamento do tipo *Alternativo* com outras características classificadas como Variantes (*Característica.tipoVariabilidade = variante*) NECESSARIAMENTE Opcionais (*Característica.ehMandatoria = false*).

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

[8] Relacionamentos Alternativos com cardinalidade mínima com valor igual a zero, 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, 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 1).

Tabela 1 - Relacionamento Alternativo: Restrições das combinações de categorias de Características

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

c. Herança

Descrição

Uma herança é um relacionamento entre uma característica mais geral e uma característica mais específica. Cada instância da característica específica é também um exemplo indireto da característica geral (OLIVEIRA *et al.,* 2005). Esse relacionamento permite uma organização hierárquica entre as características

envolvidas. A Figura 3 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 [3] do campo Restrições desse relacionamento.

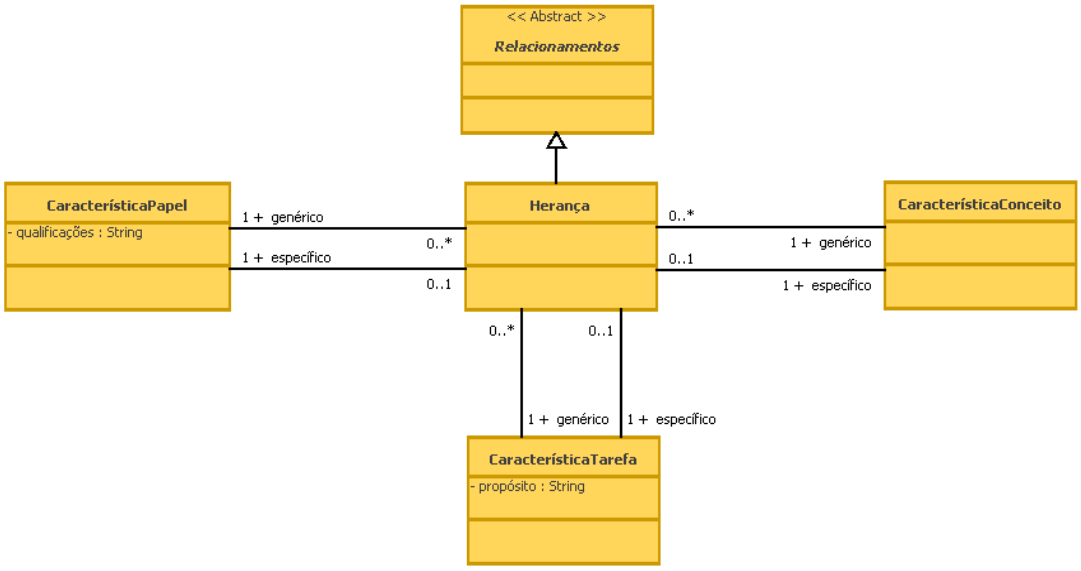


Figura 3 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento Herança

Hierarquia

Superclasse: *Relacionamentos*

Atributos

Sem atributos.

Associações

- **genérico**: Característica[1] – Referencia a característica mais geral no relacionamento de Herança.
- **específico**: Característica[1] – Referencia a característica mais específica no relacionamento de Herança.

Restrições

[1] O relacionamento de Herança definido não permite o conceito de herança múltipla. Uma característica que assume o papel específico no relacionamento só pode ter uma característica genérica como pai.

[2] Em um relacionamento do tipo Herança em que a característica que representa o elemento genérico é classificada como Opcional (*Característica.ehMandatoria=false*), as características que representam elementos

específicos devem ser NECESSARIAMENTE também classificadas como Opcionais (*Característica.ehMandatoria = false*).

[3] O relacionamento Herança só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 2):

Tabela 2 - Relacionamento Herança: Restrições das combinações de categorias de Características

Tipo de Relacionamento	**Categoria de Característica**	**Categoria de Característica**
	Origem: ***Específico***	*Destino:* ***Genérico***
Herança	CaracterísticaTarefa	CaracterísticaTarefa
	CaracterísticaConceito	CaracterísticaConceito
	CaracterísticaPapel	CaracterísticaPapel

d. 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: *Relacionamentos*

Atributos

- **estereótipo**: String[0..1] – atributo que permite uma descrição textual do tipo de ligação estabelecida pelo relacionamento.
- **papelExtremidadeOrigem:** String[0..1] – atributo que permite uma descrição textual do papel ocupado pela característica que representa a origem da ligação estabelecida pelo relacionamento.
- **papelExtremidadeDestino:** String[0..1] – atributo que permite uma descrição textual do papel ocupado pela característica que representa o destino da ligação estabelecida pelo relacionamento.
- **navegabilidadeOrigem:** Boolean – atributo que define se a ligação apresenta navegabilidade na direção da extremidade representada pela característica de origem da ligação.
- **navegabilidadeDestino:** Boolean – atributo que define se a ligação apresenta navegabilidade na direção da extremidade representada pela característica de destino da ligação.

Associações

Sem associações específicas.

Restrições

[1] A Tabela 3 apresenta o conjunto de combinações de categorias de características que podem ocorrer no relacionamento Associação. Vale ressaltar que a tabela não expressa o conceito de navegabilidade. Sendo assim, as combinações dois a dois apresentadas podem ser interpretadas como Origem – Destino, Destino – Origem ou relacionamentos bidirecionais.

Tabela 3 - Relacionamento Associação: Restrições das combinações de categorias de Características

Tipo de Relacionamento	Categoria de Característica	Categoria de Característica
Associação	CaracterísticaDisciplina	CaracterísticaDisciplina
	CaracterísticaDisciplina	CaracterísticaAtividade
	CaracterísticaDisciplina	CaracterísticaTarefa
	CaracterísticaDisciplina	CaracterísticaPrática
	CaracterísticaDisciplina	CaracterísticaConceito
	CaracterísticaAtividade	CaracterísticaAtividade
	CaracterísticaAtividade	CaracterísticaTarefa
	CaracterísticaAtividade	CaracterísticaPrática
	CaracterísticaAtividade	CaracterísticaConceito
	CaracterísticaTarefa	CaracterísticaTarefa
	CaracterísticaTarefa	CaracterísticaPrática
	CaracterísticaTarefa	CaracterísticaConceito
	CaracterísticaPrática	CaracterísticaConceito
	CaracterísticaConceito	CaracterísticaConceito
	CaracterísticaConceito	CaracterísticaPapel
	CaracterísticaConceito	CaracterísticaProdutoDeTrabalho
	CaracterísticaPapel	CaracterísticaPapel
	CaracterísticaProdutoDeTrabalho	CaracterísticaProdutoDeTrabalho

e. Agregação

Descrição

Uma associação pode representar uma agregação (isto é, um relacionamento de todo/parte) (OLIVEIRA *et al*., 2005). A Figura 4 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.

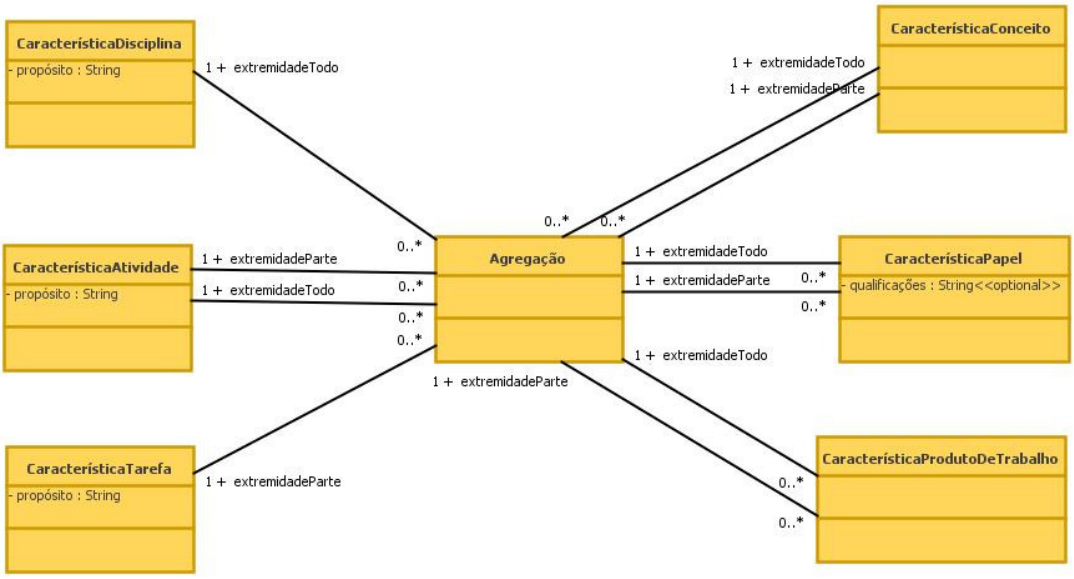


Figura 4 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento Agregação

Hierarquia

Superclasse: *Associação.*

Atributos

Sem atributos.

