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

PESC
Programa de Engenharia de Sistemas e Computação

Relatório Técnico

OdysseyProcess-FEX: Metamodelo e Notação para Representação de Variabilidades de Linha de Processos de Software

Eldânae Nogueira Teixeira
(danny@cos.ufrj.br)

Aline Vasconcelos
(apires@iff.edu.br)

Cláudia Werner
(werner@cos.ufrj.br)

Rio de Janeiro
Março de 2016

OdysseyProcess-FEX: Metamodelo e Notação para Representação de Variabilidades de Linha de Processos de Software

Eldânae Nogueira Teixeira[1], Aline Vasconcelos[2], Cláudia Werner[1]

[1]Programa de Engenharia de Sistemas e Computação (PESC) – COPPE/UFRJ
Caixa Postal 68.511 – 21945-970 – Rio de Janeiro – RJ – Brasil

[2]Instituto Federal Fluminense (IFFluminense)
Dom Bosco - 28030-130 - Campos dos Goytacazes – RJ - Brasil

Resumo

Dada a diversidade de aspectos das organizações e projetos, a tarefa de definição de procesos não é simples. O uso combinado de técnicas de reutilização de processos, como Linha de Processos de Softwrae (LPrS) e Desenvolvimento de Processos Baseado em Componentes (DPBC), visam contribuir para melhorias na produtividade e qualidade de processos e produtos associadas a redução de esforço e custos. Uma LPrS visa explorar os benefícios de reutilizar aspectos comuns que existem em uma família de processo dentro de uma organização e gerenciar suas diversidades. Desta forma, um dos tópicos essenciais que deve ser considerado na cosntrução de LPrSs consiste na identificação e representação do conhecimento de domínio de processos, por meio da análise de suas similaridades e diferenças, isto é, suas variabilidades. Neste trabalho, uma modelagem de variabilidades de LPrS é proposta com o intuito de suportar a ideia de múltiplos níveis semânticos de abstração para especificar uma LPrS de forma complementar, com modelos de características e modelos de componentes, mantendo rastreabilidade. Essa representação é alcançada por meio do metamodelo e notação, ambos denominados *OdysseyProcess-FEX*. O trabalho está inserido em uma metodologia de reutilização de processos de software sistemática que especifica uma Engenharia de LPrS que trata a variabilidade inerente à família de processos e estrutura o conhecimento do domínio por meio de componentes de processos de software.

Abstract

Given the diversity of organizations' and projects' aspects, the process definition task is not simple. The use of combined process reuse techniques, such as Software Process Lines (SPrL) and Component Based Process Definition (CBPD), aims at contributing to improvements in productivity and quality, associated with effort and costs reduction. A SPrL aims to explore the benefits of reusing common aspects that may exist in a process family within an organization and to manage its diversity. Thus, one of the essential issues to be considered in the construction of a SPrL consists of the identification and representation of the software processes domain knowledge, through an analysis of their similarities and differences, i.e., their variabilities. In this work, a SPrL variability modeling is proposed in order to support the idea of multiple semantic abstraction levels to specify the SPrL in a complementary way between the artefacts of feature and component models, with traceability rules. This representation is achieved through a metamodel and a notation, both named *OdysseyProcess-FEX*. This work is inserted in a systematic software process reuse approach by means of a Software Process Line Engineering, which treats variability property inherent to process families and structures the domain knowledge into software process components.

Sumário

1. Introdução

Diante da diversidade existente no universo de projetos de software, a tarefa de definição de processos de software torna-se não trivial. Técnicas de reutilização têm sido aplicadas na área de processos de software com o intuito de auxiliar nesta tarefa, minimizando o esforço e o custo envolvido, além de visar propiciar aumento da qualidade e adequabilidade aos projetos dos processos gerados. Desta forma, a abordagem de Linha de Processos surgiu como uma técnica sistemática de reutilização de processos, que visa aplicar os conceitos de Linha de Produtos de Software em processos. Uma Linha de Processos de Software (LPrS) pode ser definida como um conjunto de processos de software que compartilham um conjunto de características comuns e variáveis, e são desenvolvidas a partir de artefatos (*core assets*), que podem ser reutilizados e combinados entre si, segundo regras de composição e recorte, para compor e adaptar processos de software (NUNES *et al*., 2010; MAGDALENO, 2010; TEIXEIRA, 2011).

Uma abordagem de Engenharia de Linha de Processos de Software (ELPrS) compreende toda a estrutura para desenvolver, utilizar e gerenciar uma Linha de Processos de Software. Esta abordagem é composta por cinco elementos principais (Figura 1): (1) definição da Engenharia de LPrS (ELPrS): processos de desenvolvimento para (EDPS – Engenharia de Domínio de Processos de Software) e com (EAPS – Engenharia de Aplicação de Processos de Software) reutilização de processos de software; (2) representação para modelagem de variabilidade da LPrS nos diferentes níveis de abstração; (3) estabelecimento de mecanismos de mapeamento entre os artefatos dos diferentes níveis de abstração; (4) definição de procedimentos para suportar o agrupamento de componentes de processos da Arquitetura da LPrS; e (5) desenvolvimento de uma infraestrutura de reutilização.

Este documento trata do elemento de representação para modelagem de variabilidade da LPrS nos diferentes níveis de abstração complementares. Cada nível permite uma visão do conjunto de elementos de processos que o compõem e como estes se relacionam. O primeiro nível de abstração representa uma visão conceitual da

LPrS, um alto nível de abstração de fácil entendimento representado por um modelo de domínio que agrega os elementos básicos de processos de software reutilizáveis (unidades de trabalho, papéis, produtos de trabalho e ferramentas) e analisa suas propriedades de opcionalidade e variabilidade no domínio. Uma das formas de representação deste nível consiste em um modelo de características, um modelo amplamente utilizado em LPS e que visa representar o conhecimento do domínio sem a necessidade de explicitar os detalhes de mais baixo nível de abstração. Além disso, os possíveis comportamentos esperados para processos a serem derivados são especificados através de fluxos e sequenciamentos de execução.

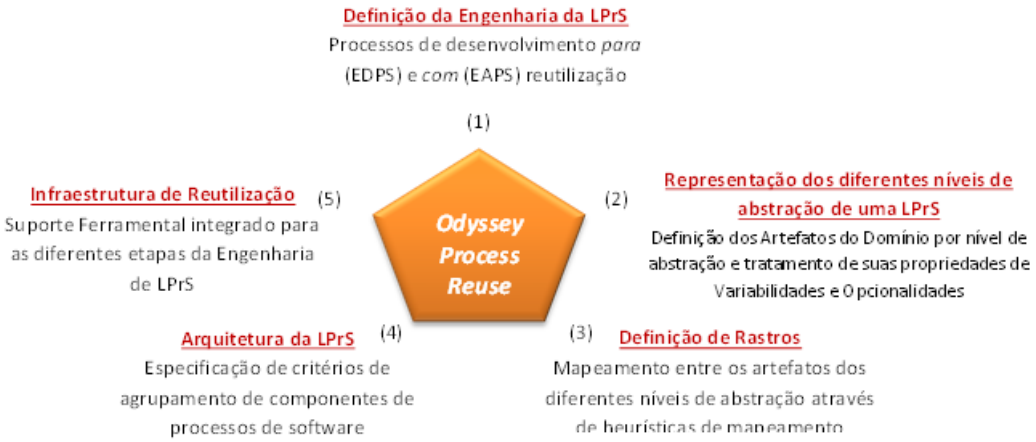


Figura 1 – Elementos principais da abordagem *OdysseyProcessReuse*

De forma adicional, um modelo de contexto pode ser especificado para caracterizar entidades e situações de contexto que apoiem a seleção de artefatos de processos reutilizáveis segundo regras de contexto estabelecidas.

O segundo nível complementa o modelo do domínio representando a LPrS através de uma visão modular organizada por componentes e tendo como representação final uma Arquitetura de Componentes da LPrS. Tal arquitetura representa um arcabouço da LPrS através do estabelecimentos das relações entre os componentes de processos de software identificados que direciona a definição de processos específicos de projetos na EAPS, garantindo a consistência de composição dos elementos de processos reutilizáveis selecionados.

Tal modelagem, proposta neste trabalho para representação da LPrS em dois níveis, deve ser realizada segundo o metamodelo e a notação *OdysseyProcess-FEX*, inicialmente definidos em TEIXEIRA (2011), ambos evoluídos neste trabalho para melhor atender os requisitos dos níveis de características e de componentes. O metamodelo visa explicitar os conceitos e as propriedades dos elementos e a notação especifica a representação gráfica dos conceitos formalizados através do metamodelo aqui definido, com o intuito de apoiar o usuário na compreensão e bom uso da notação. O nível de análise (*Process Features View*) está descrito através de três pacotes: a) Principal, que representa a parte da definição da taxonomia dos componentes, b) Relacionamento, que especifica propriedades dos relacionamentos existentes no modelo de componentes; e c) Regras de Composição, que especifica as regras de dependência e exclusividade entre os componentes de um modelo. O nível de projeto (*Process Components View*) descreve os elementos do metamodelo de componentes em um único pacote.

A estrutura de especificação de cada pacote é realizada através de uma explicação detalhada de cada elemento pertencente ao pacote 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 metamodelo. Item de descrição incluído no Pacote de Relacionamentos e no Pacote Regras de Composição. Descreve o papel dos elementos envolvidos nos relacionamentos estabelecidos;

• *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 metamodelo;

• *Notação*: especifica a descrição gráfica de representação visual do elemento definido; e

• *Exemplo*: especifica a descrição de um exemplo aplicando o elemento definido.

2. LPrS – Nível Análise – Modelagem de Características de Processos de Software (*Process Features View*)

2.1. Nível Análise (*Process Features View*) - Pacote Principal

O pacote Principal (Figura 2) do metamodelo *OdysseyProcess-FEX* no nível de análise (*Process Features View*) descreve a taxonomia dos elementos que o constituem, definindo os elementos de processos de software reutilizáveis descritos como características, seus atributos e propriedades. São definidas as características de processo de software (*Process Feature*): Unidades de Trabalho (*Work Unit*), compostas pelos elementos Atividade (*Activity*) e Tarefa (*Task*); Papel (*Role*); Produto de Trabalho (*WorkProduct*); e Ferramenta (*Tool*). Unidades de Trabalho são elementos de processo que devem ser executados com o objetivo de agregar valor ao processo gerando resultados. Atividades consistem em grupos lógicos de unidades de trabalho menores, que correspondem a outras atividades ou tarefas. Uma Tarefa pode ser descrita através dos passos que a compõem. Um passo representa uma ação a ser executada para alcançar o objetivo de uma tarefa. Ferramentas podem ser associadas a unidades de trabalho, representando algum suporte ferramental para sua execução. Papéis estabelecem um conjunto de habilidades e conhecimentos requeridos por um indivíduo ou por um grupo para executar uma unidade de trabalho. Produtos de Trabalho descrevem artefatos a serem consumidos, modificados ou produzidos pelas unidades de trabalho. O elemento de nota (classe *Comment*) também é apresentado neste pacote como forma de agregar informações adicionais para entendimento do

domínio. A classe Disciplina (*Discipline*) representa uma forma de organização do conteúdo de unidades de trabalho em pacotes.

A seguir uma descrição detalhada de cada elemento deste pacote é apresentada, seguindo um modelo organizado através de sete seções: Descrição, Hierarquia, Atributos, Associações, Restrições, Notação e Exemplo.

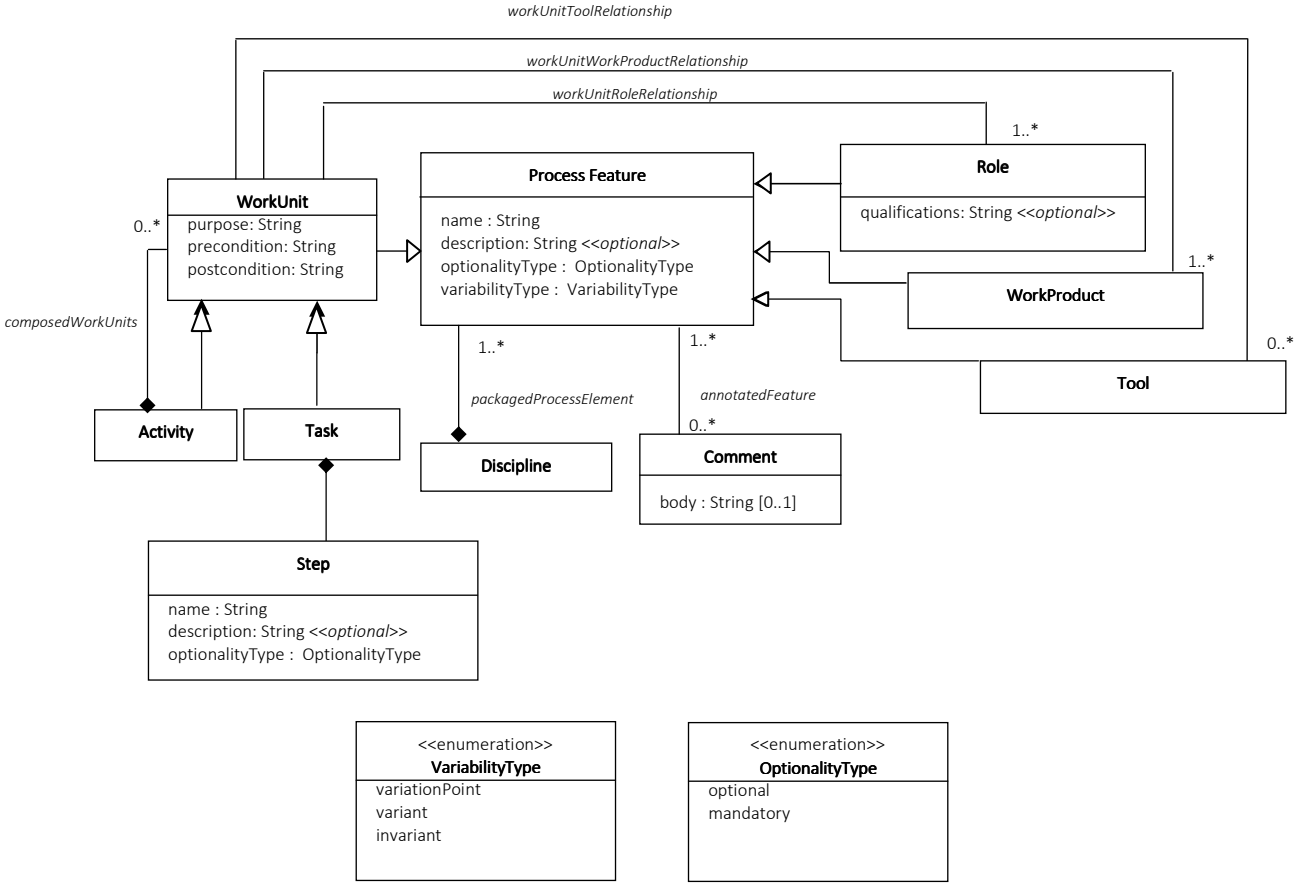


Figura 2- Metamodelo *OdysseyProcess-FEX*: Pacote Principal (*Process Features View*)

2.1.1. *ProcessFeature* (Característica de Processo)

Descrição

Representa elementos estruturais de processo de software passíveis de reutilização na definição de novos processos.