Associações

- **extremidadeTodo**: Referencia a característica que representa o todo no relacionamento de Agregação.
- **extremidadeParte**: Referencia a característica que representa parte no relacionamento de Agregação.

Restrições

[1] Somente associações binárias podem ser Agregação.

[2] O atributo que especifica a existência de navegabilidade na direção da característica que representa a origem da ligação e, consequentemente, o todo do relacionamento, deve possuir o valor negativo (*Associação. navegabilidadeOrigem = false*).

[3] O relacionamento Agregação só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 4):

Tabela 4 - Relacionamento Agregação: Restrições das combinações de categorias de Características

Tipo de Relacionamento	**Categoria de Característica**	**Categoria de Característica**
	Origem: ExtremidadeTodo	*Destino*: ExtremidadeParte
Agregação	Disciplina	Atividade
	Atividade	Atividade
	Atividade	Tarefa
	Conceito	Conceito
	Papel	Papel
	ProdutoDeTrabalho	ProdutoDeTrabalho

f. 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 Figura5 apresenta o relacionamento e as categorias de características envolvidas.

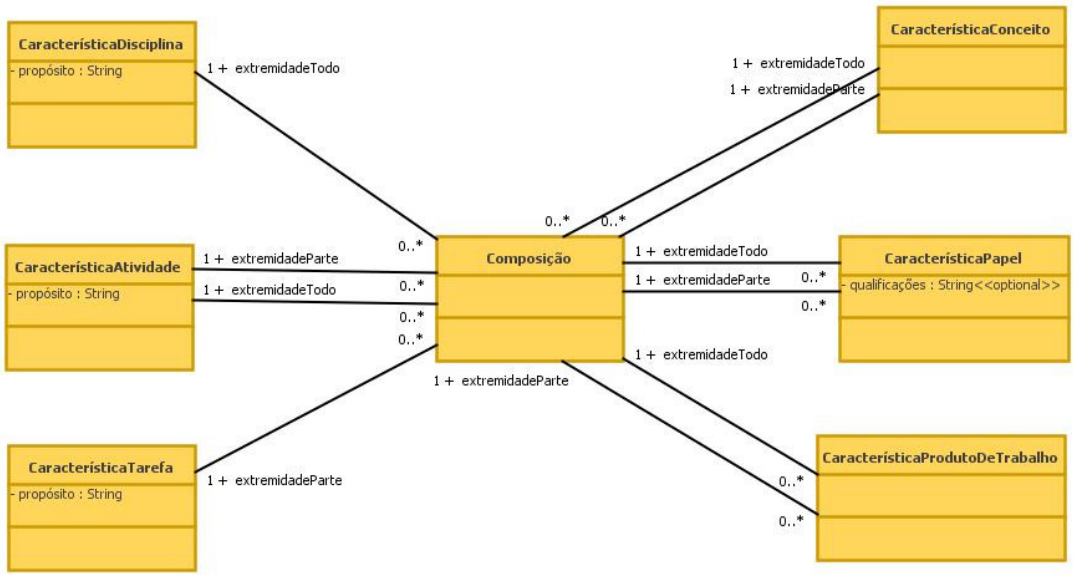


Figura 5 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento Composição

Hierarquia

Superclasse: *Relacionamentos*

Atributos

Sem atributos.

Associações

- **extremidadeTodo**: Referencia a característica que representa o todo no relacionamento de Composição.
- **extremidadeParte**: Referencia a característica que representa parte no relacionamento de Composição.

Restrições

[1] Somente associações binárias podem ser Composição.

[2] Não deve haver relacionamento de composição entre características quando a característica que representa o "todo" é opcional e a característica que representa a "parte" é mandatória.

[3] O atributo que especifica a existência de navegabilidade na direção da característica que representa a origem da ligação, e consequentemente, o todo do relacionamento, deve possuir o valor negativo (*Associação. navegabilidadeOrigem = false*).

[4] O relacionamento Composição só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 5):

Tabela 5 - Relacionamento Composição: Restrições das combinações de categorias de Características

Tipo de Relacionamento	**Categoria de Característica**	**Categoria de Característica**
	Origem: ExtremidadeTodo	*Destino*: ExtremidadeParte
Composição	Disciplina	Atividade
	Atividade	Atividade
	Atividade	Tarefa
	Conceito	Conceito
	Papel	Papel
	ProdutoDeTrabalho	ProdutoDeTrabalho

g. LigaçãoPapelTarefa

Descrição

Estabelece um relacionamento de associação entre uma característica Tarefa e uma ou várias características Papel participantes na sua execução. A Figura6 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.

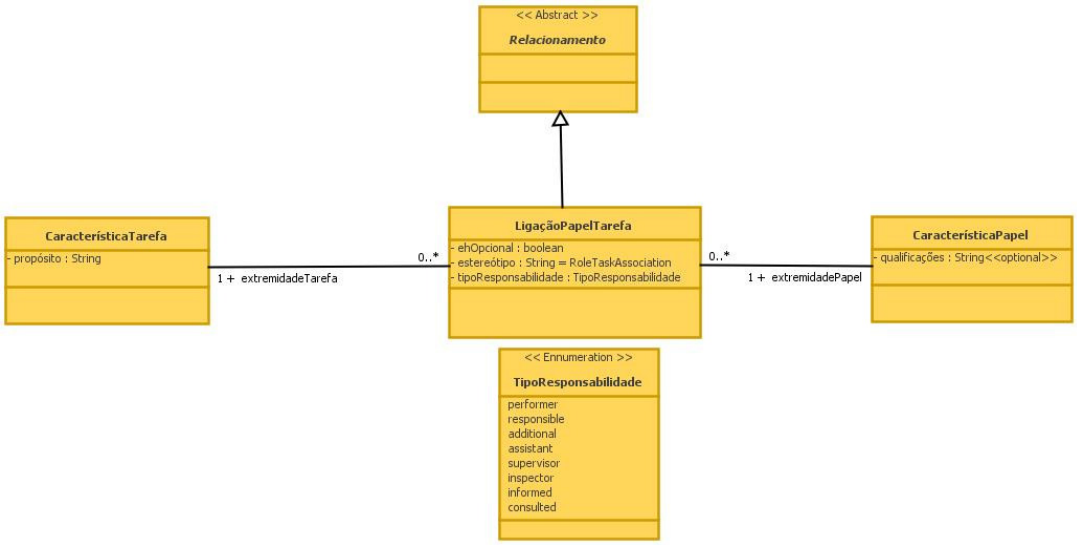


Figura 6 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento LigaçãoPapelTarefa

Hierarquia

Superclasse: *Relacionamentos*

Atributos

- **ehOpcional**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação da característica Papel na execução da unidade de trabalho representada pela característica tarefa associada. Caso *true*, o relacionamento é Opcional. Default: *true*.

Associações

- **extremidadeTarefa**: Referencia a característica que representa a tarefa envolvida no relacionamento de LigaçãoPapelTarefa.
- **extremidadePapel**: Referencia a característica que representa o papel envolvido no relacionamento de LigaçãoPapelTarefa.

Restrições

[1] O relacionamento criado pode ser classificado quanto a diferentes tipos de responsabilidades definidos através dos seguintes estereótipos: *<<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>* (Tabela 6).

[2] O relacionamento LigaçãoPapelTarefa só poderá ocorrer entre a característica Tarefa e características Papel que fazem parte da execução da Tarefa.

Tabela 6 - Tipos de Responsabilidades dos Papéis envolvidos na execução de Tarefas

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)

h. LigaçãoProdutoDeTrabalhoTarefa

Descrição

Estabelece um relacionamento de associação entre uma característica Tarefa e uma ou várias características ProdutoDeTrabalho que representam produtos de trabalho a serem consumidos, produzidos ou modificados pela execução da unidade de trabalho. A Figura 7 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.

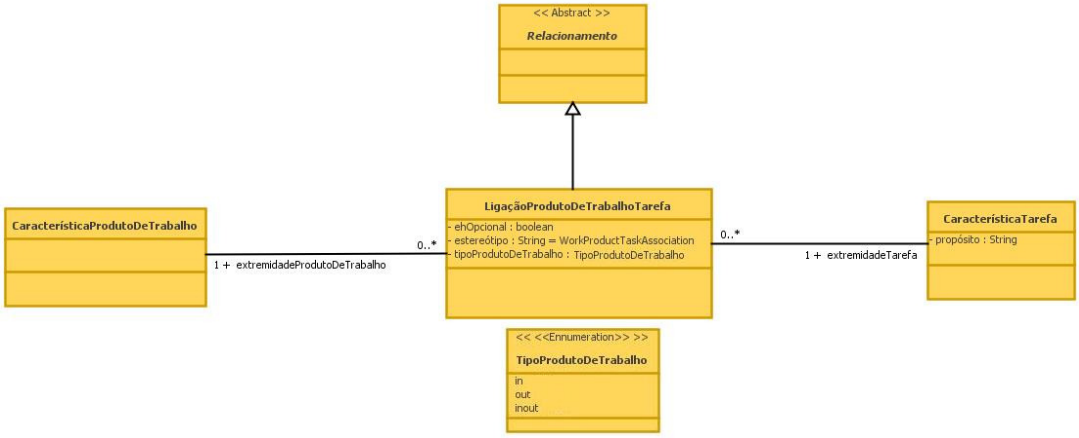


Figura 7 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento LigaçãoProdutoDeTrabalhoTarefa

Hierarquia

Superclasse: *Relacionamentos*

Atributos

- **ehOpcional**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de

participação da característica ProdutoDeTrabalho na execução da unidade de trabalho representada pela característica Tarefa associada. Caso *true*, o relacionamento é Opcional. Default: *true*.

Associações

- **extremidadeTarefa**: Referencia a característica que representa a tarefa envolvida no relacionamento de LigaçãoProdutoDeTrabalhoTarefa.
- **extremidadeProdutoDeTrabalho**: Referencia a característica que representa o Produto de Trabalho envolvido no relacionamento de LigaçãoProdutoDeTrabalhoTarefa.

Restrições

[1] O relacionamento criado pode ser classificado quanto a diferentes tipos de produtos de trabalho definidos através dos seguintes estereótipos: *<<in>>, <<out>>, <<inout>>* (Tabela 7).

Tabela 7 - Tipos Produtos de Trabalho

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

[2] O relacionamento LigaçãoProdutoDeTrabalhoTarefa só poderá ocorrer entre a características Tarefa e características ProdutoDeTrabalho que fazem parte na execução da Tarefa.

i. LigaçãoPapelProdutoDeTrabalho

Descrição

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. A Figura 8 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.

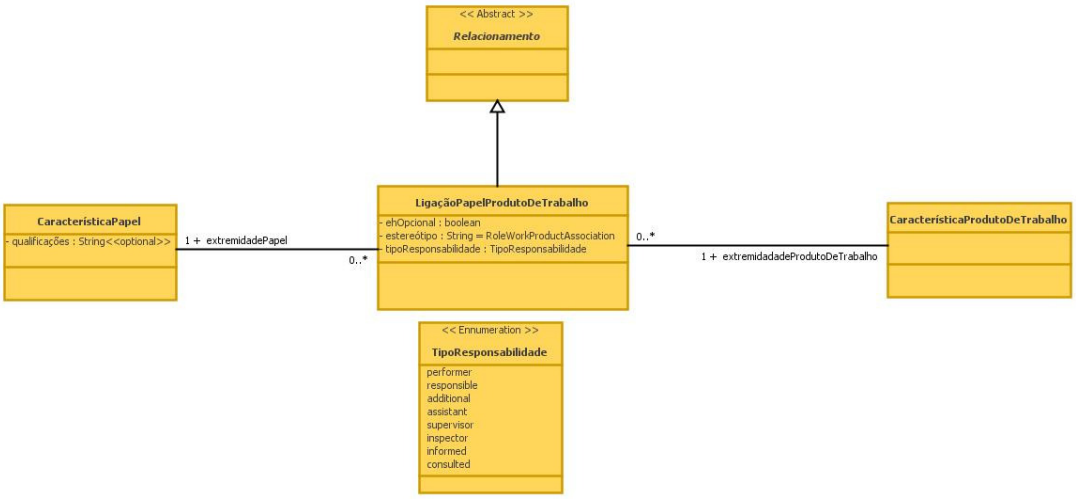


Figura 8 - Meta-modelo *OdysseyProcess-FEX*: Relacionamento LigaçãoPapelProdutoDeTrabalho

Hierarquia

Superclasse: *Relacionamentos*

Atributos

- **ehOpcional**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação de responsabilidade da característica Papel no Produto de Trabalho associado. Caso *true*, o relacionamento é Opcional. Default: *true*.

Associações

- **extremidadeProdutoDeTrabalho**: Referencia a característica que representa o produto de trabalho envolvido no relacionamento de LigaçãoPapelProdutoDeTrabalho.
- **extremidadePapel**: Referencia a característica que representa o papel envolvido no relacionamento de LigaçãoPapelProdutoDeTrabalho.

Restrições

[1] A associação criada pode ser classificada quanto a diferentes tipos de responsabilidades definidos através dos seguintes estereótipos: *<<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>* (Tabela 6).

[2] O relacionamento *LigaçãoPapelProdutoDeTrabalho* só poderá ocorrer entre a características ProdutoDeTrabalho e características Papel com alguma responsabilidade sobre o Produto de Trabalho.

Anexo III

ODYSSEYPROCESS-FEX: PACOTE REGRAS DE COMPOSIÇÃO

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 1.

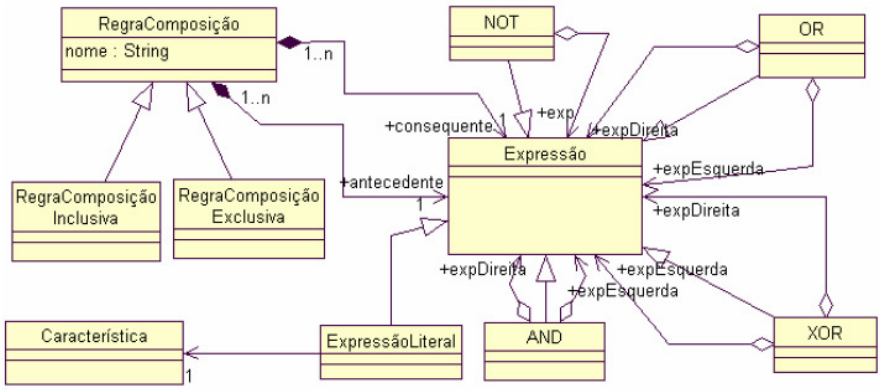


Figura 1 - Meta-modelo *OdysseyProcess-FEX*: Pacote Regras de Composição (OLIVEIRA, 2006)

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 IV

FORMULÁRIO DE CONSENTIMENTO

OBJETIVO DO ESTUDO

Este estudo visa compreender a viabilidade de aplicação do meta-modelo e notação *OdysseyProcess-FEX*, cujo propósito é modelar características do domínio de acordo com a abordagem de Linhas de Processos de Software.

IDADE

Eu declaro ter mais de 18 anos de idade e concordar em participar de um estudo conduzido por Eldanae Nogueira Teixeira na Universidade Federal do Rio de Janeiro.

PROCEDIMENTO

Este estudo acontecerá em duas etapas. Na primeira etapa, é apresentado todo o conjunto de documentos que serão utilizados neste estudo, parte de um modelo de características de um domínio de processos de software e a especificação do domínio que deverá ser modelado. Na segunda etapa, os participantes deverão realizar a modelagem de variabilidades especificada, de forma manual, com o auxílio do preenchimento de tabelas e criação de um esboço do modelo final. Por último, um formulário de avaliação sobre o estudo realizado deve ser preenchido pelo participante.

CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

BENEFÍCIOS E LIBERDADE DE DESISTÊNCIA

Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi, serão estudados visando entender a eficiência dos procedimentos e as técnicas que me foram ensinadas.