Hierarquia

Subclasses: *WorkUnit (UnidadeDeTrabalho), Role (Papel), WorkProduct (Produto de Trabalho) e Tool (Ferramenta).*

Atributos

- ***name* (nome)**: String[1] – atributo que define o nome do elemento.
- ***description* (descrição)**: String[0..1] – atributo que permite uma descrição textual sobre o elemento.
- ***optionalityType* (tipoOpcionalidade)**: *OptionalityType*[1] (TipoOpcionalidade[1]) – atributo que define a classificação do elemento de processo quanto a sua opcionalidade. Pode assumir os valores *mandatory* (mandatório) e *optional*(opcional). Indica se o elemento é mandatório no domínio, isto é, se o elemento estará presente em todos os processos instanciados a partir do modelo, ou se o elemento é opcional. O tipo de opcionalidade é representado pela lista enumerada *OptionalityType*(TipoOpcionalidade). Default: *optional*(opcional).
- ***variabilityType* (tipoVariabilidade)**: *VariabilityType*[1] (TipoVariabilidade[1]) – atributo que indica o tipo de variabilidade que o elemento apresenta. Pode assumir os valores *invariant* (invariante), variant (variante) e *variationpoint* (ponto de variação). O tipo de variabilidade é representado pela lista enumerada *VariabilityType (TipoVariabilidade)*. Default: *invariant* (invariante).

Associações

Sem associações específicas.

Restrições

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

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

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

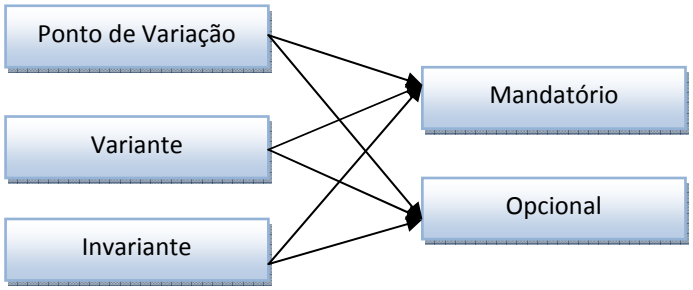


Figura 3- Classificação ortogonal dos elementos

Notação

Em uma modelagem de característica, cada característica será definida como um elemento de processo e será representada por uma caixa retangular com um ícone e um estereótipo designando o tipo/categoria do elemento. Os ícones foram derivados da representação do SPEM v2.0 pacote Conteúdo do Método. Os estereótipos visam auxiliar na identificação visual dos elementos. Desta forma, a representação visual será apresentada em cada elemento do pacote abaixo.

Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

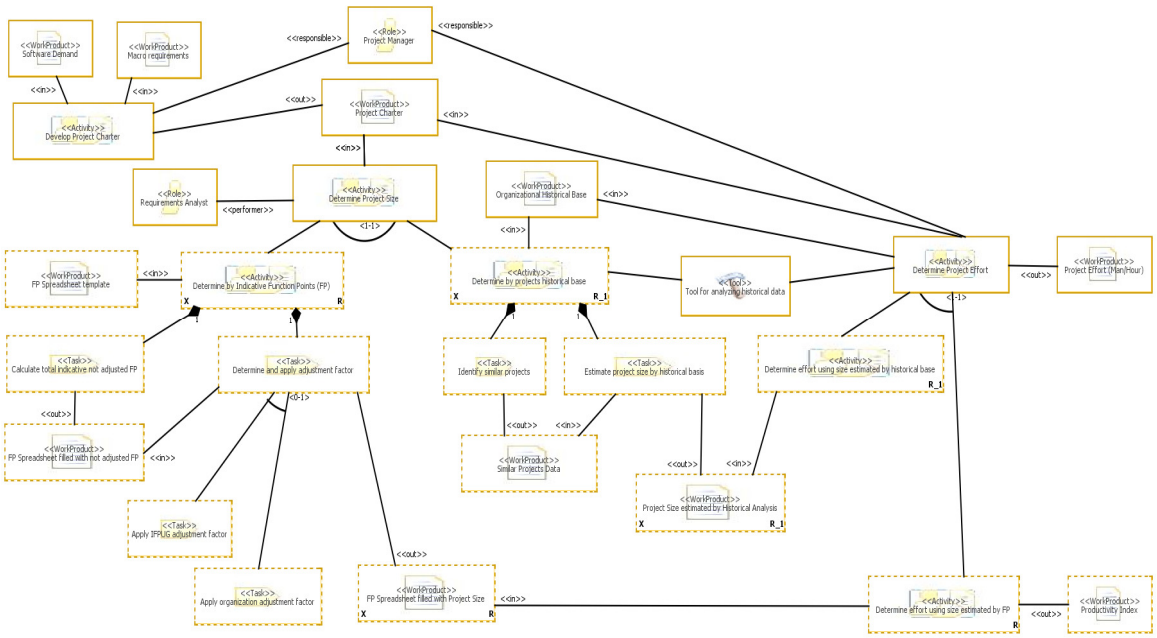


Figura 4 - Representação Geral de Modelo de Características de Processo de Software

2.1.2. *VariabilityType* (TipoVariabilidade)

Descrição

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

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

Hierarquia

Sem hierarquia específica.

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

[1] As classificações de elementos com relação à variabilidade são mutuamente excludentes entre si. Desta forma, um elemento não pode receber simultaneamente dois tipos diferentes de classificação quanto à variabilidade.

Notação

A classificação de variabilidade é graficamente representada através do relacionamento alternativo conforme abaixo. Para maiores detalhes, a descrição do relacionamento deve ser consultada.

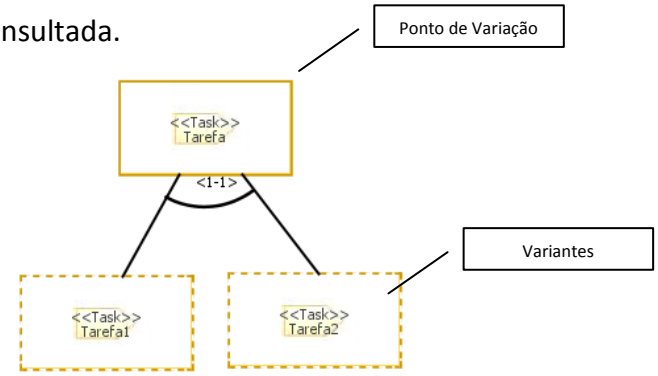


Figura 5 – Representação Gráfica da Classificação de Variabilidade de Características de Processos de Software

Exemplo

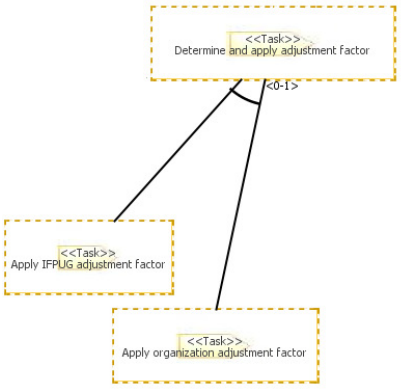


Figura 6 – Representação de Exemplo de Ponto de variação e variantes no domínio de Planejamento de Projetos de Software

2.1.3. *OptionalityType* (TipoOpcionalidade)

Descrição

OptionalityType (TipoOpcionalidade) é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo de opcionalidade presente em uma Característica de Processo. Pode assumir os seguintes valores: *mandatory* (mandatório) e *optional* (opcional).

- *Mandatório*: representa elemento de processo que deverá estar presente em todos os processos instanciados a partir do modelo.
- *Opcional*: representa elemento de processo que pode ou não estar presente em processos instanciados a partir do modelo.

Hierarquia

Sem hierarquia específica.

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

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

Notação

A classificação de opcionalidade é graficamente representada através de um retângulo formado por uma linha tracejada para as características classificadas como opcionais e uma linha contínua para as características mandatórias:

Classificação de Opcionalidade de Elemento de Processo	Representação Gráfica
Característica Mandatória	
Característica Opcional	

Exemplo

O exemplo abaixo se refere a uma parte do domínio de Planejamento de Projetos de Software.

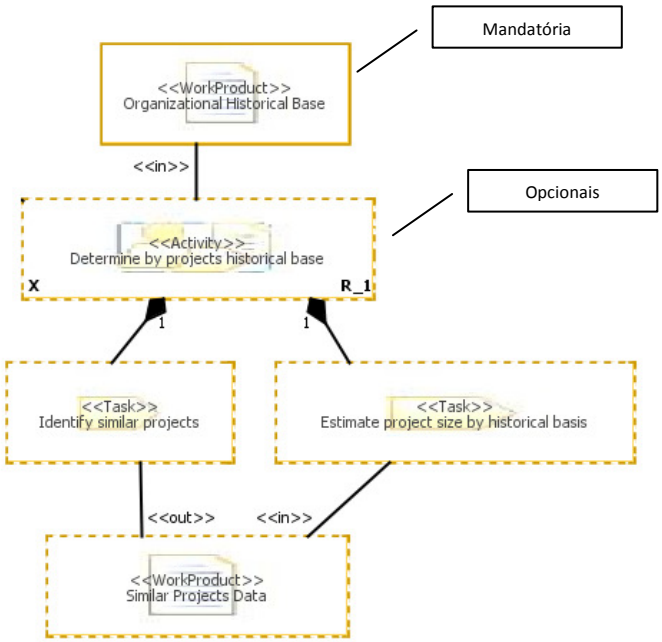


Figura 7 – Exemplos de representação gráfica de características de processo mandatórias e opcionais

2.1.4. *WorkUnit* (Unidade de Trabalho)

Descrição

Elemento que representa o conceito de uma unidade de trabalho em diferentes níveis de granularidade (atividades e tarefas). Pode ser entendida como uma unidade composta por ações que executam alguma transformação em artefatos do processo e podem ser atribuídas a determinados papéis.

Hierarquia

Superclasse: *ProcessFeature* (Característica de Processo)

Subclasses: *Activity* (Atividade) e *Task* (Tarefa)

Atributos

- ***purpose* (propósito)**: String [1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela unidade de trabalho a ser executada para alcançar determinado valor de resultado dentro do processo.
- ***precondition* (pré-condição)**: String [1] – atributo que estabelece um conjunto de restrições: condições que permitem o início da execução da unidade de trabalho.
- ***postcondition* (pós-condição)**: String [1] – atributo que estabelece um conjunto de restrições: condições que permitem concluir a execução da unidade de trabalho, estabelecendo os resultados esperados.

Associações

- ***workUnitRoleRelation*** (relação entre papel e unidade de trabalho): WorkUnitRoleRelationship [1..*] (LigaçãoPapelUnidadeDeTrabalho [1..*]) – especifica relações de responsabilidades entre unidades de trabalho e papéis.
- ***workUnitWorkProductRelation*** (relação entre produto de trabalho e unidade de trabalho): WorkUnitWorkProductRelationship [1..*] (LigaçãoProdutoDeTrabalhoUnidadeDeTrabalho [1..*]) – especifica relações entre unidades de trabalho e produtos de trabalho.
- ***workUnitToolRelation*** (relação entre produto de trabalho e ferramenta): WorkUnitToolRelationship [0..*] (LigaçãoFerramentaUnidadeDeTrabalho [1..*]) – especifica relações entre unidades de trabalho e ferramentas.

As classes *WorkUnitRoleRelation, WorkUnitWorkProductRelation* e *workUnitToolRelation* pertencem ao pacote Relacionamentos.

Restrições

Sem restrições específicas.

Notação

A representação gráfica é específica das subclasses atividade e tarefa.

Exemplo

Olhar exemplo nas subclasses atividade e tarefa.

2.1.5. *Activity* (Atividade)

Descrição

Elemento de processo que representa o agrupamento de unidades de trabalhos menores, representadas por outras atividades ou por tarefas, e que produzem um resultado esperado para o processo.

Hierarquia

Superclasse: *WorkUnit (UnidadeDeTrabalho)*

Atributos

- ***purpose* (propósito)**: String [1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela unidade de trabalho a ser executada para alcançar determinado valor de resultado dentro do processo.
- ***precondition* (pré-condição)**: String [1] – atributo que estabelece um conjunto de restrições: condições que permitem o início da execução da unidade de trabalho.
- ***postcondition* (pós-condição)**: String [1] – atributo que estabelece um conjunto de restrições: condições que permitem concluir a execução da unidade de trabalho, estabelecendo os resultados esperados.

Associações

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

da atividade por outras unidades de trabalho.

Restrições

[1] Uma *Activity* (Atividade) representa o agrupamento de trabalho definido por outras unidades de trabalho (*WorkUnit*) que são descritas por elementos da categoria atividade (*Acitivity*) ou por unidades ainda menores representadas por elementos da categoria tarefa (*Task*).

Notação

Característica desta categoria apresentam o ícone e o estereótipo descrito abaixo.

Elemento de Processo	Ícone	*Estereótipo*
Atividade		***<<activity>>***

Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

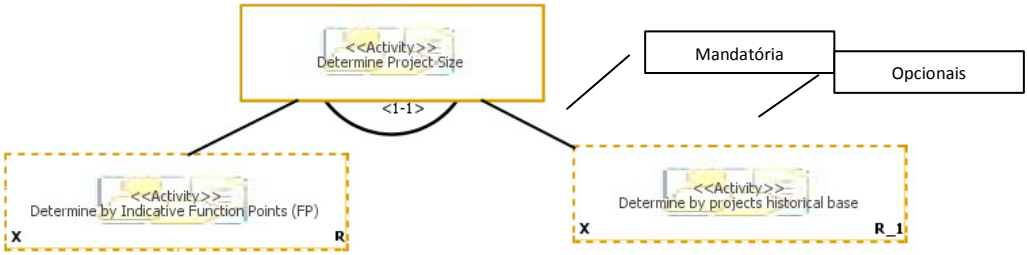


Figura 8 - Representação de Características de Processo de Software da Categoria Atividade

2.1.6. *Task* (Tarefa)

Descrição

Elemento de processo que representa uma unidade de trabalho descrita através de passos. Sua granularidade é definida como o menor agrupamento possível de ações.

Hierarquia

Superclasse: *WorkUnit (UnidadeDeTrabalho)*

Atributos

- ***purpose* (propósito)**: String [1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pela unidade de trabalho a ser executada

para alcançar determinado valor de resultado dentro do processo.

- ***precondition* (pré-condição)**: String [1] – atributo que estabelece um conjunto de restrições: condições que permitem o início da execução da unidade de trabalho.
- ***postcondition* (pós-condição)**: String [1] – atributo que estabelece um conjunto de restrições: condições que permitem concluir a execução da unidade de trabalho, estabelecendo os resultados esperados.

Associações

- ***composedTask*** (tarefa composta) : *Step* [1..*] (Passo [1..*]) – especifica relações de composição da tarefa por um conjunto de passos.

Restrições

Sem restrições específicas.

Notação

Característica desta categoria apresentam o ícone e o estereótipo descrito abaixo.

Elemento de Processo	Ícone	*Estereótipo*
Tarefa		***<<task>>***

Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

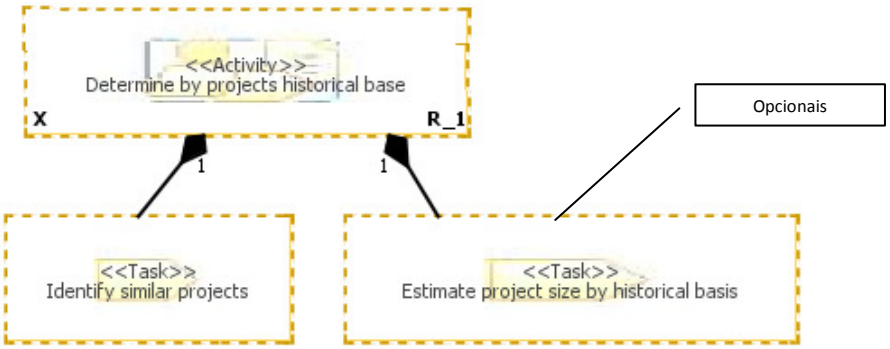


Figura 9 - Representação de Características de Processo de Software da Categoria Tarefa

2.1.7. *Step* (Passo)

Descrição

Corresponde a uma única ação atômica que compõem o trabalho a ser executado por uma tarefa. O fluxo de passos descreve como atingir o propósito definido pela tarefa a qual pertence.