Os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado. Também entendo que sou livre para realizar perguntas a qualquer momento, solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo ou comunicar minha desistência de participação. Por fim, declaro que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

PESQUISADORES RESPONSÁVEIS

Eldanae Nogueira Teixeira

Andréa Magalhães Magdaleno

Cláudia Maria Lima Werner

Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ

Nome (em letra de forma): _____

Assinatura: _____

Data:_____

Anexo V

FORMULÁRIO DE CARACTERIZAÇÃO DO PARTICIPANTE

Este formulário contém algumas perguntas sobre sua experiência acadêmica e profissional.

1. Formação acadêmica

( ) Pós-Doutorado
( ) Doutorado
( ) Doutorando
( ) Mestrado
( ) Mestrando
( ) Graduação

Ano de ingresso: _____
Ano de conclusão (ou previsão de conclusão): _____

2. Formação geral

Por favor, indique o grau de sua experiência nesta seção, seguindo a escala de 5 pontos abaixo:

0 = nenhum
1 = estudei em aula ou em livro
2 = pratiquei em projetos em sala de aula
3 = usei em projetos pessoais
4 = usei em projetos na indústria

2.1. Definição de Processos de Software	0	1	2	3	4
2.2. Linha de Produtos de Software	0	1	2	3	4
2.3. Modelagem de Características	0	1	2	3	4
2.4. Linha de Processos de Software	0	1	2	3	4

3. Experiência no domínio exemplo

Esta seção será utilizada para compreender quão familiar você é com o domínio que será utilizado para as atividades durante o experimento. Por favor, indique o grau de experiência nesta seção, seguindo a escala de 3 pontos abaixo:

0 = Eu não tenho familiaridade com este domínio. Eu nunca trabalhei com este assunto.
1 = Eu utilizo isto algumas vezes, mas não sou um especialista.
2 = Eu sou muito familiar com este domínio. Eu me sentiria confortável fazendo isto.

3.1. Engenharia de Requisitos de Software	0	1	2

Obrigada por sua colaboração!

Anexo VI
LEITURA INTRODUTÓRIA

Conceitos Gerais – Linha de Processos de Software

A Linha de Processos de Software (LPS) é uma abordagem cujo principal objetivo é desenvolver artefatos do domínio de processos de software, com o propósito de serem reutilizados em novos processos específicos de projeto. A Engenharia de Linha de Processos de Software (ELPS) compreende toda a estrutura necessária para construir, utilizar e gerir uma LPS. Ela é constituída de duas grandes fases: a Engenharia de Domínio de Processos de Software (EDPS), que define uma sistemática para o desenvolvimento de artefatos de processo reutilizáveis e a Engenharia de Aplicação de Processos de Software (EAPS), que define um processo para a instanciação de processos específicos de projetos baseados nos artefatos gerados na EDPS.

A EDPS possui a etapa de Análise do Domínio de Processos de Software (ADPS), que se preocupa com a identificação e representação do conhecimento do domínio de processos de software, através de uma análise de suas similaridades e diferenças. Neste sentido, é importante que na ADPS se tenha uma forma sistemática de representar os artefatos de processos que constituem a base dos processos a serem instanciados, bem como suas opcionalidades e variabilidades.

OdysseyProcess-FEX

No estudo de observação que irá participar é utilizada a notação *OdysseyProcess-FEX*, que representa simbolicamente o meta-modelo de mesmo nome. O meta-modelo combina a representação dos conceitos de características, variabilidade e opcionalidade aos elementos que constituem a definição de processos de software, como unidades de trabalho, papéis e produtos de trabalho.

A notação *OdysseyProcess-FEX* permite a classificação de características de processos de software quanto a sua categoria, variabilidade e opcionalidade. Podemos identificar um conjunto de categorias de características especificadas na Tabela 1.

Tabela 1 - Categorias de Características na notação *OdysseyProcess-FEX*

Categoria		Ícone	Estereótipo
Disciplina	Característica que representa uma categorização de trabalho baseado em uma similaridade de interesses e cooperação de esforços, relacionado com uma grande área de interesse no âmbito do domínio como um todo.		***<<discipline>>***
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. Pode prover 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.		***<<task >>***
Prática	Característica que representa uma 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.		***<<practice>>***
Conceito	Característica que descreve ideias-chave, aborda temas gerais ou princípios básicos que permeiam os diferentes elementos que constituem um processo.		***<<concept>>***
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 tarefa.		***<<work product>>***

Os diferentes tipos de categoria são mutuamente exclusivos entre si, ou seja, uma característica não pode pertencer simultaneamente a mais de uma categoria. O mesmo é válido para classificações de variabilidades e opcionalidades.

O conceito de variabilidade será trabalhado através da classificação de uma característica como:

> *Ponto de variação*: característica que reflete a parametrização no domínio, ou seja, representa um ponto do domínio que deve ser configurado segundo a escolha de alternativas dentro de um conjunto de características associadas, denominadas variantes.
>
> *Variante*: elemento necessariamente ligado a um ponto de variação, que atua como alternativa para se configurar aquele ponto de variação.
>
> *Invariante*: elemento fixo, que não é configurável no domínio.

A opcionalidade determina a obrigatoriedade ou não da presença de determinado elemento nos múltiplos processos a serem instanciados a partir dos artefatos da linha. Vale ressaltar que a opcionalidade é referente ao domínio como um todo. A

classificação quanto à opcionalidade é representada graficamente através da diferenciação do contorno do retângulo da característica. Características mandatórias são representadas por um retângulo formado por uma linha contínua. Características opcionais são representadas por um retângulo formado por uma linha tracejada. Exemplos de características opcionais podem ser identificados na Figura 1, como as características *Controle de Concorrência* e suas variantes.

O meta-modelo *OdysseyProcess-FEX* proposto valoriza a semântica dos relacionamentos em um modelo de características. No total, oito relacionamentos foram especificados no meta-modelo (Tabela 2): *Alternativo, Herança, Associação, Agregação, Composição, LigaçãoPapelTarefa, LigaçãoProdutoDeTrabalhoTarefa e LigaçãoPapelProdutoDeTrabalho*.

Tabela 2 - Relacionamentos notação *OdysseyProcess-FEX*

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.	
Herança	Uma herança é um relacionamento entre uma característica mais geral e uma mais específica. É representado como uma linha com triângulo não-preenchido entre as características envolvidas, que aponta para a característica geral.	
Associação	Uma associação representa a relação, uni ou bidirecional, entre duas características. O relacionamento pode possuir um estereótipo que especifica o tipo da ligação. É, normalmente, representado com uma linha contínua que conecta duas características, ou uma linha contínua que conecta uma única característica (com duas extremidades distintas). Uma seta na extremidade de uma associação indica navegabilidade.	
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 PapelTarefa	Estabelece um relacionamento de associação entre uma característica Tarefa e uma ou várias características Papel participantes na sua execução.	
Ligação ProdutoDeTrabalho Tarefa	Estabelece um relacionamento de associação entre uma característica Tarefa e uma ou várias características ProdutoDeTrabalho que representam produtos de trabalho a serem consumidos, produzidos ou modificados pela execução da unidade de trabalho.	
Ligação Papel ProdutoDeTrabalho	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.	

Restrições de dependência ou mútua exclusividade, existentes entre características, são expressas através de Regras de Composição. Uma Regra de Composição tem a seguinte estrutura: ***Antecedente + palavra-chave + Consequente***. O antecedente e o consequente são expressões, que podem ser literais ou booleanas, e denotam uma característica do domínio ou uma combinação entre estas. A palavra-chave apresentada na estrutura define o tipo de Regras de Composição: Regras Inclusivas são denotadas pela palavra-chave "*requer*", enquanto que Regras Exclusivas são denotadas pela palavra-chave "*exclui*". São graficamente representadas por uma marcação, no canto inferior do retângulo que representa a característica envolvida na regra estabelecida. Vale destacar que a marcação é apresentada em todas as características pertencentes a uma Regra de Composição estabelecida. Para Regras de Composição Inclusivas, a marcação usada com a numeração corresponde a "R_n" e é apresentada no canto inferior direito. Já para Regras de Composição Exclusivas, a marcação usada com a numeração corresponde a "X_n" e é apresentada no canto esquerdo. Um exemplo de Regra de Composição Inclusiva pode ser identificada na Figura 1, com as características *Armazenamento via Repositório Central e Controle de Concorrência*.