Hierarquia

Sem hierarquia específica.

Atributos

- ***name* (nome)**: String [1] – atributo que define o nome do passo.
- ***description* (descrição)**: String [0..1] – atributo que permite uma descrição textual sobre a ação a ser executada.
- ***optionalityType* (tipoOpcionalidade)**: *OptionalityType* [1] (TipoOpcionalidade[1]) – atributo que define a classificação do passo quanto a sua opcionalidade durante a execução da tarefa ao qual está associado. Pode assumir os valores *mandatory* (mandatório) e *optional* (opcional). Indica se o passo é mandatório no domínio, isto é, se estará presente todas as vezes que a tarefa associada é instanciada a partir do modelo, ou se sua participação na execução da tarefa é opcional, indicando a possibilidade de alternativas no fluxo de execução de uma tarefa. O tipo de opcionalidade é representado pela lista enumerada *OptionalityType* (TipoOpcionalidade). Default: *optional* (opcional).

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Apesar da relevância da informação para o entendimento da execução da tarefa, a granularidade de um passo pode ser considerada pequena para representação em um modelo de característica que representa o domínio como um todo. Desta forma, diagramas complementares podem ser definidos associados ao modelo principal para visualização gráfica da associação de uma tarefa com seus respectivos passos. A representação segue os ícones do SPEM v2.0 para Tarefa e Passos associados do Pacote Conteúdo de Método.

Exemplo

Elemento de Processo Tarefa representado como uma característica do domínio	Representação Complementar do Elemento Tarefa associado a seus respectivos passos de execução
. 	

2.1.8. *Role* (Papel)

Descrição

Elemento de processo 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: *ProcessFeature* (Característica de Processo)

Atributos

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

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Característica desta categoria apresentam o ícone e o estereótipo descrito abaixo.

Elemento de Processo	Ícone	*Estereótipo*
Papel		*<<role>>*

Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

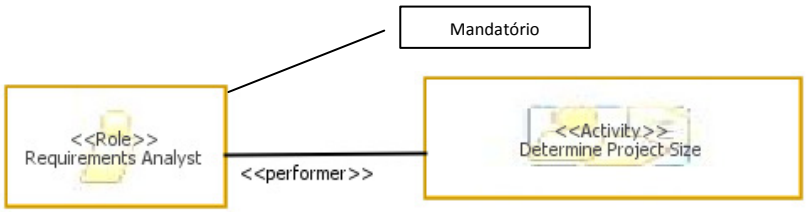


Figura 10 - Representação de Características de Processo de Software da Categoria Papel

2.1.9. *WorkProduct* (ProdutoDeTrabalho)

Descrição

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

Hierarquia

Superclasse: *ProcessFeature* (Característica de Processo)

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Característica desta categoria apresentam o ícone e o estereótipo descrito abaixo.

Elemento de Processo	Ícone	*Estereótipo*
Produto de Trabalho		***<<workproduct>>***

Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

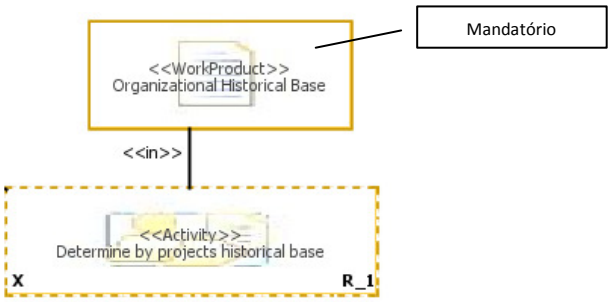


Figura 11 - Representação de Características de Processo de Software da Categoria Produto de Trabalho

2.1.10. Tool (Ferramenta)

Descrição

Elemento de processo que descreve capacidades de uma unidade de automação para apoiar a exceção da unidade de trabalho associada.

Hierarquia

Superclasse: *ProcessFeature* (Característica de Processo)

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Característica desta categoria apresentam o ícone e o estereótipo descrito abaixo.

Elemento de Processo	Ícone	*Estereótipo*
Ferramenta		***<<tool>>***

Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

Figura 12 - Representação de Características de Processo de Software da Categoria Ferramenta

2.1.11. *Discipline* (Disciplina)

Descrição

Representa o conceito de pacote para organização do domínio em escopos menores ou subdomínios. Elemento que representa uma categorização de trabalho baseado em uma similaridade de interesses e propósito de resultados. A disciplina é um conjunto de elementos de processos que estão relacionadas com uma grande área de interesse no âmbito do domínio como um todo.

Hierarquia

Sem hierarquia específica.

Atributos

Sem atributos.

Associações

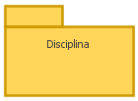
- ***packagedProcessFeatures*** (ElementosDeProcessosEmpacotados): *ProcessFeature* [*] (CaracterísticaDeProcesso [*]) – especifica os elementos de processos que foram agrupados e pertencem a uma disciplina.

Restrições

Sem restrições específicas.

Notação

O elemento que representa uma disciplina será representado pelo ícone pacote da UML.

- Ícone: 

Exemplo

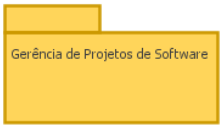


Figura 13 - Representação do Elemento Disciplina de Gerência de Projetos de Software

2.1.12. *Comment* (Nota)

Descrição

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

Hierarquia

Sem hierarquia específica.

Atributos

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

nota associada a um elemento de processo.

Associações

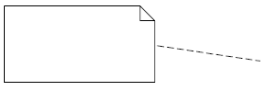
- ***annotatedFeature*** (característica comentada) : *ProcessFeature* [*] (CaracterísticaDeProcesso [*]) – referência as elementos que possuem comentários associados.

Restrições

Sem restrições específicas.

Notação

O elemento Nota que representa uma anotação textual será representada pelo ícone de Comentário da UML.

Elemento de Processo	Representação Gráfica
Nota	

Exemplo

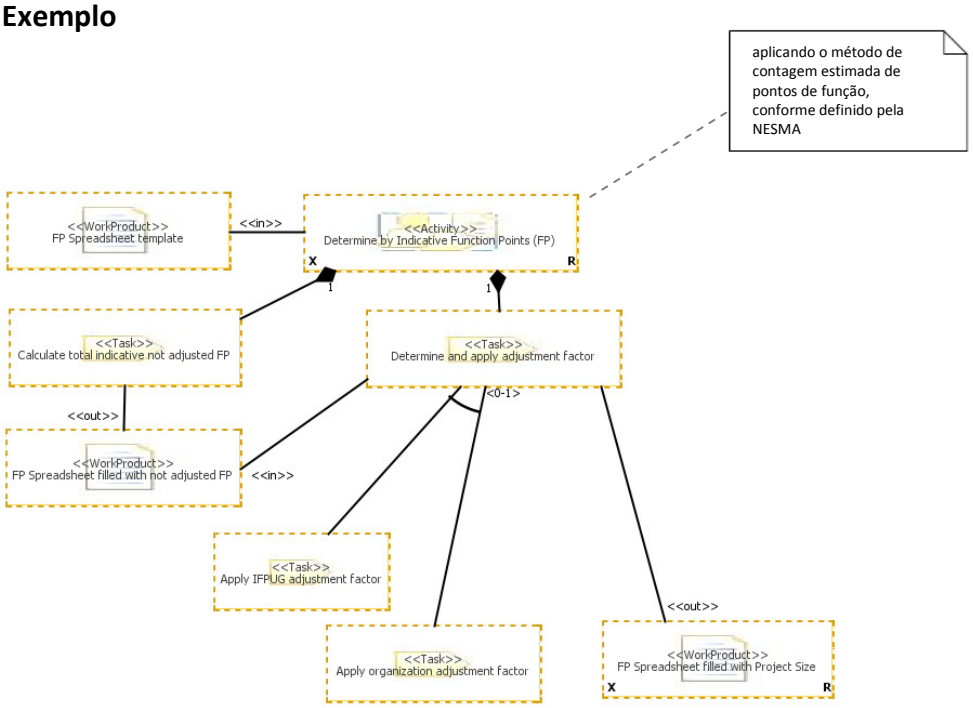


Figura 14 - Exemplo Representação do Elemento Nota

2.2. Nível Análise (*Process Features View*) - Pacote Relacionamentos

O pacote Relacionamentos (*Relationships*) consiste na representação das relações disponibilizadas entre os elementos de processos descritos no pacote Principal. Está estruturado na classe abstrata *Relationship* (Relacionamento) e suas especializações: *AlternativeRelationship* (Alternativo)*; Association* (Associação) e suas especializações – *Aggregation* (Agregação) *e Composition* (Composição)*; WorkUnitRoleRelationship* (LigaçãoUnidadeDeTrabalhoPapel)*; WorkUnitWorkProductRelationship* (LigaçãoUnidadeDeTrabalhoProdutoDeTrabalho)*; WorkProductRoleRelationship* (LigaçãoProdutoDeTrabalhoPapel); *WorkUnitToolRelationship* (LigaçãoUnidadeDeTrabalhoFerramenta) *e ControlFlow*(Fluxo de Controle) (Figura 15).

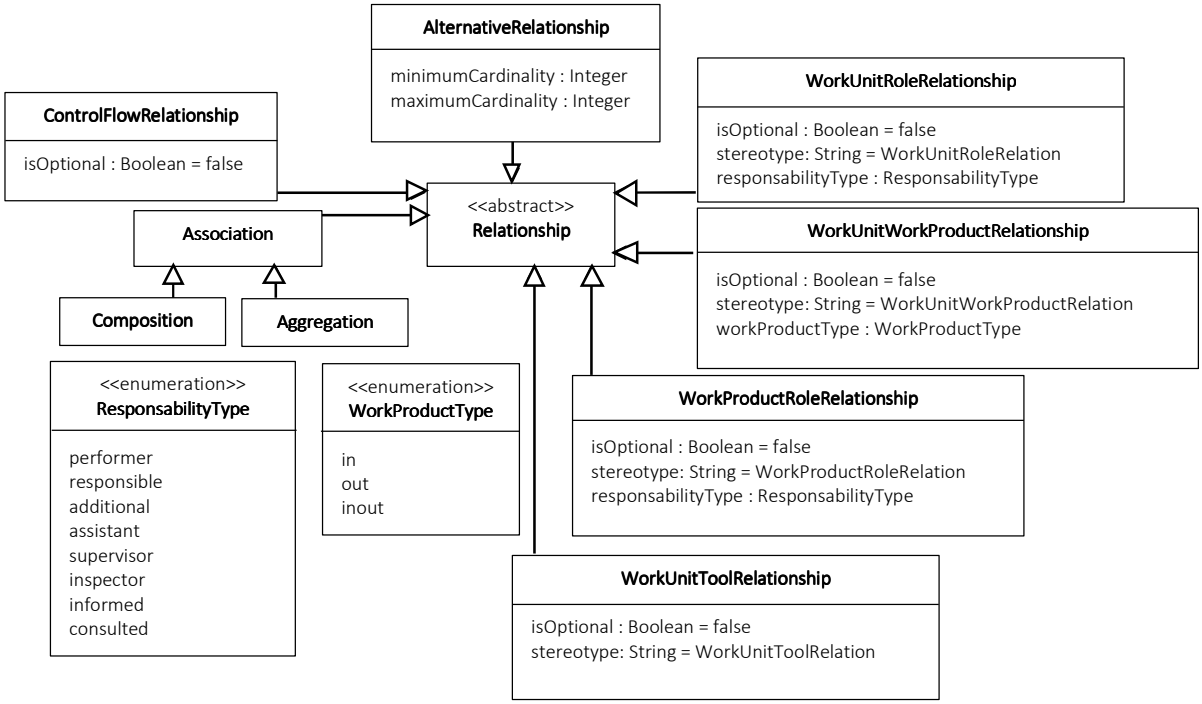


Figura 15- Metamodelo *OdysseyProcess-FEX*: Pacote Relacionamentos (*Relationship*)

A seguir uma descrição detalhada de cada elemento deste pacote é apresentada, seguindo um modelo organizado através de sete seções: Descrição, Hierarquia, Atributos, Associações, Restrições, Notação e Exemplo.

2.2.1. *Relationship* (Relacionamentos)

Descrição

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

Hierarquia

SubClasses*: AlternativeRelationship (Alternativo), Association (Associação) e subclasses Aggregation (Agregação) e Composition (Composição), WorkUnitRoleRelationship* (LigaçãoUnidadeDeTrabalhoPapel)*; WorkUnitWorkProductRelationship* (LigaçãoUnidadeDeTrabalhoProdutoDeTrabalho)*; WorkProductRoleRelationship* (LigaçãoProdutoDeTrabalhoPapel); *WorkUnitToolRelationship* (LigaçãoUnidadeDeTrabalhoFerramenta) *e ControlFlow* (Fluxo de Controle).

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Sem notação específica. A representação será específica para cada subclasse definida.

Exemplo

Olhar exemplo da Figura 4.

2.2.2. *AlternativeRelationship* (Alternativo)

Descrição

Relacionamento existente entre um ponto de variação e suas variantes. Denota a pertinência de uma variante a um determinado ponto de variação (OLIVEIRA *et al.,* 2005). A Figura 16 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de elementos que podem participar desse tipo de relacionamento. As restrições quanto à combinação das categorias de elementos são apresentadas no item [11] do campo Restrições deste relacionamento.

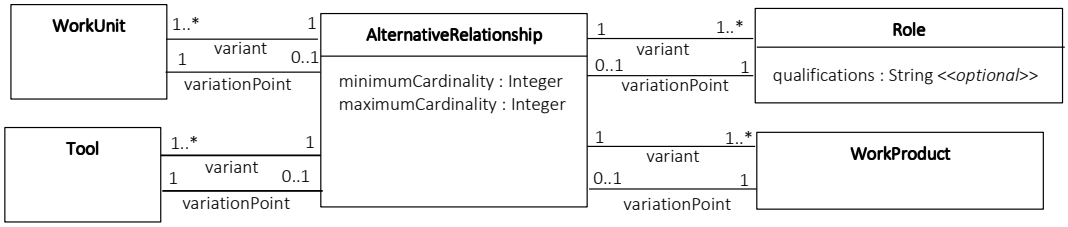


Figura 16- Metamodelo *OdysseyProcess-FEX*: *AlternativeRelationship* (Alternativo)

Hierarquia

Superclasse: *Relationship* (Relacionamento).

Atributos

Cardinalidade: atributo que define o número de instâncias de variantes (sem repetição do mesmo elemento) que poderão ocupar o ponto de variação. As cardinalidades são representadas através da definição de um intervalo, em que o valor mínimo e máximo são determinados, ou seja, podem ser representadas através de intervalos fixos.

Desta forma, este atributo é definido pela combinação dos seguintes atributos:

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

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

Associações

- ***variant* (variante)**: *ProcessFeature* [1..*] – Referencia os elementos do tipo Variante que fazem parte do relacionamento Alternativo.
- ***variationPoint* (pontoVariação)**: *ProcessFeature* [1] - Referencia o elemento de processo do tipo Ponto de Variação que faz parte do relacionamento Alternativo.

Restrições

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

[2] Elementos de processos classificados como Ponto de Variação: *ProcessFeature.variabilityType = variationPoint* (CaracterísticaDeProcesso*.tipoVariabilidade = pontoVariação*) são caracterizados como origem do relacionamento do tipo *Alternativo*.

[3] Elementos de processos classificados como Variante: *ProcessFeature.variabilityType = variant*(CaracterísticaDeProcesso*.tipoVariabilidade = variante*) são caracterizados como destino do relacionamento do tipo *Alternativo*.

[4] Elementos de processos classificados como Variantes: *ProcessFeature.variabilityType = variant*(CaracterísticaDeProcesso*.tipoVariabilidade = variante*) e Mandatórios: *ProcessFeature.optionalityType =*

mandatory(CaracterísticaDeProcesso*.tipoOpcionalidade = mandatório*) devem ter um relacionamento do tipo *Alternativo* com outro elemento de processo classificada como Ponto de Variação: *ProcessFeature.variabilityType = variationPoint*(CaracterísticaDeProcesso*.tipoVariabilidade = pontoVariação*) NECESSARIAMENTE Mandatório: *ProcessFeature.optionalityType = mandatory* (CaracterísticaDeProcesso*.tipoOpcionalidade = mandatório*).

[5] Elementos de processos classificados como Variantes: *ProcessFeature.variabilityType = variant*(CaracterísticaDeProcesso*.tipoVariabilidade = variante*) e Opcionais: *ProcessFeature.optionalityType = optional*(CaracterísticaDeProcesso*.tipoOpcionalidade = opcional*) podem ter um relacionamento do tipo Alternativo com outro elemento de processo classificado como Ponto de Variação: *ProcessFeature.variabilityType = variationPoint*(CaracterísticaDeProcesso*.tipoVariabilidade = pontoVariação*) que seja Mandatório: *ProcessFeature.optionalityType = mandatory*(CaracterísticaDeProcesso*.tipoOpcionalidade = mandatório*). Neste caso, a obrigatoriedade do ponto de variação indica que pelo menos uma das variantes ligadas a ele deve ser selecionada no processo específico de projeto derivado.

[6] Elementos de processos classificados como Ponto de Variação: *ProcessFeature.variabilityType = variationPoint*(CaracterísticaDeProcesso*.tipoVariabilidade = pontoVariação*) e Opcionais: *ProcessFeature.optionalityType = optional*(CaracterísticaDeProcesso*.tipoOpcionalidade = opcional*) devem ter um relacionamento do tipo *Alternativo* com outros elementos classificados como Variantes: *ProcessFeature.variabilityType = variant*(CaracterísticaDeProcesso*.tipoVariabilidade = variante*) NECESSARIAMENTE Opcionais: *ProcessFeature.optionalityType = optional*(CaracterísticaDeProcesso*.tipoOpcionalidade = opcional*).

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

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

[9] Relacionamentos Alternativos com cardinalidade mínima com valor superior a zero (*minimumCardinality*> 0), representam relacionamentos em que os elementos que representam o ponto de variação são classificados como mandatórios.

[10] Não pode ocorrer repetição do mesmo tipo de elemento como instância de um ponto de variação, ou seja, o número de instâncias de variantes atribuído como cardinalidade não permite ocupação por repetição de elementos como alternativa para configuração do ponto de variação.

[11] O relacionamento Alternativo só poderá ocorrer entre as seguintes combinações de categorias de elementos de processos (Tabela 1).

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

Tipo de Relacionamento	**Categoria de elemento de processo**	**Categoria de elemento de processo**
	Origem: ***Ponto de Variação***	*Destino:* ***Variante***
Alternativo	Atividade	Atividade
	Tarefa	Tarefa
	Papel	Papel
	ProdutoDeTrabalho	ProdutoDeTrabalho
	Ferramenta	Ferramenta

Notação

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.

- Ícone: 

Exemplo

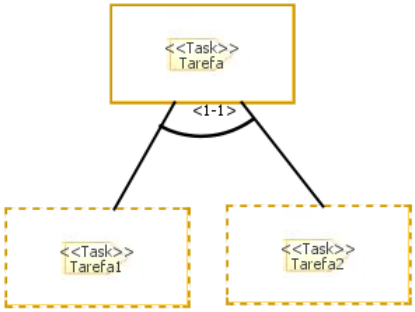


Figura 17- Exemplo de Representação Gráfica do Relacionamento Alternativo *Association* (Associação)

2.2.3. 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 dois elementos. O relacionamento pode possuir um estereótipo que especifica o tipo da ligação.

As restrições quanto à combinação das categorias de elementos que podem ser aplicadas aos relacionamentos Composição e Agregação são apresentadas no item [3] do campo Restrições.

Hierarquia

Superclasse: *Relationship* (Relacionamento).

Subclasses: *Aggregation* (Agregação) e *Composition* (Composição)

Atributos

Sem atributos específicos.

Associações

Sem associações específicas.

Restrições

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

[2] Somente associações binárias podem ser um relacionamento *Association* (Associação).

[3] O relacionamento *Composition* (Composição) e o relacionamento *Aggregation* (Agregação) só poderão ocorrer entre as seguintes combinações de categorias de elementos de processo (Tabela 2):

Tabela 2 - Restrições das combinações de categorias de elementos

Tipo de Relacionamento	Categoria de elemento de processo	Categoria de elemento de processo
	Origem: ExtremidadeTodo	*Destino*: ExtremidadeParte
Composition (Composição)	Atividade	Atividade
	Atividade	Tarefa
	Papel	Papel
Aggregation (Agregação)	ProdutoDeTrabalho	ProdutoDeTrabalho
	Ferramenta	Ferramenta

Notação

A representação gráfica é especificada nas subclasses (seções 2.2.4 e 2.2.5). Olhar exemplos nas seções 2.2.4 e 2.2.5.

Exemplo

Olhar exemplos nas seções 2.2.4 e 2.2.5.

2.2.4. Agregação

Descrição

Uma associação pode representar uma agregação (isto é, um relacionamento de todo/parte) (OLIVEIRA *et al.*, 2005). A Figura 18 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de elementos que podem participar desse tipo de relacionamento.

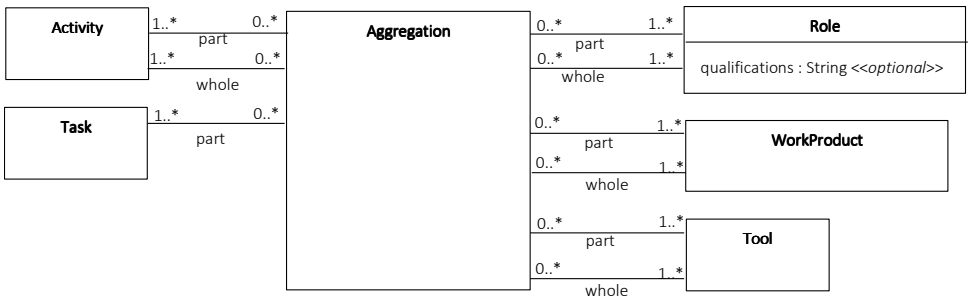


Figura 18 - Metamodelo *OdysseyProcess-FEX*: Relacionamento *Aggregation* (Agregação)

Hierarquia

Superclasse: *Association* (*Associação)*

Atributos

Sem atributos.

Associações

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

Restrições

Sem restrições específicas.

Notação

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.

- Ícone: ◇——→

Exemplo

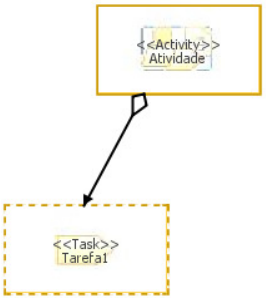


Figura 19- Exemplo de Representação Gráfica do Relacionamento Agregação

2.2.5. Composição

Descrição

Uma associação pode representar uma composição (isto é, um relacionamento de todo/parte). A composição é um relacionamento mais forte do que agregação, e requer que em um dado momento, uma instância esteja incluída em no máximo uma composição (OLIVEIRA *et al*., 2005). Neste relacionamento, as partes não existem independentes do todo. A Figura 20 apresenta o relacionamento e as categorias de elementos de processo envolvidas.

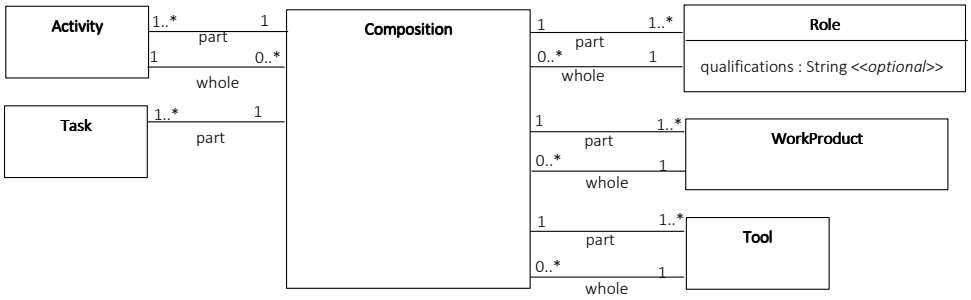


Figura 20- Metamodelo *OdysseyProcess-FEX*: Relacionamento *Composition* (Composição)

Hierarquia

Superclasse: *Association* (Associação)

Atributos

Sem atributos.

Associações

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

Restrições

[1] Não deve haver relacionamento *Composition* (Composição) entre elementos de processo quando o elemento que representa o "todo" é opcional e o elemento que representa a "parte" é mandatório.

[2] Não deve haver relacionamento *Composition* (Composição) entre elementos de processo quando o elemento que representa o "todo" é mandatório e o elemento que representa a "parte" é opcional. Neste caso, deve ser utilizado o relacionamento *Aggregation* (Agregação).

Notação

Representa um relacionamento de todo/parte em que 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.

- Ícone: 

Exemplo

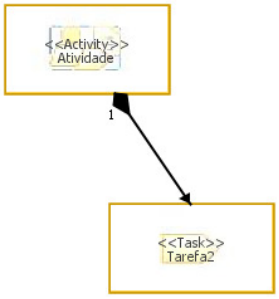


Figura 21- Exemplo de Representação Gráfica do Relacionamento Composição

2.2.6. *WorkUnitRoleRelationship* (LigaçãoUnidadeDeTrabalhoPapel)

Descrição

Estabelece um relacionamento de associação entre um elemento de processo da categoria Unidade de Trabalho (Atividade ou Tarefa) e um ou vários elementos de processo da categoria Papel participantes na execução da unidade de trabalho. A Figura 22 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de elementos que podem participar desse tipo de relacionamento.

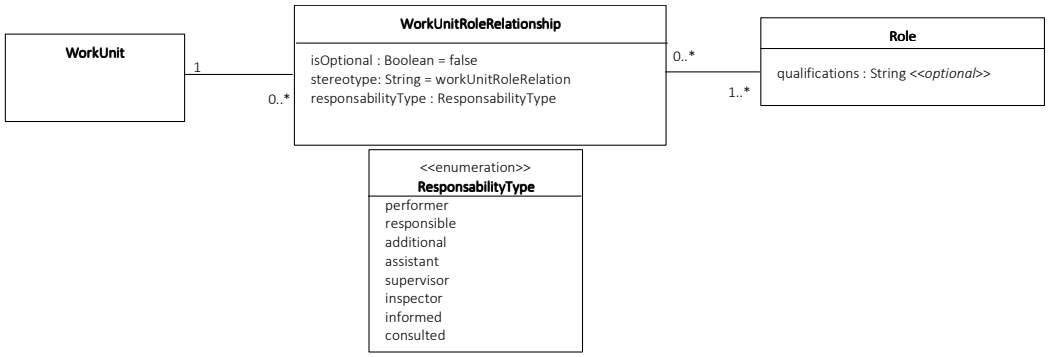


Figura 22- Metamodelo *OdysseyProcess-FEX*: *WorkUnitRoleRelationship* (LigaçãoUnidadeDeTrabalhoPapel)

Hierarquia

Superclasse: *Relationship* (Relacionamento*)*

Atributos

- ***isOptional* (ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação do elemento de processo Papel (*Role*) na execução da unidade de trabalho (*WorkUnit*) representada pelo elemento de processo atividade (*Activity*) ou tarefa (*Task*) associada. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** String = *workUnitRoleRelation*– atributo que especifica o tipo da associação criada.
- ***responsabilityType*** (tipoResponsabilidade): ResponsabilityType. A classe ResponsabilityTyperepresenta uma enumeração dos tipos de

responsabilidades que podem ser assumidos por um papel participante da execução de uma unidade de trabalho. De forma inicial foi definido um conjunto de responsabilidades possíveis representadas pelos estereótipos: *<<performer>>, <<responsible>>, <<additional>>, <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted*>>(Tabela 3).

Tabela 3- Tipos de Responsabilidades dos papéis envolvidos na execução de unidades de trabalho

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

Associações

Sem associações específicas.

Restrições

[1] É importante que toda unidade de trabalho apresente um papel como responsável pela sua execução.

[2] Em uma representação de hierarquia de unidades de trabalho (atividades e tarefas), os papéis associados ao todo ou ao ponto de variação participam da execução das subunidades de trabalho associadas. Caso a participação de cada papel não seja modelada separadamente nas subunidades, pode-se considerar que tais papéis participam da execução de todas as subunidades com o mesmo tipo de responsabilidade que foi definido para a unidade que representa o todo ou o ponto de variação.

Notação

Possui a representação como uma linha simples entre os elementos.

- Estereótipos: Tal relacionamento pode apresentar diferentes estereótipos que indicam o tipo de participação que determinado papel possui na execução da unidade de trabalho associada. Um conjunto inicial de possíveis responsabilidades foi definido: *<<performer>>; <<responsible>>;*

<<additional>>; <<assistant>>; <<supervisor>>;<<inspector>>; <<informed>>;<<consulted>>.

- Tal relação possui o estereótipo <<*optional*>> para indicar que a relação é opcional no domínio (atributo *ehOpcional* = true).

Exemplo

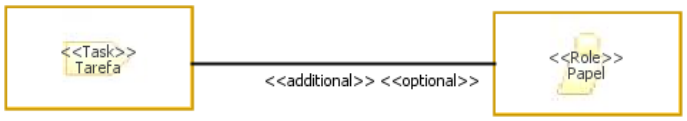


Figura 23- Exemplo de Representação Gráfica do Relacionamento

2.2.7. *WorkUnitWorkProdutRelationship* (LigaçãoUnidadeDeTrabalhoProdutoDeTrabalho)

Descrição

Estabelece um relacionamento de associação entre um elemento de processo da categoria Unidade de Trabalho (Atividade ou Tarefa) e um ou vários elementos da categoria ProdutoDeTrabalho que representam artefatos a serem consumidos, produzidos ou modificados pela execução da unidade de trabalho. A Figura 24 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de elementos que podem participar desse tipo de relacionamento.