Exemplo de Uso da Notação *OdysseyProcess-FEX* - Domínio Disciplina de Gerência de Configuração de Software

Para ilustrar os conceitos acima apresentados será utilizado um modelo de características parcial da disciplina de Gerência de Configuração de Software. Uma de suas atividades obrigatórias consiste no *Controle de Versões*. Para essa atividade, podemos destacar como tarefas mandatórias: (1) *Identificação de Itens de Configuração* (IC's) e (2) *Armazenamento de Versões de IC's*. A tarefa de Armazenamento de Versões de IC's é representada como um ponto de variação associado, através do relacionamento alternativo, com suas duas variantes: *Armazenamento via Repositório Central* ou *Armazenamento via Repositório Distribuído.*

A tarefa de *Controle de Concorrência* é opcional e deve ser aplicada sempre que houver o uso de um repositório centralizado. Esta tarefa pode ser realizada através de uma de duas tarefas alternativas: *Controle via Política Otimista* e *Controle via Política Pessimista*. Uma Regra de Composição Inclusiva foi criada para especificar a relação de dependência: "Armazenamento via Repositório Central *requer* Controle de

Concorrência", e pode ser visualizada através da marcação R presente no canto inferior direito das duas características correspondentes.

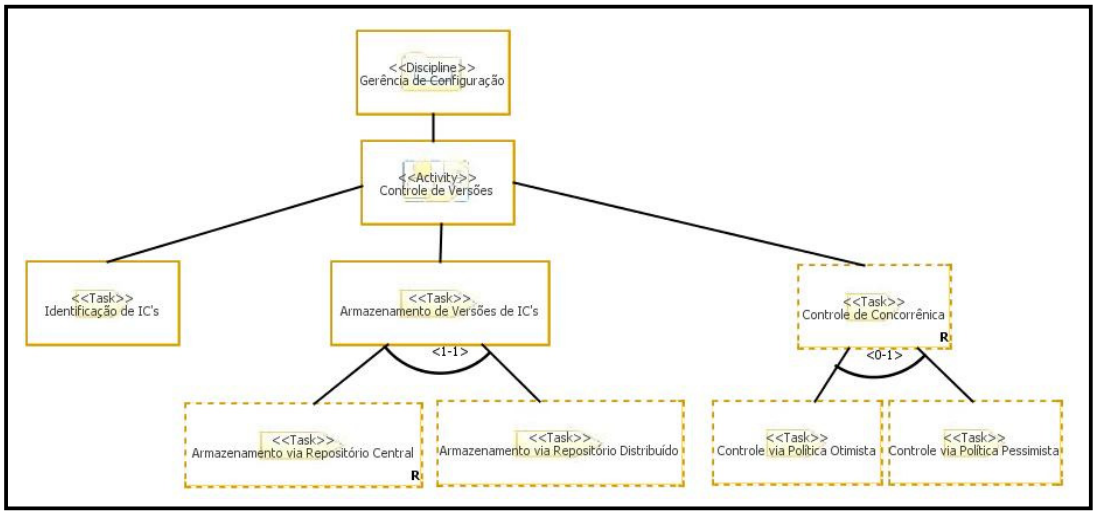


Figura 1 - Modelo de Características de parte do domínio de Gerência de Configuração usando a notação *OdysseyProcess-FEX*

Anexo VII

ODYSSEYPROCESS-FEX: REGRAS DE BOA-FORMAÇÃO

O meta-modelo *OdysseyProcess-FEX* possui um conjunto de restrições e propriedades, que juntas constituem as regras de boa formação do modelo. Essas regras direcionam a construção e a verificação de consistência de um modelo de características do domínio de processos de software. Um conjunto limitado dessas regras foi selecionado para possível consulta durante a realização deste estudo.

Restrições Taxonomia

R1: Uma *CaracterísticaDisciplina* representa o agrupamento lógico de características da categoria *CaracterísticaTarefa* ou de características da categoria *CaracterísticaAtividade.*

R2: Uma *CaracterísticaAtividade* representa o agrupamento de trabalho expresso por outras características da categoria *CaracterísticaAtividade* ou por unidades elementares representadas por características da categoria *CaracterísticaTarefa.*

Restrições Relacionamentos

Relacionamento Alternativo

R3: O relacionamento Alternativo só poderá ocorrer entre Características do tipo Ponto de Variação (*Característica.tipoVariabilidade = pontoVariação*), origem do relacionamento, e do tipo Variante (*Característica.tipoVariabilidade=variante*), destino do relacionamento.

R4: Características classificadas como Variante (*Característica.tipoVariabilidade = variante*) e Mandatória (*Característica.ehMandatoria = true*) devem ter um relacionamento do tipo *Alternativo* com outra característica classificada como Ponto de Variação (*Característica.tipoVariabilidade = pontoVariação*) NECESSARIAMENTE Mandatória (*Característica.ehMandatoria = true*).

R5: Características classificadas como Ponto de Variação (*Característica.tipoVariabilidade = pontoVariação*) e Opcionais (*Característica.ehMandatoria = false*) devem ter um relacionamento do tipo *Alternativo* com outras características classificadas como Variantes (*Característica.tipoVariabilidade = variante*) NECESSARIAMENTE Opcionais (*Característica.ehMandatoria = false*).

R6: O relacionamento Alternativo só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 1):

Tabela 1 - Relacionamento Alternativo: Restrições das combinações de categorias de Características

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

Relacionamento Herança

R7: O relacionamento de Herança definido não permite o conceito de herança múltipla. Uma característica que assume o papel específico no relacionamento só pode ter uma característica genérica como pai.

R8: Em um relacionamento do tipo Herança em que a característica que representa o elemento genérico é classificada como Opcional (*Característica.ehMandatoria=false*), as características que representam elementos específicos devem ser NECESSARIAMENTE também classificadas como Opcionais (*Característica.ehMandatoria = false*)

R9: O relacionamento Herança só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 2):

Tabela 2 - Relacionamento Herança: Restrições das combinações de categorias de Características

Tipo de Relacionamento	Categoria de Característica	Categoria de Característica
	Origem: ***Específico***	*Destino:* ***Genérico***
Herança	CaracterísticaTarefa	CaracterísticaTarefa
	CaracterísticaConceito	CaracterísticaConceito
	CaracterísticaPapel	CaracterísticaPapel

Relacionamento Associação

R10: A Tabela 3 apresenta o conjunto de combinações de categorias de características que podem ocorrer no relacionamento Associação. Vale ressaltar que a tabela não expressa o conceito de navegabilidade. Sendo assim, as combinações dois a dois

apresentadas podem ser interpretadas como Origem – Destino, Destino – Origem ou relacionamentos bidirecionais.

Tabela 3 - Relacionamento Associação: Restrições das combinações de categorias de Características

Tipo de Relacionamento	Categoria de Característica	Categoria de Característica
Associação	CaracterísticaDisciplina	CaracterísticaDisciplina
	CaracterísticaDisciplina	CaracterísticaAtividade
	CaracterísticaDisciplina	CaracterísticaTarefa
	CaracterísticaDisciplina	CaracterísticaPrática
	CaracterísticaDisciplina	CaracterísticaConceito
	CaracterísticaAtividade	CaracterísticaAtividade
	CaracterísticaAtividade	CaracterísticaTarefa
	CaracterísticaAtividade	CaracterísticaPrática
	CaracterísticaAtividade	CaracterísticaConceito
	CaracterísticaTarefa	CaracterísticaTarefa
	CaracterísticaTarefa	CaracterísticaPrática
	CaracterísticaTarefa	CaracterísticaConceito
	CaracterísticaPrática	CaracterísticaConceito
	CaracterísticaConceito	CaracterísticaConceito
	CaracterísticaConceito	CaracterísticaPapel
	CaracterísticaConceito	CaracterísticaProdutoDeTrabalho
	CaracterísticaPapel	CaracterísticaPapel
	CaracterísticaProdutoDeTrabalho	CaracterísticaProdutoDeTrabalho

Relacionamento Agregação

R11: O relacionamento Agregação só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 4):

Tabela 4 - Relacionamento Agregação: Restrições das combinações de categorias de Características

Tipo de Relacionamento	Categoria de Característica	Categoria de Característica
	Origem: ExtremidadeTodo	*Destino*: ExtremidadeParte
Agregação	Disciplina	Atividade
	Atividade	Atividade
	Atividade	Tarefa
	Conceito	Conceito
	Papel	Papel
	ProdutoDeTrabalho	ProdutoDeTrabalho

Relacionamento Composição

R12: Não deve haver relacionamento de composição entre características quando a característica que representa o "todo" é opcional e a característica que representa a "parte" é mandatória.