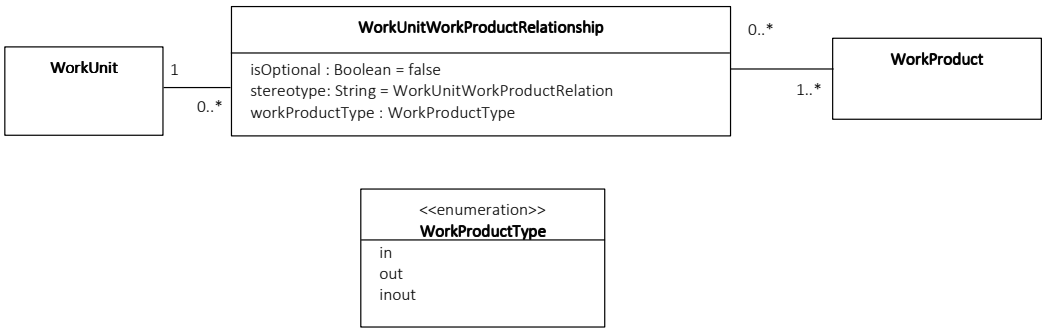


Figura 24 - Metamodelo *OdysseyProcess-FEX*: WorkUnitWorkProductRelationship(LigaçãoUnidadeDeTrabalhoProdutoDeTrabalho)

Hierarquia

Superclasse: *Relacionamentos*

Atributos

- ***isOptional* (ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação do elemento de processo ProdutodeTrabalho (*WorkProduc*t) na execução da unidade de trabalho (*WorkUnit*) representada pelo elemento de processo atividade (*Activity*) ou tarefa (*Task*) associada. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** String = *workUnitWorkProductRelation*– atributo que especifica o tipo da associação criada.
- ***workProductyType*** (tipoProdutoDeTrabalho): *WorkProductType*. A classe *WorkProductType* representa uma enumeração dos tipos que produtos de trabalho podem assumir, representados pelos estereótipos: *<<in>>, <<out>>, <<inout*>>(Tabela 4).

Tabela 4- Tipos dos produtos de trabalho envolvidos na execução de unidades 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*

Associações

Sem associações específicas.

Restrições

[1] Em uma representação de hierarquia de unidades de trabalho (atividades e tarefas), os produtos associados ao todo ou ao ponto de variação são produtos de entrada ou saída execução das subunidades de trabalho associadas. É importante que a participação de cada produto de trabalho seja modelada separadamente nas subunidades. É interessante que só os produtos trocados com o meio sejam representados associados com a unidade de trabalho que representa o todo ou ao ponto de variação. Produtos de trabalho que são trocados internamente devem ser representados associados apenas nas subunidades.

Notação

Possui a representação como uma linha simples entre os elementos.

- Estereótipos: Tal relacionamento pode apresentar diferentes estereótipos que indicam o tipo de participação do produto de trabalho na execução da unidade de trabalho associada:
  - Produto de Trabalho a ser consumido: **<<*in*>>**
  - Produto de Trabalho a ser produzido: **<<*out*>>**
  - Produto de Trabalho a ser modificado: **<<*inout*>>**
- Tal relação possui o estereótipo <<*optional*>> para indicar que a relação é opcional no domínio (atributo *ehOpcional* = true).

Exemplo

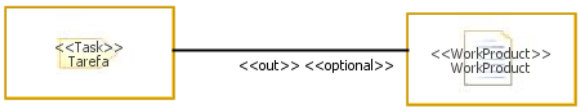


Figura 25 - Exemplo de Representação Gráfica do Relacionamento

2.2.8. *WorkProductRoleRelationship* (LigaçãoPapelProdutoDeTrabalho)

Descrição

Estabelece um relacionamento de associação entre um elemento de processo da categoria Produto de Trabalho (*WorkProduct*) e um ou vários elementos de processo da categoria Papel (*Role*) com algum tipo de responsabilidade sobre o produto de trabalho. A Figura 26 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de elementos que podem participar desse tipo de relacionamento.

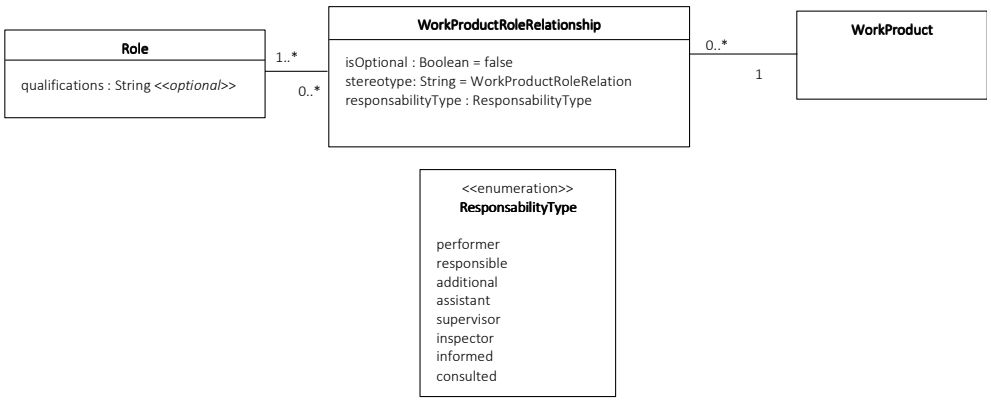


Figura 26- Metamodelo *OdysseyProcess-FEX: WorkProductRoleRelationship* (LigaçãoPapelProdutoDeTrabalho)

Hierarquia

Superclasse: *Relacionamentos*

Atributos

- ***isOptional* (ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de participação do elemento de processo Papel (*Role*) no consumo, produção ou modificação de um produto de trabalho (*WorkProduct*) por uma unidade de trabalho (*WorkUnit*). Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** String = *workProductRoleRelation*– atributo que especifica o tipo de associação criada.
- ***responsabilityType*** (tipoResponsabilidade): ResponsabilityType. A classe ResponsabilityType representa uma enumeração dos tipos de responsabilidades que podem ser assumidos por um papel que possui algum tipo de responsabilidade sobre o produto de trabalho associado. De forma inicial foi definido um conjunto de responsabilidades possíveis representadas pelos estereótipos: *<<performer>>, <<responsible>>, <<additional>>,* <<assistant>>, <<supervisor>>, <<inspector>>, <<informed>>, <<consulted>>(Tabela 3).

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Possui a representação como uma linha simples entre os elementos.

- Estereótipos: Tal relacionamento pode apresentar diferentes estereótipos que indicam o tipo de participação que determinado papel possui para um

produto de trabalho associado. Um conjunto inicial de possíveis responsabilidades foi definido: *<<performer>>; <<responsible>>; <<additional>>; <<assistant>>; <<supervisor>>;<<inspector>>; <<informed>>;<<consulted>>.*

- Tal relação possui o estereótipo <<*optional*>> para indicar que a relação é opcional no domínio (atributo *ehOpcional* = true).

Exemplo

Figura 27- Exemplo de Relacionamento na LPrS de Planejamento de Projetos de Software

2.2.9. *WorkUnitToolRelationship* (LigaçãoUnidadeDeTrabalhoFerramenta)

Descrição

Estabelece um relacionamento de associação entre um elemento de processo da categoria Unidade de Trabalho (Atividade ou Tarefa) e um ou vários elementos de processo da categoria Ferramenta que podem prover algum tipo de apoio ferramental na execução da unidade de trabalho. A Figura 28 apresenta o relacionamento e os possíveis papéis assumidos pelo conjunto de categorias de elementos que podem participar desse tipo de relacionamento.

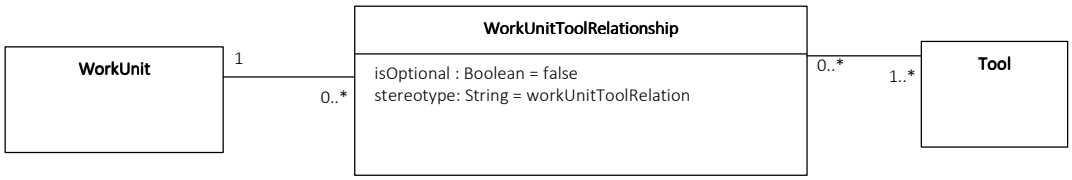


Figura 28- Metamodelo *OdysseyProcess-FEX*: *WorkUnitToolRelationship* (LigaçãoUnidadeDeTrabalhoTool)

Hierarquia

Superclasse: *Relationship* (Relacionamento*)*

Atributos

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

de participação do elemento de processo Ferramenta (*Tool*) na execução da unidade de trabalho (*WorkUnit*) representada pelo elemento de processo atividade (*Activity*) ou tarefa (*Task*) associada. Caso *true*, o relacionamento é Opcional. Default: *false*.

- ***stereotype* (estereótipo):** String = *workUnitToolRelation*– atributo que especifica o tipo da associação criada.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Possui a representação como uma linha simples entre os elementos.

- Tal relação possui o estereótipo <<*optional*>> para indicar que a relação é opcional no domínio (atributo *ehOpcional* = true).

Exemplo

Figura 29- Exemplo de Representação Gráfica do Relacionamento

2.2.10. *ControlFlow* (Fluxo de Controle)

Descrição

O fluxo de controle define relações de conexão entre unidades de trabalho e seu fluxo de execução. A definição dos diferentes fluxos de controle permite descrever os possíveis comportamentos da família de processos descrita por uma Linha de Processos de Software. Pode ser implementado por cinco tipos específicos de relação (Figura 30): (1) Fluxo de Sequência (*SequenceFlow*); (2)

Fluxo Paralelo (*ParallelFlow*); (3) Fluxo com sincronismo (*SynchronousFlow*); (4) Fluxo de Decisão (*DecisionFlow*); e (5) Fluxo de Junção(*Merge Flow*).

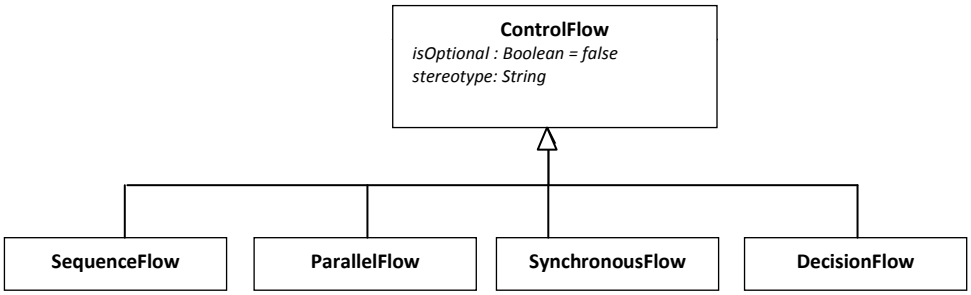


Figura 30 - Metamodelo *OdysseyProcess-FEX*: *ControlFlowRelations* (Relações de Fluxo de Controle)

Hierarquia

Superclasse: *Relacionamentos*

Subclasses: (1) Fluxo de Sequência (*SequenceFlow*); (2) Fluxo Paralelo (*ParallelFlow*); (3) Fluxo com sincronismo (*SynchronousFlow*); e (4) Fluxo de Decisão (*DecisionFlow*).

Atributos

- ***isOptional* (ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Indica a opcionalidade de realização do fluxo de controle nos processos específicos de projeto a serem derivados. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** String

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Sem notações específicas. As representações gráficas estão associadas aos tipos especificados pelas subclasses.

Exemplo

Olhar exemplos nas subclasses associadas.

2.2.11. *SequenceFlow* (Fluxo de Sequência)

Descrição

O fluxo de sequência (Figura 31) descreve uma relação de fluxo ordenado de execução entre duas unidades de trabalho (origem e destino), indicando que a execução da atividade destino só pode ser iniciada com o fim da execução da atividade anterior.

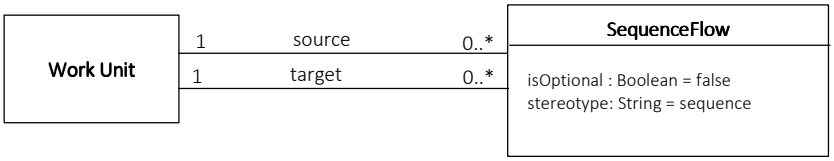


Figura 31 - *Metamodelo OdysseyProcess-FEX:* SequenceFlow (Fluxo de Sequência)

Hierarquia

Superclasse: *ControlFlow*(Fluxo de Controle)

Atributos

- ***isOptional* (ehOpcional)**: *Boolean* – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** *String*= *sequence* (sequência)

Associações

- ***source* (origem)**: Referencia o elemento de processo que representa a origem do fluxo de execução na sequência, ou seja, represente a unidade de trabalho de início de execução do fluxo.
- ***target* (destino)**: Referencia o elemento de processo que representa o destino do fluxo de execução na sequência, ou seja, representa a unidade de trabalho de fim de execução do fluxo.

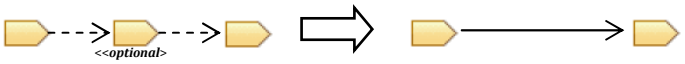
Restrições

[1] Apenas elementos de processo da categoria Unidade de Trabalho (atividade e/ou tarefa) podem participar deste relacionamento.

[2] O relacionamento só pode ser estabelecido entre duas unidades de trabalho, sendo uma a origem e a outra o final do fluxo.

[3] Relações de Sequência entre duas unidades de trabalho onde uma delas ou as duas são classificadas como opcionais: *ProcessFeature.optionalityType = optional* (CaracterísticaDeProcesso.*tipoOpcionalidade = opcional*) são também classificadas como opcionais: *SequenceFlow.isOptional = true* (FluxoDeSequência.ehOpcional = verdadeiro). Caso contrário, com duas unidades de trabalho classificadas como mandatórias, o fluxo também será classificado como mandatório *SequenceFlow.isOptional = false* (FluxoDeSequência.ehOpcional = falso).

[4] Em tempo de derivação de processo a partir do modelo de domínio, na ausência do elemento opcional do fluxo criar um fluxo de sequência entre os elementos anterior e posterior:

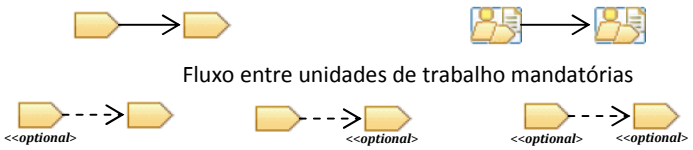


Notação

Representa uma ordem sequencial de execução entre duas unidades de trabalho através de uma linha com uma seta indicando a direção da sequência da origem para o destino.

- Sequência mandatória: ⟶
- Sequência opcional: - - - - - ->

Exemplo



Fluxo entre unidades de trabalho mandatórias

Fluxo entre unidades de trabalho opcionais

Figura 32- Exemplo de Representações Gráficas do Fluxo Sequencial

2.2.12. *DecisionFlow* (Fluxo de Decisão)

Descrição

O fluxo de Decisão (Figura 33) descreve um ponto em um fluxo de execução onde uma escolha entre múltiplos caminhos deve ser realizada. A relação é representada por apenas um fluxo de entrada e múltiplos caminhos de execução que representam fluxos alternativos. Tal tipo de relação introduz condições em unidades de trabalhos e seus fluxos de execução.

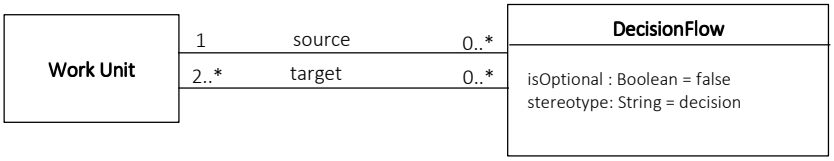


Figura 33 - *Metamodelo OdysseyProcess-FEX: DecisionFlow* (Fluxo de Decisão)

Hierarquia

Superclasse: *ControlFlow*(Fluxo de Controle)

Atributos

- ***isOptional* (ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** String = decision (decisão)

Associações

- ***source* (origem)**: Referencia o elemento de processo que representa a origem do fluxo de execução na sequência, ou seja, represente a unidade de trabalho de início de execução do fluxo.
- ***target* (destino)**: Referencia os elementos de processo que representamos destinos dos múltiplos fluxos alternativos de execução, ou seja, representam as possibilidades de unidades de trabalho de final de execução dos fluxos.

Restrições

[1] Apenas elementos de processo da categoria Unidade de Trabalho (atividade e/ou tarefa) podem participar deste relacionamento.

[2] O relacionamento só pode ser estabelecido entre três ou mais unidades de trabalho, sendo uma única origem e múltiplos alternativas de execução.

[3] Caso apenas um elemento do destino é classificado como mandatório, o relacionamento deve ser classificado como opcional *DecisionFlow.isOptional = true* (FluxoDeDecisão.ehOpcional = verdadeiro), com a aresta relacionada ao fluxo opcional pontilhada e o nó de decisão pontilhado. Nestes casos, durante a derivação de um processo específico que não inclui nenhum dos elementos opcionais, o fluxo passa a ser um fluxo sequencial.

[4] Relações de decisão com dois ou mais elementos de destino classificados como mandatórios, o relacionamento deve ser classificado como mandatório DecisionFlow.isOptional = false (FluxoDeDecisão.ehOpcional = falso) e mantido mesmo com a ausência do elemento de origem nos processos a serem derivados a partir da LPrS definida. Neste caso, durante a derivação de um processo específico, a não inclusão de um elemento opcional impacta apenas na omissão da aresta opcional.

Notação

Representa um ponto envolvendo uma escolha entre múltiplos fluxos de execução com uma unidade de trabalho como início do fluxo e múltiplas unidades de trabalho como alternativas excludentes entre si interligadas por linhas simples e com um nó de decisão entre a origem e os destinos.

Nó de decisão: Mandatório ◆ Opcional ◇

Exemplo

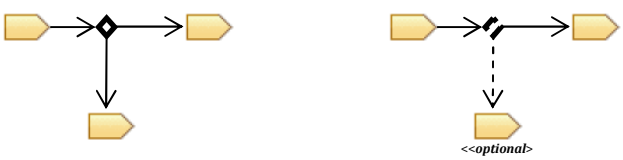


Figura 34- Representações Gráficas do Fluxo de Decisão

2.2.13. *ParallelFlow* (Fluxo Paralelo)

Descrição

O fluxo paralelo (Figura 35) descreve uma relação entre três ou mais unidades de trabalho, indicando que o fluxo de execução é dividido em múltiplos fluxos concorrentes (execução paralela).

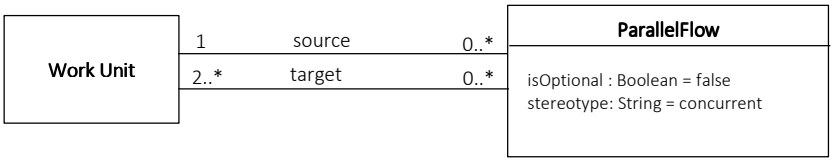


Figura 35 - *Metamodelo OdysseyProcess-FEX: ParallelFlow* (Fluxo Paralelo)

Hierarquia

Superclasse: *ControlFlow*(Fluxo de Controle)

Atributos

- ***isOptional* (ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** String = concurrent (concorrente)

Associações

- ***source* (origem)**: Referencia o elemento de processo que representa a origem do fluxo de execução na sequência, ou seja, represente a unidade de trabalho de início de execução do fluxo.
- ***target* (destino)**: Referencia os elementos de processo que representamos destinos dos múltiplos fluxos concorrentes de execução, ou seja, representam as unidades de trabalho de final de execução dos fluxos.

Restrições

[1] Apenas elementos de processo da categoria Unidade de Trabalho (atividade e/ou tarefa) podem participar deste relacionamento.

[2] O relacionamento só pode ser estabelecido entre três ou mais unidades de trabalho, sendo uma única origem e múltiplos destinos de execução.

[3] Relações de paralelismo com dois ou mais elementos de destino classificados como mandatórios, o relacionamento deve ser classificado como mandatório *ParalllelFlow.isOptional = false* (FluxoParalelo.ehOpcional = falso) e mantido mesmo com a ausência do elemento de origem nos processos a serem derivados a partir da LPrS definida. Apenas as arestas relacionadas aos fluxos opcionais devem ser pontilhadas. Neste caso, durante a derivação de um processo específico a não inclusão de um elemento opcional impacta apenas na omissão da aresta opcional e na manutenção do paralelismo na execução.

[4] Se apenas um elemento do destino é classificado como mandatório, o relacionamento deve ser classificado como opcional *ParalllelFlow.isOptional = true* (FluxoParalelo.ehOpcional = verdadeiro) e poderá ser omitido ou substituído por outro tipo de fluxo de controle nos processos a serem derivados a partir da LPrS definida. Desta forma, a classificação do relacionamento independente da classificação de opcionalidade do elemento de origem.

Notação

Representa relação entre três ou mais unidades de trabalho, indicando que o fluxo de execução é dividido em múltiplos fluxos concorrentes (execução paralela) com uma unidade de trabalho como início do fluxo e múltiplas unidades de trabalho como sequências de execução interligadas por linhas simples e com um nó de divisão entre a origem e os destinos.

- Nó de Divisão (*fork node*): Mandatório | Opcional ⋮

Exemplo

Figura 36- Representações Gráficas do Fluxo Paralelo

2.2.14. *SynchronousFlow* (Sincronismo)

Descrição

O fluxo com sincronismo (Figura 37) descreve uma relação entre três ou mais unidades de trabalho, indicando que múltiplos fluxos de execução concorrentes devem ser sincronizados ao final e unidos em um único destino.

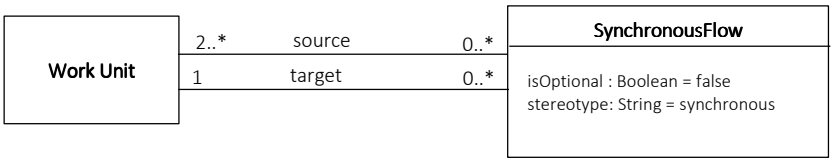


Figura 37 - *Metamodelo OdysseyProcess-FEX: SynchronousFlow* (Fluxo com sincronismo)

Hierarquia

Superclasse: *ControlFlow*(Fluxo de Controle)

Atributos

- ***isOptional* (ehOpcional)**: Boolean – atributo que define a classificação do relacionamento quanto a sua opcionalidade. Caso *true*, o relacionamento é Opcional. Default: *false*.
- ***stereotype* (estereótipo):** String = synchronous (sincronismo).

Associações

- ***source* (origem)**: Referencia os elementos de processo que representam as origens dos múltiplos fluxos de execução, ou seja, represente as unidades de trabalho de início de execução dos fluxos.
- ***target* (destino)**: Referencia o elemento de processo que representa o único destino dos múltiplos fluxos concorrentes de execução, ou seja, representa a unidade de trabalho de final de execução dos fluxos.

Restrições

[1] Apenas elementos de processo da categoria Unidade de Trabalho (atividade e/ou tarefa) podem participar deste relacionamento.

[2] O relacionamento só pode ser estabelecido entre três ou mais unidades de trabalho, sendo múltiplas origens e um único destino de execução, com a restrição de sincronismo ao final da execução.

[3] Relações com sincronismo em que dois ou mais elementos de origem são classificados como mandatórios, o relacionamento deve ser classificado como mandatório *SynchronousFlow.isOptional = false* (FluxoComSincronismo.ehOpcional = falso) e mantido mesmo com a ausência do elemento de destino nos processos a serem derivados a partir da LPrS definida. Nestes casos, durante a derivação de um processo específico que não inclui nenhum dos elementos opcionais passa a ter um fluxo sequencial.

[4] Se apenas um elemento da origem é classificado como mandatório, o relacionamento deve ser classificado como opcional *SynchronousFlow.isOptional = true* (FluxoComSincronismo.ehOpcional = verdadeiro) e poderá ser omitido ou substituído por outro tipo de fluxo de controle nos processos a serem derivados a partir da LPrS definida. Neste caso, durante a derivação de um processo específico a não inclusão de um elemento opcional impacta apenas na omissão da aresta opcional e na manutenção do sincronismo na execução.

Notação

Representa relação entre três ou mais unidades de trabalho, indicando que múltiplos fluxos de execução concorrentes são unidos em um fluxo (execução síncrono) com múltiplas unidades de trabalho como inícios dos fluxos e uma unidade de trabalho como sequência final de execução interligadas por linhas simples e com um nó de junção entre as origens e o destino.

- Nó de Junção (*Join node*): Mandatório | Opcional ⋮

Exemplo

Figura 38- Representações Gráficas do Fluxo Síncrono

2.3. Nível Análise (*Process Features View*) - Pacote Regras de Composição

O pacote de Regras de Composição (*CompositionRules*) 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 elementos de processo e pode ser visualizado na Figura 39.

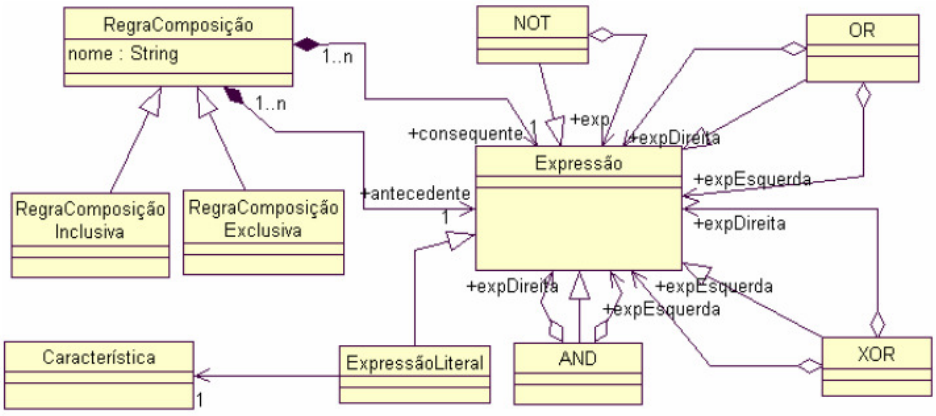


Figura 39 - Metamodelo *OdysseyProcess-FEX*: Pacote Regras de Composição (OLIVEIRA, 2006)

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

2.3.1. RegraComposição

Descrição

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

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] Elementos de processos dependentes entre si não podem ser mutuamente exclusivos, e vice-versa.

Notação

As Regras de Composição são graficamente representadas por uma marcação, em todas as características pertencentes à regra, no canto inferior do retângulo que representa a característica envolvida na regra estabelecida. A marcação também apresenta uma numeração (n), que representa a ordem sequencial de criação das regras. Marcações específicas são usadas em cada tipo de regra de composição descritos abaixo.

Exemplo

Olhar exemplo nas seções de 99cada tipo de regra de composição abaixo.

2.3.2. RegraComposiçãoInclusiva

Descrição

Regras de composição que indicam dependência entre dois ou mais elementos de processo. 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 um elemento de processo A requer o elemento de processo B, e o elemento de processo B requer o elemento de processo A, existirão duas Regras de Composição.

Notação

As Regras de Composição Inclusivas são graficamente representadas por uma marcação "R_n", no canto inferior direito, em todas as características pertencentes à regra. A marcação também apresenta uma numeração (n), que representa a ordem sequencial de criação das regras.

Exemplo

Regras de Composição Inclusiva do domínio de Planejamento de Projetos de Software

- R: Determine effort using size estimated by FP *requires* (Determine by Indicative Function Points (FP) AND FP Spreadsheet filled with Project Size)
- R_1: Determine effort using size estimated by historical base *requires* (Determine by projects historical base AND Project Size estimated by Historical Analysis)

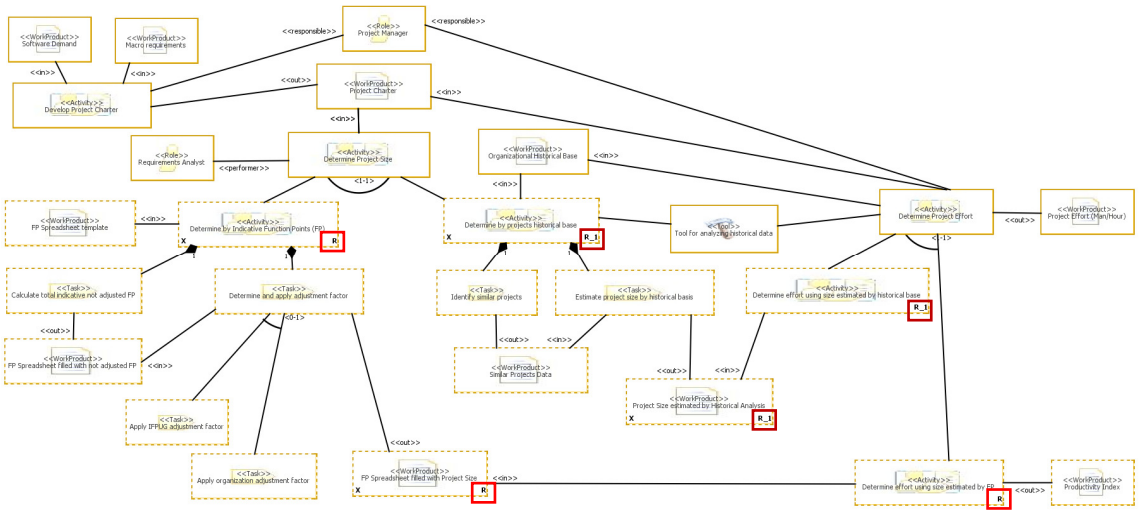


Figura 40 - Representação de Regra de Composição Inclusiva no Modelo de Características de Processo de Software

2.3.3. RegraComposiçãoExclusiva

Descrição

Regras de composição que indicam mútua exclusividade entre dois ou mais elementos de processo. 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 elementos de processo mandatórios, somente elementos de processo opcionais.

Notação

As Regras de Composição Exclusivas são graficamente representadas por uma marcação "X_n", no canto inferior esquerdo, em todas as características pertencentes à regra. A marcação também apresenta uma numeração (n), que representa a ordem sequencial de criação das regras.

Exemplo

Regras de Composição Exclusiva do domínio de Planejamento de Projetos de Software

- X: (Determine by Indicative Function Points (FP) AND FP Spreadsheet filled with Project Size) *excludes* (Determine by projects historical base AND Project Size estimated by Historical Analysis)

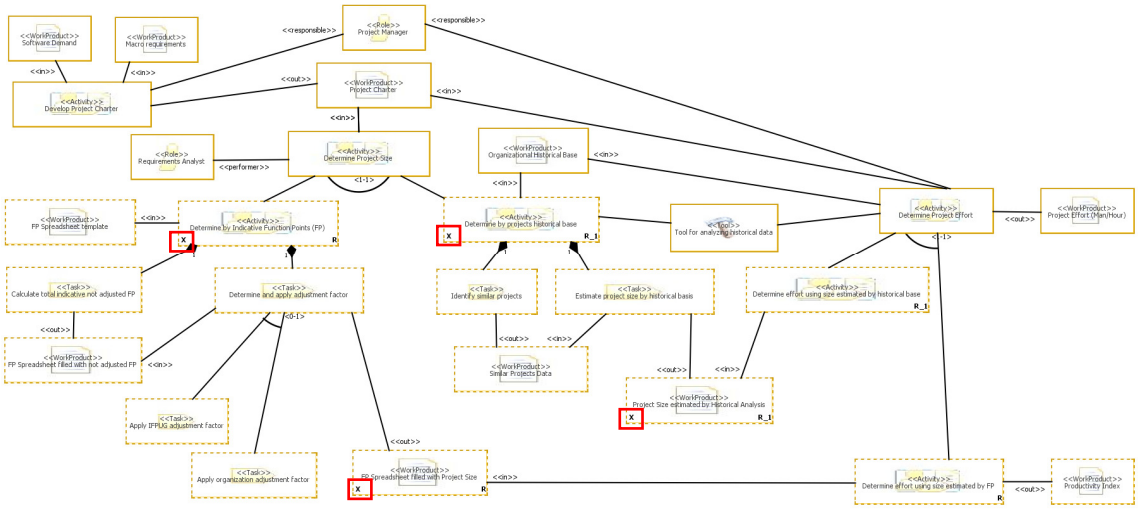


Figura 41 - Representação de Regra de Composição Exclusiva no Modelo de Características de Processo de Software

2.3.4. 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.