R13: O relacionamento Composição só poderá ocorrer entre as seguintes combinações de categorias de Características (Tabela 5).

Tabela 5 - Relacionamento Composição: Restrições das combinações de categorias de Características

Tipo de Relacionamento	**Categoria de Característica**	**Categoria de Característica**
	Origem: ExtremidadeTodo	*Destino*: ExtremidadeParte
Composição	Disciplina	Atividade
	Atividade	Atividade
	Atividade	Tarefa
	Conceito	Conceito
	Papel	Papel
	ProdutoDeTrabalho	ProdutoDeTrabalho

Relacionamentos LigaçãoPapelTarefa/ LigaçãoProdutoDeTrabalhoTarefa/ LigaçãoPapelTarefa

R14: O relacionamento *LigaçãoPapelTarefa* só poderá ocorrer entre a característica Tarefa e as características Papel que fazem parte na execução da Tarefa.

R15: O relacionamento *LigaçãoProdutoDeTrabalhoTarefa* só poderá ocorrer entre a característica Tarefa e características ProdutoDeTrabalho que fazem parte na execução da Tarefa.

R16: O relacionamento *LigaçãoPapelProdutoDeTrabalho* só poderá ocorrer entre a características ProdutoDeTrabalho e características Papel com alguma responsabilidade sobre o Produto de Trabalho.

R17: Os relacionamentos *LigaçãoPapelTarefa* e *LigaçãoPapelProdutoDeTrabalho* podem ser classificados quanto a diferentes tipos de responsabilidades especificadas pela Tabela 6.

Tabela 6 - Tipos de Responsabilidades dos Papéis envolvidos na execução de Tarefas

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)

R18*:* O relacionamento *LigaçãoProdutoDeTrabalhoTarefa* criado pode ser classificado quanto a diferentes tipos de produtos de trabalho definidos através dos seguintes estereótipos: *<<in>>, <<out>>, <<inout>>* (Tabela 7).

Tabela 7 - Tipos Produtos de Trabalho

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

Restrições Regras de Composição

R19: Uma Regra de Composição não pode ser contraditória, i.e., não pode ter antecedente e consequente iguais.

R20: Uma Regra de Composição não pode ter antecedente definido e consequente nulo ou vice versa.

R21: Características dependentes entre si não podem ser mutuamente exclusivas, e vice-versa.

R22: Em uma Regra de Composição Inclusiva, o consequente só poderá ser opcional se o antecedente for opcional.

R23: 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.

R24: Uma Regra de Composição Exclusiva não deve envolver características mandatórias, somente características opcionais.

Anexo VIII
DESCRIÇÃO DA TAREFA

Instruções Gerais

Este é um estudo de observação, por isso, sempre que possível, verbalize seus pensamentos, para que o experimentador possa melhor avaliar os resultados obtidos. Pergunte e comente tudo que achar necessário.

Contextualização

A Engenharia de Requisitos de Software consiste em uma das disciplinas que constituem o processo de desenvolvimento de software e pode ser entendida como um processo pelo qual requisitos de um produto de software são definidos. Processos de Engenharia de Requisitos podem variar muito em função de diferentes características de organizações e projetos. Existem diversos fatores que contribuem para a variabilidade nesses processos, dentre eles a cultura organizacional, domínios de aplicação e objetivos dos projetos.

Ainda que diferentes projetos possuam processos com características específicas, é possível estabelecer um conjunto de atividades e tarefas básicas que deve ser considerado na definição de um processo de Engenharia de Requisitos. A descrição a seguir representa parte do domínio de processos da disciplina de Engenharia de Requisitos, aqui modelada utilizando um modelo de características com a notação *OdysseyProcess-FEX.* A descrição gráfica do modelo exemplo encontra-se no documento em Anexo IX (Descrição do Modelo Exemplo). Vale ressaltar que a especificação do domínio foi limitada como forma de simplificar o entendimento para a realização do estudo.

Dentre as atividades principais podemos citar: *Levantamento de Requisitos; Análise de Requisitos; Especificação de Requisitos* e *Validação de Requisitos*.

- A atividade de ***Levantamento de Requisitos*** corresponde à descoberta de requisitos através da execução das tarefas: (1) *Definir Visão*, que permite a identificação dos diferentes envolvidos e de uma definição inicial do problema a ser resolvido, e (2) *Elicitação de Requisitos*, que corresponde à tarefa de coletar informações de diferentes fontes para especificar as necessidades a serem atendidas. A tarefa de *Elicitação de Requisitos* apresenta-se como um ponto de variação dentro do domínio, podendo ser realizada através de diferentes variantes: *Realizar Entrevistas, Aplicar Questionários, Usar Cenários, Construir Protótipos e Realizar*

Brainstorming. Essas alternativas podem ser executadas em conjunto, de forma complementar. A tarefa *Definir Glossário*, ainda envolvida nesta atividade, é opcional e tem como função principal definir termos envolvidos no projeto a ser desenvolvido para construir um entendimento comum entre os envolvidos. Os produtos de trabalho *Documento Visão* e *Documento de Requisitos* são artefatos resultantes das duas primeiras tarefas. A tarefa *Definir Glossário* possui como produto de trabalho um documento chamado *Glossário*, que é considerado opcional no domínio como um todo.

- A atividade de ***Análise de Requisitos*** corresponde a um trabalho de negociação e priorização dos requisitos a serem desenvolvidos. Possui como tarefas mandatórias: (1) *Priorizar Requisitos* e (2) *Negociar Requisitos*. A tarefa de *Classificar e Organizar Requisitos* depende de características de projeto e de sistemas, sendo portanto opcional.

- A atividade de ***Especificação de Requisitos*** corresponde à documentação dos requisitos levantados em diferentes graus de formalismo, representando um ponto de múltiplas alternativas, desde descrições básicas das funcionalidades a serem disponibilizadas pelo sistema a ser desenvolvido, como no caso de metodologias ágeis, que usam *estórias de usuários* como forma de representação de requisitos; até especificações mais formais que usam *documentos em linguagens naturais* associados a documentos que usam *modelos como casos de usos*.

- A atividade de ***Verificação & Validação de Requisitos*** corresponde a tarefas de verificação e validação dos requisitos com relação a aspectos de consistência, resolução de conflitos, conformidade e atendimentos de interesses. É composta da atividade de *Revisão de Requisitos*, mandatória para realizar verificações nos requisitos levantados, e das tarefas *Elaborar Roteiro de Homologação* e *Homologar,* que constituem a parte de validação. A tarefa *Elaborar Roteiro de Homologação* possui como produto de trabalho o documento *Roteiro de Homologação.*

- Uma das práticas identificadas dentro desse domínio corresponde à utilização de casos de uso para dirigir todo o trabalho de desenvolvimento de software. Esta foi representada através da característica ***Desenvolvimento Dirigido por Caso de Uso***.

A adoção desta prática requer a execução da tarefa *Documentar usando Casos de Uso*.

- O principal **papel** presente na disciplina de Engenharia de Requisitos consiste no papel de *Analista de Requisitos,* que é o executante de várias das tarefas descritas. O papel *Stakeholder* é considerado como adicional ao papel do analista e é consultado para a execução de tarefas envolvidas nas atividades de Levantamento, Análise e Validação de Requisitos. A participação do *Desenvolvedor* é opcional e pode ser adicionada durante atividades de Especificação de Requisitos. A participação de cada um dos papéis pode ser visualizada através da perspectiva de papéis representada no modelo anexo no Documento de Descrição do Modelo Exemplo.

Especificação do Refinamento do Domínio

Demandas crescentes impostas pela constante competitividade de mercado têm levado a indústria de software a adaptar sua forma tradicional de desenvolvimento de sistemas. Abordagens como a reutilização de software têm sido aplicadas como forma de construir sistemas a partir de software existentes, reduzindo o esforço e o custo de produção envolvidos. A abordagem de Linha de Produtos de Software (LPS) surge como uma técnica sistemática de aplicar a reutilização de software em todas as fases de desenvolvimento. A prática de LPS se aplica no desenvolvimento de uma família de sistemas de software que compartilham um conjunto de características comuns e controladas e atendem a um segmento de mercado em particular.

A Engenharia de Requisitos em abordagens de LPS apresenta determinadas particularidades:

- A atividade de *Levantamento de Requisitos* deve levar em consideração a necessidade de trabalhar com múltiplos sistemas que serão construídos a partir da linha desenvolvida. Assim, o **escopo da linha deve ser identificado** como forma de determinar claramente o segmento de mercado que será atendido e o número desejável de membros da linha.

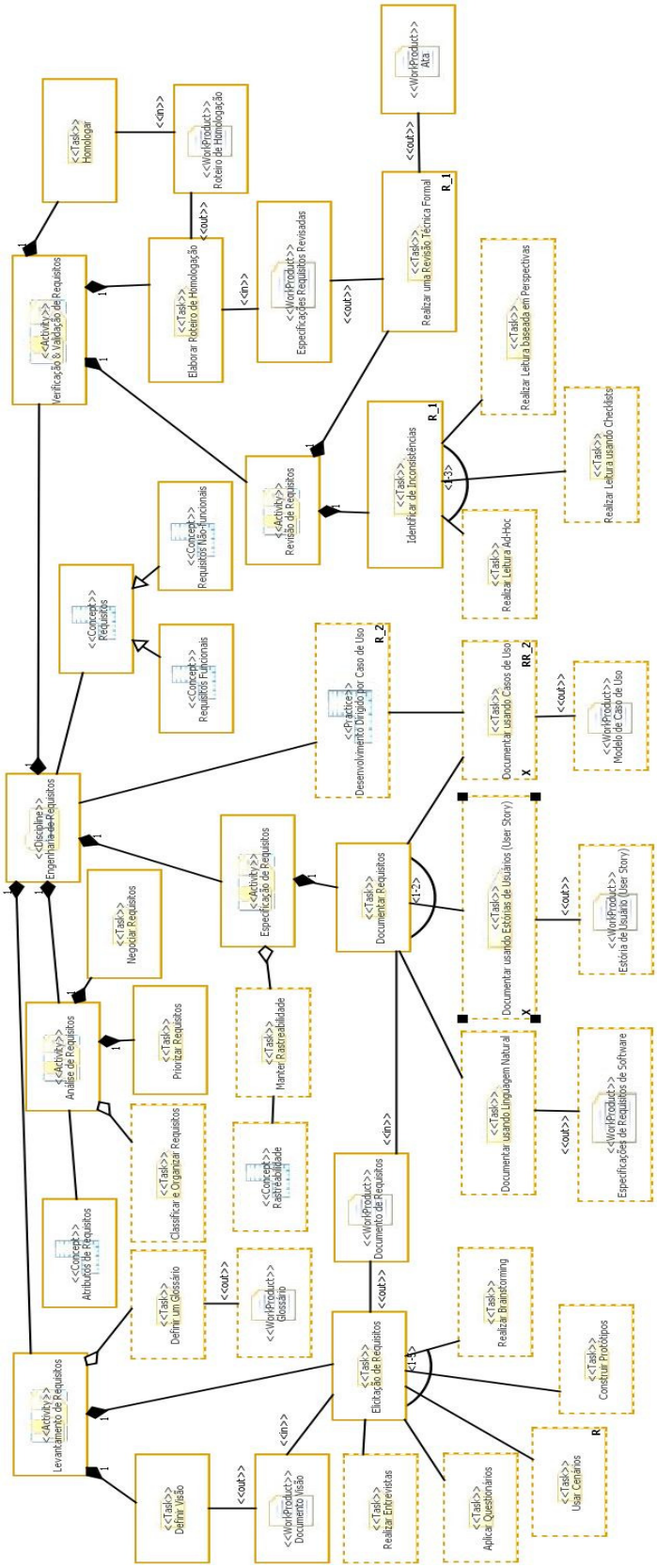
- Na atividade de *Análise de Requisitos*, deve ser considerado o **conceito de variabilidade** fortemente presente na abordagem de LPS. Desta forma, a **tarefa de identificação de similaridades e variabilidades do domínio deve ser conduzida** com o intuito de diferenciar os requisitos comuns a todas as aplicações

possivelmente derivadas a partir de linha dos requisitos que representam especificidades e distinguem os diferentes produtos da linha.

- A descrição deste conhecimento recolhido e analisado pode ser realizada através da adaptação da **modelagem de caso de uso com a adição do conceito de pontos de variação**, local ou região na especificação de requisitos onde uma mudança pode ocorrer. Outra forma de documentação, amplamente utilizada, consiste na **modelagem de características (modelo de *features*)**, onde uma característica pode ser entendida como um aspecto, uma qualidade, um requisito de um sistema ou sistemas de software. No entanto, esses modelos podem ser considerados complementares e podem ser usados conjuntamente. Neste caso, uma **tarefa deve ser conduzida com o propósito de relacionar esses modelos** para obter um melhor aproveitamento desses.

Assim, a modelagem do domínio de processos que contemplam a disciplina de Engenharia de Requisitos aqui apresentada, deve ser refinada com o propósito de incorporar os elementos de processos que representam as particularidades da disciplina de Engenharia de Requisitos em Linha de Produtos de Software. Sua tarefa consiste em modelar esses refinamentos baseados no seu conhecimento sobre a abordagem de LPS e as descrições expostas anteriormente. A modelagem será realizada de forma manual, com o auxílio do preenchimento de tabelas (Anexo X: Tabelas de Especificação da Modelagem Final) e criação de um esboço do modelo final. Para tanto, as definições da modelagem de característica proposta através do meta-modelo e notação *OdysseyProcess-FEX* devem ser observadas. O conjunto de regras de boa formação deve ser consultado.

Modelo de Características: Disciplina de Engenharia de Requisitos



Regras de Composição

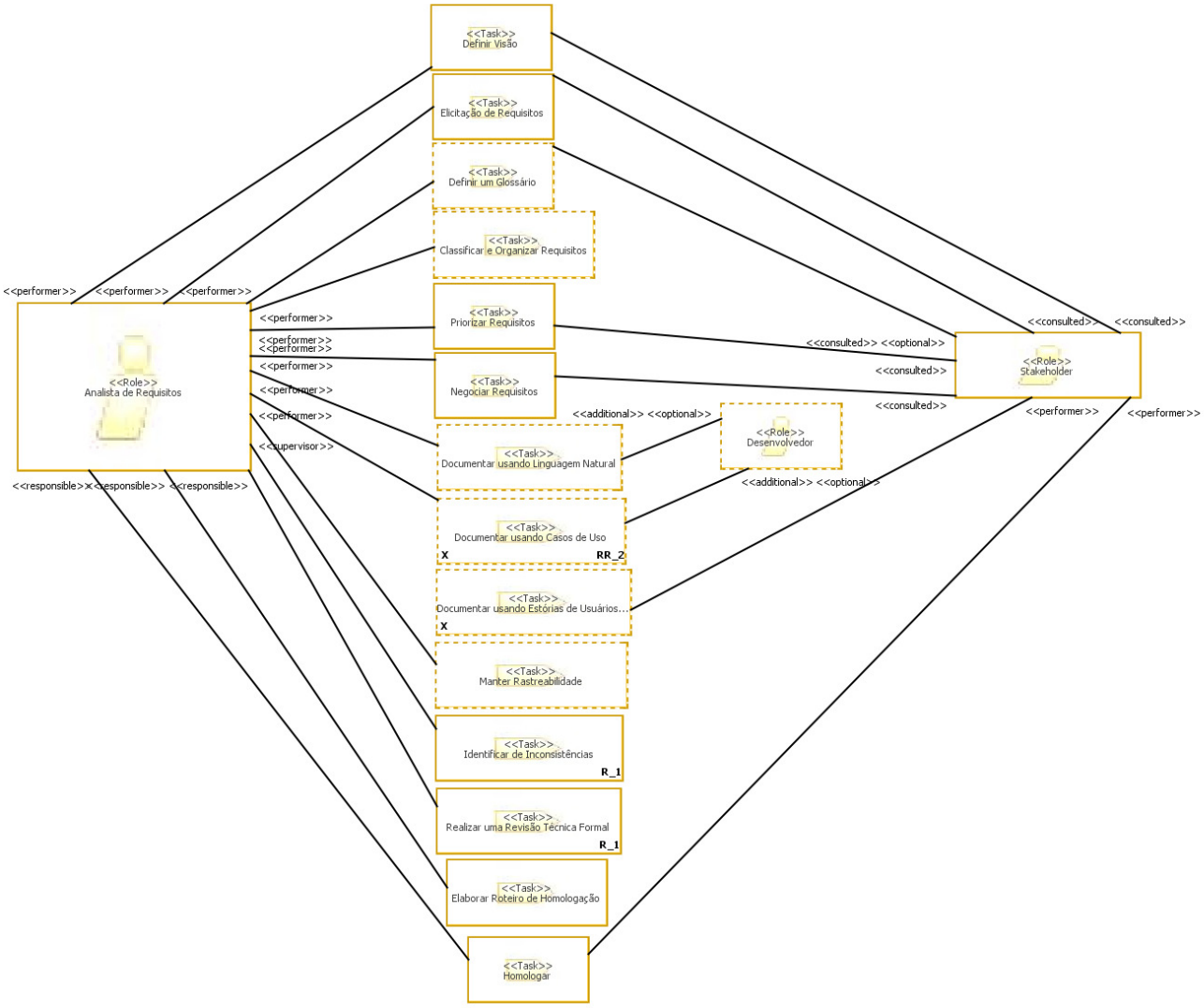
R: Documentar usando Casos de Uso *requer* Usar Cenários