Notação

Sem notações específicas.

Exemplo

-

2.3.5. ExpressaoLiteral

Descrição

Expressão Literal é a expressão mais elementar de uma Regra de Composição. É representada por um único elemento de processo.

Hierarquia

Superclasse: *Expressão*

Atributos

- **CaracterísticaDeProcesso**: ProcessFeature [1] – Elemento de Processo que constitui a expressão literal.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Sem notações específicas.

Exemplo

-

2.3.6. 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.

Notação

Sem notações específicas.

Exemplo

Exemplo do Antecedente da Regra de Composição Exclusiva do domínio de Planejamento de Projetos de Software: *Determine by Indicative Function Points (FP)* ***AND*** *FP Spreadsheet filled with Project Size*

2.3.7. 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.

Notação

Sem notações específicas.

Exemplo

Determine by Indicative Function Points (FP) ***OR*** *FP Spreadsheet filled with Project Size*

2.3.8. 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.

Notação

Sem notações específicas.

Exemplo

Determine by Indicative Function Points (FP) ***XOR*** *FP Spreadsheet filled with Project Size*

2.3.9. 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.

Notação

Sem notações específicas.

Exemplo

NOT *FP Spreadsheet filled with Project Size*

3. LPrS – Nível Projeto – Modelagem de Componente s de Processos de Software (*Process Components View*)

O metamodelo do nível de projeto (*Process Components View*) da *OdysseyProcess-FEX* é constituído por componentes, seus atributos específicos e relações via o estabelecimento de interfaces (Figura 42). Alguns desses elementos possuem relações com o metamodelo do nível de características *(Process Features View*) que estão descritas ao longo desta seção.

A seguir uma descrição detalhada de cada elemento é apresentada, seguindo a estrutura dividida nos cinco itens de descrição: Descrição, Hierarquia, Atributos, Associações e Restrições.

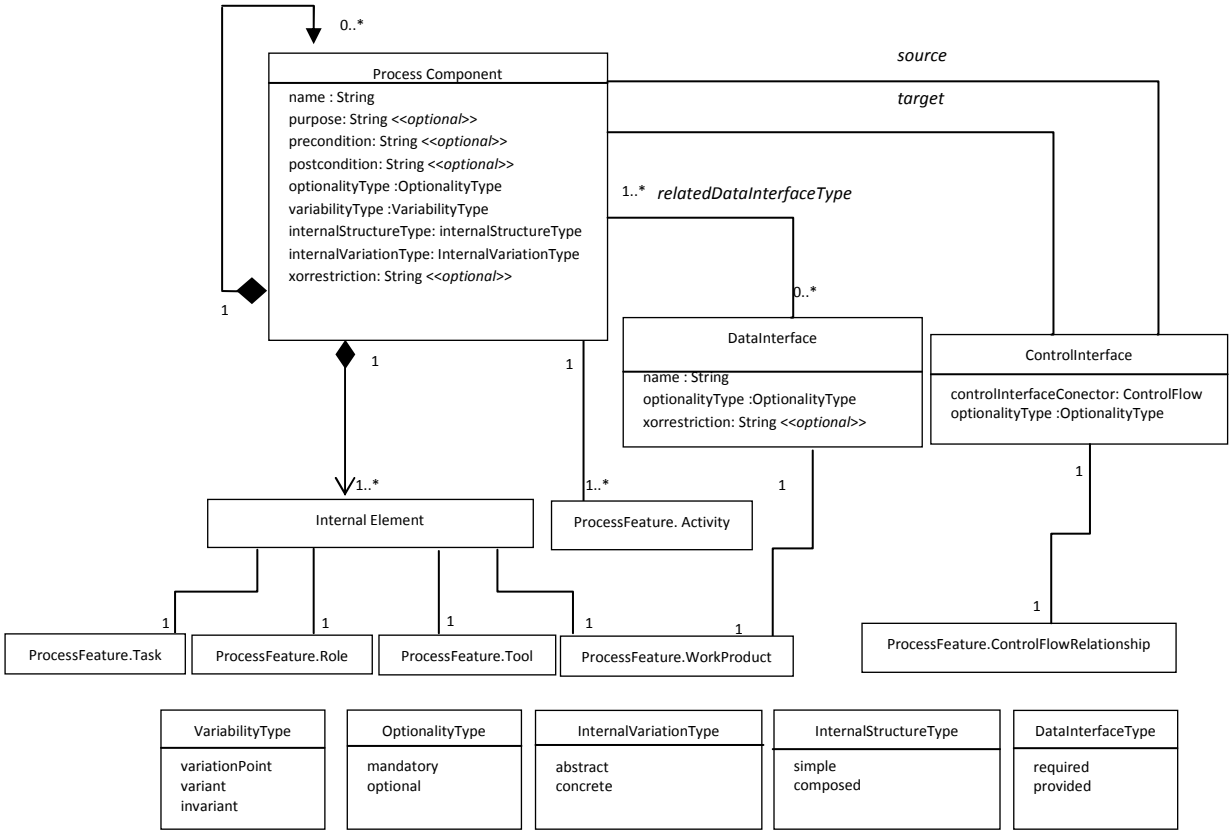


Figura 42 - Metamodelo do Nível de Componentes do *OdysseyProcess-FEX*: Pacote Principal

3.1. ComponenteProcesso

Descrição

Um componente de processo pode ser entendido como uma abstração de um fragmento de processo baseada no princípio da "caixa-preta". Desta forma, um componente representa uma parte modular de um processo que encapsula seu conteúdo e se comunica com o ambiente via interfaces. Cada componente é composto por unidades de trabalho (atividades e tarefas) que representam exatamente suas possibilidades de realização, ou seja, unidades de trabalho que quando executadas apresentam o comportamento do componente e produzem os resultados esperados (produtos de trabalho) que agregam valor ao projeto.

Um componente de processo deve ser definido minimamente por uma atividade. Uma unidade de reutilização menor poderia introduzir uma complexidade desnecessária e tornar atividades de reutilização mais difíceis.

Hierarquia

Sem hierarquia específica.

Atributos

- ***name* (nome)**: String[1] – atributo que define o nome do componente.
- ***purpose* (propósito)**: String [*..1] – atributo que estabelece a finalidade ou o objetivo a ser alcançado pelo componente a ser executado para alcançar determinado valor de resultado dentro do processo.
- ***precondition* (pré-condição)**: String [0..*] – atributo que estabelece um conjunto de restrições como critérios de entrada para o componente, ou seja, condições que permitem o início da execução do componente.
- ***postcondition*** (**pós-condição)**: String [0..*] – atributo que estabelece um conjunto de restrições como critérios de saída para o componente, ou seja, condições que permitem concluir a execução do componente, estabelecendo os resultados que precisam ser atingidos ao final desta execução.
- ***optionalityType* (tipoOpcionalidade): *OptionalityType*[1] (TipoOpcionalidade[1])** – atributo que define a classificação do componente quanto a sua opcionalidade. Pode assumir os valores *mandatory* (mandatório) e *optional*(opcional). Indica se o componente é mandatório no domínio, isto é, se o componente estará presente em todos os processos instanciados a partir do modelo ou se o componente é opcional. O tipo de opcionalidade é representado pela lista enumerada *OptionalityType*(TipoOpcionalidade). Default: *optional*(opcional).
- ***variabilityType* (tipoVariabilidade): *VariabilityType*[1] (TipoVariabilidade[1]) –** atributo que indica o tipo de variabilidade que o componente apresenta. Pode assumir os valores *invariant* (invariante), *variant* (variante) e *variationpoint* (ponto de variação). O tipo de variabilidade é representado pela lista enumerada

VariabilityType (TipoVariabilidade). Default: *invariant* (invariante).

- ***internalStructureType* (tipoEstruturaInterna): *InternalStructureType* [1] (TipoEstruturaInterna [1])** – atributo que indica a granularidade do componente podendo ser minimamente uma atividade ou podendo ser composto por outro(s) componente(s), podendo assumir os valores *simple* (simples) e *composed* (composto), respectivamente. O tipo de estrutura interna é representado pela lista enumerada *InternalStructureType* (TipoEstruturaInterna). Default: *simple* (simples).
    - Granularidade mínima: Atividade
    - Granularidade intermediária: mais de uma Atividade
    - Granularidade máxima: um ou mais Componentes
- ***internalVariationType* (tipoVariaçãoInterna): *InternalVariationType* [1] (TipoVariaçãoInterna [1])** – atributo que indica se existe algum tipo de variação interna no componente, ou seja, se existe alguma necessidade de configuração dos elementos que o compõem (unidades de trabalho, produtos de trabalho, papéis ou ferramentas. Pode assumir os valores *concrete* (concreto) e *abstract* (abstrato). Um componente que não possui variação interna a ser resolvida (todos os elementos de processo que compõem o componente estão definidos da maneira que serão instanciados, sem nenhuma decisão adicional de definição) é classificado como concreto. Caso contrário será classificado como abstrato. O tipo de variação interna é representado pela lista enumerada *InternalVariationType* (TipoVariaçãoInterna). Default: *concrete* (concreto).
- ***xorrestriction* (restrição de mútua exclusividade**): String [0..1] – representa uma restrição indicando a relação de mútua exclusividade entre componentes.

Associações

- ***internalElement*** (elemento interno): *ProcessFeature.Task* [*] | *ProcessFeature.Role* [1..*] | *ProcessFeature.Tool* [0..*] | *ProcessFeature.WorkProduct* [0..*] - especifica os elementos de processos que foram agrupados e pertencem a um componente.

  1. Possui as tarefas associadas à atividade que compõe o componente (*relatedActivity*).

  2. No mínimo, possui um papel associado.

     a. A cada papel associado podem ser relacionados um ou mais agentes que podem desempenhar aquele papel durante a execução do componente. Caso haja informação da atuação de determinado agente em execuções deste componente já realizadas, atribuir essa informação caracterizando em que contexto esse agente atuou. A caracterização dos projetos atuados deve ser associada nestes casos.

  3. Pode possuir uma ou mais ferramentas associadas.

     a. A cada ferramenta associada pode ser relacionado um script de execução, caso determinada unidade de trabalho representada no componente possa ser automatizada.

  4. Pode possui um ou mais produtos de trabalho que são trocados apenas por unidades de trabalho internas ao componente.

     a. A cada produto de trabalho associado pode ser relacionado um *template* indicando o formato de resultado esperado.

- ***relatedActivity*** (atividade relacionada): ProcessFeature.Activity [1] – especifica a atividade do nível de análise que o componente representa.

- ***internalComponents*** (componentes internos) : *ProcessComponent* [0..*]

– no caso de componente composto (*ProcessComponent.internalStructureType = composed*), especifica os componentes que o compõem.

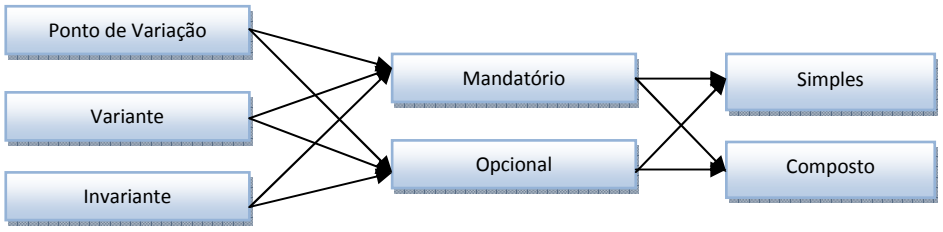
Restrições

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

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

[3] As classificações de componentes com relação à estrutura interna são mutuamente excludentes entre si. Desta forma, um componente não pode receber simultaneamente dois tipos diferentes de classificação (simples ou composto).

[4] As classificações quanto à opcionalidade, variabilidade e estrutura interna são ortogonais entre si. Desta forma, um componente pode receber uma classificação quanto à opcionalidade, outra classificação complementar quanto à variabilidade e outra quanto à estrutura interna.



Classificação ortogonal dos componentes

[5] As classificações de componentes com relação à variação interna são mutuamente excludentes entre si. Desta forma, um componente não pode receber simultaneamente dois tipos diferentes de classificação (concreto ou abstrato).

[6] Cada componente envolvido em uma regra de composição inclusiva como antecedente deve possuir uma pré-condição especificada indicando a dependência com os componentes que compõem o consequente da regra. Caso a dependência for de produtos de trabalho, interfaces de dados devem ser estabelecidas representando essa dependência e interfaces de controle devem ser estabelecidas de forma que os consequentes da regra sejam realizados antes dos componentes especificados no antecedente da regra.

[7] Cada componente envolvido em uma regra de composição exclusiva deve possuir uma restrição indicando a relação de mútua exclusividade estabelecida na regra, incluindo os componentes envolvidos. Cada componente deve ser marcado com um estereótipo <<XOR>> no diagrama e não devem se comunicar via interfaces.

Notação

O elemento componente de processo de software possui a seguinte representação gráfica abaixo:

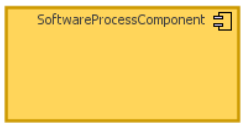


Figura 43 - Representação Geral de um Componente de Processo de Software

De acordo com as classificações atribuídas a um componente de processo, estereótipos serão utilizados para representá-las graficamente:

[1] Componente de Processo Opcional -> estereótipo: <<*optional*>>. Para componentes de processo mandatórios não serão atribuídos estereótipos.

[2] Componente de Processo Ponto de Variação -> estereótipo: <<*variation point*>>. Para os componente de processo variantes relacionados -> estereótipo: <<*variant*>>. Para componentes de processo invariantes não serão atribuídos estereótipos.

[3] Componente de Processo Compostos -> estereótipo: <<*composed*>>. Para componentes de processo simples não serão atribuídos estereótipos. No caso de

componentes de processos compostos por agrupamento via aplicação de critérios um estereótipo de composição específico será atribuído: <<*composed by grouping*>>.

[4] Componente de Processo Abstratos -> estereótipo: <<*abstract*>>. Para componentes de processo concretos não serão atribuídos estereótipos.

[5] Componente de Processo com restrições de mútua exclusividade -> estereótipo: <<XOR>>.

Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

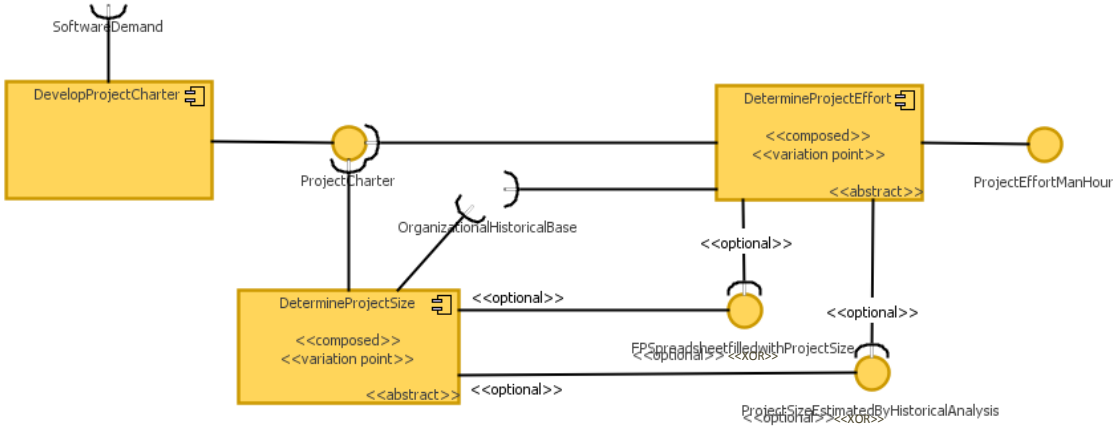


Figura 44 - Representação Geral de um Componente de Processo de Software

3.2. *OptionalityType* (TipoOpcionalidade)

Descrição

OptionalityType (TipoOpcionalidade) é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo de opcionalidade presente em um elemento do domínio. Pode assumir os seguintes valores: *mandatory* (mandatório) e *optional* (opcional).

1. *Mandatório*: são elementos obrigatórios no domínio. Classificação que pode ser aplicadas a componentes de processo e a interfaces. No caso de componentes, representam componentes que devem ser instanciados em todas as definições de processos derivadas a partir da LPrS modelada.
2. *Opcional*: são elementos opcionais no domínio. Classificação que pode ser aplicadas a componentes de processo e a interfaces. A escolha pela

seleção ou não dependerá do contexto do projeto específico. No caso de componentes, representam componentes cuja presenção não é mandatória nas definições de processos derivadas a partir da LPrS. A escolha pela seleção ou não dependerá do contexto do projeto específico.

Hierarquia

Sem hierarquia específica.

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

De acordo com as classificações atribuídas a um componente de processo, estereótipos serão utilizados para representá-las graficamente:

[1] Componente de Processo Opcional -> estereótipo: <<*optional*>>. Para componentes de processo mandatórios não serão atribuídos estereótipos.

Exemplo

O elemento componente de processo de software opcional possui a seguinte representação gráfica abaixo:

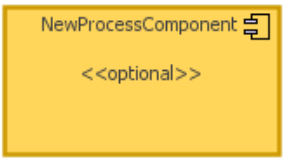


Figura 45 - Representação Geral de um Componente de Processo de Software Opcional

3.3. *VariabilityType* (TipoVariabilidade)

Descrição

VariabilityType (TipoVariabilidade) é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo de variabilidade presente em um componente de

processo. Pode assumir os seguintes valores: “ponto de variação”, “variante” e “invariante”.

1. *Pontos de variação*: são elementos 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

[1] No caso de componentes variantes de um mesmo ponto de variação mutuamente excludentes entre si, não devem ser estabelecidas interfaces entre eles, nem de dados nem de controle.

Notação

De acordo com as classificações atribuídas a um componente de processo, estereótipos serão utilizados para representá-las graficamente:

[1] Componente de Processo Ponto de Variação -> estereótipo: <<*variation point*>>. Para os componente de processo variantes relacionados -> estereótipo: <<*variant*>>. Para componentes de processo invariantes não serão atribuídos estereótipos.

Exemplo

Os elementos componentes de processo de software ponto de variação e variantes possuem a seguinte representação gráfica abaixo:

Figura 46 - Representação Geral de Componente de Processo de Software Ponto de Variação e Variante

3.4. *InternalStructureType* (TipoEstruturaInterna)

Descrição

InternalStructureType (TipoEstruturaInterna) é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo da composição interna de um componentes de processo. Pode assumir os seguintes valores: "simples" e "composto".

1. *Simples*: são componentes de processo estruturados internamente apenas por atividades e tarefas do domínio.
2. *Composto*: são componentes de processo estruturados internamente por outros componentes de processo.

Hierarquia

Sem hierarquia específica.

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

De acordo com as classificações atribuídas a um componente de processo, estereótipos serão utilizados para representá-las graficamente:

[1] Componente de Processo Compostos -> estereótipo: <<*composed*>>. Para componentes de processo simples não serão atribuídos estereótipos. No caso de componentes de processos compostos por agrupamento via aplicação de critérios um estereótipo de composição específico será atribuído: <<*composed by grouping*>>.

Exemplo

O elemento componente de processo de software composto possui a seguinte representação gráfica abaixo:

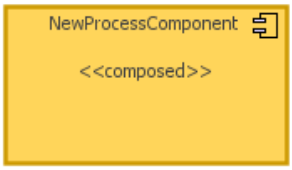


Figura 47 - Representação Geral de um Componente de Processo de Software Composto

3.5. *InternalVariationType* (TipoVariaçãoInterna)

Descrição

InternalVariationType (TipoVariaçãoInterna) é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo da variação interna de um componentes de processo. Pode assumir os seguintes valores: "concreto" e "abstrato".

1. *Concreto*: são componentes de processo sem nenhuma configuração adicional a ser realizada na definição de um processo derivado a partir da LPrS.
2. *Abstrato*: são componentes de processo com variação interna a ser resolvida, ou seja, possuem parte dos elementos de processo que os compõem com variação (variabilidade ou opcionalidade), restando decisões adicionais para o momento de definição de um processo derivado a partir da linha.

Hierarquia

Sem hierarquia específica.

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

De acordo com as classificações atribuídas a um componente de processo, estereótipos serão utilizados para representá-las graficamente:

[1] Componente de Processo Abstratos -> estereótipo: <<*abstract*>>. Para componentes de processo concretos não serão atribuídos estereótipos.

Exemplo

O elemento componente de processo de software abstrato possui a seguinte representação gráfica abaixo:

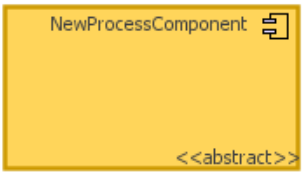


Figura 48 - Representação Geral de um Componente de Processo de Software Abstrato

3.6. *DataInterface* (InterfaceDados)

Descrição

Representa o meio de troca de dados entre componentes de processos e entre componentes e o meio. Corresponde à definição da relação de um produto de trabalho com unidade(s) de trabalho envolvida(s) na execução do componente que extrapola a fronteira do componente, ou seja, troca entre componentes ou do componente com o meio. Cada interface deve estar associada a um produto de trabalho que pode ser requerido (entrada do componente) ou provido (saída do componente).

Hierarquia

Superclasse: *Interface*

Atributos

- ***name* (nome)**: String [1] – atributo que define o nome da interface.

- ***optionalityType (tipoOpcionalidade): OptionalityType[1] (TipoOpcionalidade[1]) –*** atributo que define a classificação da interface quanto a sua opcionalidade. Pode assumir os valores *mandatory* (mandatório) e *optional* (opcional). Indica se a interface é mandatória no domínio, isto é, se estará presente em todos os processos instanciados a partir do modelo ou se é opcional. O tipo de opcionalidade é representado pela *lista enumerada OptionalityType (TipoOpcionalidade), seção 3.2. Default: mandatory (mandatório).*
- ***xorrestriction* (restrição de mútua exclusividade**): String [0..1] – representa que o produto de trabalho relacionado à interface participa de uma relação de mútua exclusividade.

Associações

- ***relatedWorkProduct*** (produto de trabalho relacionado): *ProcessFeature.WorkProduct* [1] – especifica um produto de trabalho relacionado ao componente como resultado (provido) ou insumo (requerido).
- ***relatedDataInterfaceType*** (relação com um componente de processo): *ProcessComponent [1]* – especifica a relação entre uma interface de dados e um componente. Para cada relação criada, deve-se especificar o tipo da relação que pode assumir os valores *"required" (requerida) e "provided" (provida)*. Indica se a interface representa uma relação de entrada ou saída de um produto de trabalho. O tipo de interface de dado é representado pela lista enumerada *DataInterfaceType*(TipoInterfaceDado), seção 3.7. Default: "*required*" (requerida). Além disso, o tipo de opcionalidade da relação também deve ser atribuído segundo os tipos representados pela *lista enumerada OptionalityType (TipoOpcionalidade), seção 3.2. Default: mandatory (mandatório).*

Restrições

[1] Cada interface de dados de um componente interno a um componente composto deve ser mapeada como uma interface de dados do componente composto e tais interfaces devem estar conectadas por uma relação de delegação.

[2] Uma interface de dados só pode assumir um tipo de interface de dados (requerida ou provida) quanto associada a um componente.

[3] É importante observar se componentes que participam de restrições de mútua exclusividade não requerem produtos uns dos outros.

Notação

O elemento interface de dado possui as seguintes representações gráficas abaixo:

- Interface de Dado Requerida

- Interface de Dado Provida

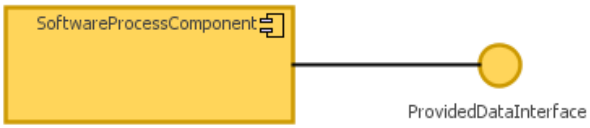
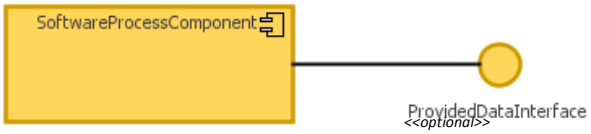


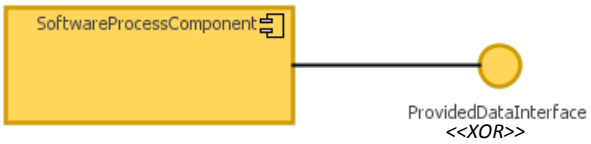
Figura 49 - Representações Gerais de uma Interface de Dado de Componente de Processo de Software

De acordo com as classificações atribuídas a uma interface de dados de um componente de processo, estereótipos serão utilizados para representá-las graficamente:

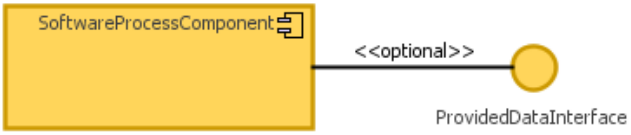
[1] Interface de Dados Opcional -> estereótipo: <<*optional*>>. Para interface de dados mandatórias não serão atribuídos estereótipos.



[2] Interface de Dado em que o produto de trabalho relacionado está envolvida em restrições de mútua exclusividade -> estereótipo: <<XOR>>.



[3] Relação Interface de Dado e Componente Opcional - estereótipo: <<*optional*>>. Para relações mandatórias não serão atribuídos estereótipos.



Exemplo

O exemplo abaixo se refere ao domínio de Planejamento de Projetos de Software.

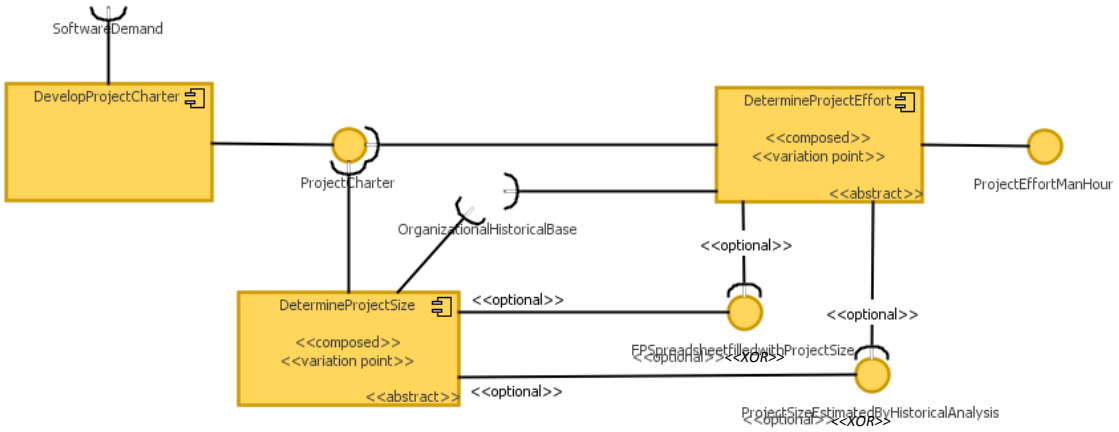


Figura 50 - Representação de Interfaces de Dados em um Modelo de Componentes de Processo de Software

3.7. DataInterfaceType

DataInterfaceType (TipoInterfaceDado) é uma classe do tipo "*ennumeration*", cujos literais determinam o tipo de uma interface de dados estabelecida para um Componente de Processo. Pode assumir os valores: "*required*" (requerida) e "provided" (provida), onde:

1. *required* (requerida): define um produto de trabalho especificado como entrada do componente; e
2. *provided* (provida): define um produto de trabalho especificado como saída do componente.

Hierarquia

Superclasse: Sem hierarquia específica.

Atributos

Sem atributos.

Associações

Sem associações específicas.

Restrições

Sem restrições específicas.

Notação

Olhar descrição Seção 3.6.

Restrições

Olha exemplo Seção 3.6.

3.8. ControlInterface

Representa o fluxo de execução entre componentes de processos. Uma interface de controle compreende um componente como origem e outro componente destino do fluxo. Entre eles um conector é utilizado para estabelecer uma regra de associação entre os componentes. Nessa abordagem, a regra de associação é presentada pelos tipos de fluxos de controle especificados no nível de análise: sequencial, paralelo, síncrono e de decisão.

Hierarquia

Superclasse: *Interface*

Atributos

- *controlInterfaceType* (tipoInterfaceControle): *ControlFlow*[1] (FluxoDeControle[1]) – atributo que define a classificação da interface de controle quanto ao seu tipo. Pode assumir os valores de um dos tipos de

fluxos de controle disponíveis no Nível de Análise. Desta forma, uma interface de controle pode ser do tipo: (1) Fluxo de Sequência (*SequenceFlow*); (2) Fluxo Paralelo (*ParallelFlow*); (3) Fluxo com sincronismo (*SynchronousFlow*); e (4) Fluxo de Decisão (*DecisionFlow*), conforme descrita na seção 2.2.10, trocando as associações de origem e destino de unidade de trabalho para componente de processo.

Associações

- *source* (origem): Referencia o componente de processo que representa a origem (início de execução) do fluxo. No caso do tipo de interface de controle ser um Fluxo com sincronismo (*SynchronousFlow*), a origem é representada por mais de um componente de processo.
- *target* (destino): Referencia o componente de processo que representa o destino origem (fim de execução) do fluxo. No caso do tipo de interface de controle ser um Fluxo Paralelo (*ParallelFlow*), o destino é representada por mais de um componente de processo.

Restrições

[1] É importante observar se os componentes destinos não produzem resultados requeridos por nenhum dos componentes origens ou componentes anteriores a sua execução.

Notação

A representação dos fluxos seguem a representação proposta no nível de análise.

Exemplo

Figura 51 - Representação de Interfaces de Controle em um Modelo de Componentes de Processo de Software

MAGDALENO, A. M., "***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, 2010.

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

TEIXEIRA, E.N., "***OdysseyProcess-FEX: Uma Abordagem para Modelagem de Linha de Processos de Software***". Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2011.