R_1: Realizar Revisão Técnica Formal *requer* Identificação de Inconsistências

R_2: Desenvolvimento Dirigido por Caso de Uso *requer* Documentar usando Casos de Uso

X: Documentar usando Estórias de Usuários (User Story) *excludes* Documentar usando Casos de Uso

Modelo Exemplo pela perspectiva de papéis



Anexo X

FORMULÁRIO DE APOIO À MODELAGEM DE VARIABILIDADES

Taxonomia

Categoria

Sigla	Categoria
DIS	*Característica Disciplina*
ATV	*Característica Atividade*
TAR	*Característica Tarefa*
PRAT	*Característica Prática*
CONC	*Característica Conceito*
PAP	*Característica Papel*
PRTR	*Característica Produto de Trabalho*

Tipo Opcionalidade

Sigla	Tipo Opcionalidade
O	*Opcional*
M	*Mandatória*

Tipo Variabilidade

Sigla	Tipo Variabilidade
I	*Invariante*
PV	*Ponto de Variação*
V	*Variante*

Descrição do tipo de Ação realizada no estudo

Sigla	Tipo Ação
A1	*Inclusão*
A2	*Remoção*
A3	*Alteração*

ID	Nome da Característica	Categoria	Tipo Opcionalidade	Tipo Variabilidade	Ação realizada	Descrição
C1						
C2						
C3						
C4						
C5						
C6						
C7						
C8						
C9						
C10						

ID	Nome da Característica	Categoria	Tipo Opcionalidade	Tipo Variabilidade	Ação realizada	Descrição
C11						
C12						
C13						
C14						
C15						
C16						
C17						
C18						
C19						
C20						

Relacionamentos

Tipos Relacionamento

Sigla	Descrição Relacionamento
ALT	*Alternativo* • Especificar cardinalidade mínima (carMin) e cardinalidade máxima (carMax)
HER	*Herança*
ASS	*Associação*
AGR	*Agregação*
COM	*Composição*
LPT	*LigaçãoPapelTarefa* • Especificar tipo Opcionalidade: (***O***. *Opcional* / ***M***. *Mandatório)* • Especificar estereótipo*: <<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>.*
LPRODT	*LigaçãoProdutoDeTrabalhoTarefa* • Especificar tipo Opcionalidade: (***O***. *Opcional* / ***M***. *Mandatório)* • Especificar estereótipo*: <<in>>, <<out>>, <<inout>>.*
LPPROD	*LigaçãoPapelProdutoDeTrabalho* • Especificar tipo Opcionalidade: (***O***. *Opcional* / ***M***. *Mandatório)* • Especificar estereótipo*: <<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>.*

Descrição do tipo de Ação realizada no estudo

Sigla	Tipo Ação
A1	*Inclusão*
A2	*Remoção*
A3	*Alteração*

ID	Tipo Relacionamento	Característica Origem (ID/Nome)	Característica Destino (ID/Nome)	Ação realizada	Descrição
R1					
R2					
R3					
R4					
R5					
R6					
R7					
R8					
R9					
R10					

ID	Tipo Relacionamento	Característica Origem (ID/Nome)	Característica Destino (ID/Nome)	Ação realizada	Descrição
R11					
R12					
R13					
R14					
R15					
R16					
R17					
R18					
R19					
R20					

Regras de Composição

Tipo da Regra de Composição

Sigla	Descrição Regra
RCI	*Regra de Composição Inclusiva* • Antecedente + *requer* + Consequente
RCE	*Regra de Composição Exclusiva* • Antecedente + *exclui* + Consequente

Operadores

AND, OR, XOR, NOT

Descrição do tipo de Ação realizada no estudo

Sigla	Tipo Ação
A1	*Inclusão*
A2	*Remoção*
A3	*Alteração*

ID	Tipo da Regra de Composição	Regra de Composição	Ação realizada	Descrição
RC1				
RC2				
RC3				
RC4				
RC5				
RC6				
RC7				
RC8				
RC9				
RC10				

Anexo XI

AVALIAÇÃO GERAL

Analise as afirmações a seguir:

1. O material enviado foi suficientemente claro e permitiu a absorção de conhecimento necessário para utilização da notação na execução da tarefa. Justifique.

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

2. O tempo para execução da tarefa foi suficiente. Em caso negativo, ou parcialmente negativo, a falta de tempo está associada à complexidade: (1) da tarefa, (2) do exemplo escolhido, (3) do entendimento dos formulários, (4) do entendimento da notação, (5) outro? Justifique.

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

3. O meta-modelo e a notação *OdysseyProcess-FEX* são de fácil entendimento. Justifique.

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

4. Taxonomia e representação gráfica de elementos de processo de software

4a. Em algum momento existiu a necessidade de representar algum elemento de processo que não foi previsto pela notação. Em caso afirmativo ou parcialmente afirmativo, qual(is)?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

4b. Algum dos elementos de processos identificados não deveria estar representado neste modelo. Em caso afirmativo ou parcialmente afirmativo, qual(is)?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

_____

4c. Os símbolos utilizados para representar os elementos associados aos estereótipos são suficientemente claros, ou seja, permitem uma clara distinção entre os elementos. Em caso negativo, ou parcialmente negativo, o que pode dificultar a identificação dos diferentes elementos?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

4d. A representação de opcionalidades e variabilidades é suficientemente explícita. Em caso negativo, ou parcialmente negativo, o que dificulta a identificação desses conceitos?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

_____

4e. O modelo final gerado foi suficientemente satisfatório. Em caso negativo, ou parcialmente negativo, o que poderia ser melhorado?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

5. Relacionamentos

5a. Em algum momento você sentiu necessidade de incluir um relacionamento não contemplado. Em caso afirmativo, ou parcialmente afirmativo, qual(is)?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

_____

5b. Em algum momento as restrições de participação de categorias de elementos em um relacionamento prejudicaram o estabelecimento de algum relacionamento desejado. Em caso afirmativo, ou parcialmente afirmativo, qual(is)?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

---

6. Suporte Computacional

6a. Você acredita ser viável a modelagem manual de variabilidades de domínios de processos mais extensos e complexos. Em caso negativo, ou parcialmente negativo, aponte como o uso de um suporte computacional poderia auxiliar adequadamente a utilização da notação proposta?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente

_____

_____

_____

6d. O uso de um suporte computacional para guiar a modelagem, usando notificações que identificam e inviabilizem a ocorrência de violação de alguma restrição imposta pelo meta-modelo seria considerado invasivo. (Ou seja, caso uma ação que viole uma das regras definidas pelo meta-modelo ocorra, ela será interrompida associada ao uso de texto informativo da violação ocorrida). Em caso afirmativo, ou parcialmente afirmativo, que ação poderia ser considerada mais apropriada?

□ Discordo totalmente □ Discordo parcialmente □ Indiferente

□ Concordo parcialmente □ Concordo totalmente