Uma extensão do BPMN para modelagem de Processos de Desenvolvimento de Software: BPMN*t*

Fabio L. Fonseca[1], Toacy C. Oliveira[1], Eliana B. Pereira[2]

[1]PESC/COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

[2]FACIN – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Rio Grande do Sul – RS – Brasil

{fabiolf, toacy}@cos.ufrj.br, eliana.pereira@pucrs.br

Abstract. *The BPMN 2.0 is a meta-model and notation widely used in the Market to model business process featuring a high semantic representation capability. Nevertheless, when modeling Software Development Processes using BPMN, one can observe the lack of an important feature: process tailoring. This article proposes the BPMNt, a BPMN conservative extension that aims to aggregate a tailoring mechanism presented in SPEM 2.0. BPMNt uses the extensibility classes already present in the BPMN meta-model. Also, it proposes a set of consistency rules used to evaluate the final process.*

Resumo. *O BPMN 2.0 é um meta-modelo e notação para a modelagem de processos de negócio bastante difundida no mercado apresentando uma alta representatividade semântica. No entanto, ao modelar Processos de Desenvolvimento de Software utilizando o BPMN, nota-se a falta de uma importante capacidade, a de realizar process tailoring. Este trabalho propõe o BPMNt, que é uma extensão conservativa do BPMN que visa incorporar um mecanismo de tailoring presente no SPEM. O BPMNt utiliza as classes de extensibilidade já existentes no meta-modelo do BPMN. Propõe, ainda, um conjunto de regras de consistência utilizadas para validar o processo final.*

1. Introdução

Com a evolução do software e a sua crescente importância para a manutenção e aprimoramento do estilo de vida moderno, torna-se essencial ações no sentido de garantir a sua qualidade. Neste contexto crescem de importância as técnicas para modelagem do Processo de Desenvolvimento de Doftware (PDS), que visam aumentar o nível de qualidade [Fuggetta, 2000] das etapas de criação do software, desde sua concepção até sua implantação, tornando o processo padronizado e pré-estabelecido.

No entanto, assim como existem sistemas voltados para domínios diferentes, existem PDSs com objetivos diferentes e mais adequados a determinados projetos, equipes e sistemas. Não existe hoje um processo universal que atenda a todos os tipos de projetos [Lindvall & Rus, 2000].

Além disso, um PDS deve ser facilmente entendível por pessoas, já que são seres humanos que executarão a maior parte das suas atividades [Fuggetta, 2000]. Um artefato normalmente utilizado para apoiar a documentação de informações dos

processos são os modelos e diagramas que o expõem de maneira visual, facilitando, assim, o seu entendimento.

Um PDS pode ser modelado de diversas maneiras. Uma delas é utilizando o SPEM [OMG, 2010] (*Software & Systems Process Enginnering Meta-Model*), atualmente na versão 2.0. O SPEM 2.0 é um meta-modelo para a especificação de processos de software baseado no MOF 2.0 [OMG, 2006] (*Meta Object Facility*). Foi concebido para ser utilizado na descrição de um ou uma família relacionada de PDSs.

Porém, ainda que existam vários PDSs apropriados às várias abordagens possíveis para o desenvolvimento de software, na maioria das vezes um PDS deve ser customizado para um projeto que, por exemplo, deixa de executar algumas atividades e/ou as troca por outras atividades específicas. Isto é conhecido como *Process Tailoring*, que consiste na manipulação de um PDS incorporando novos elementos ou removendo elementos existentes [Pereira *et al*, 2008] porém não necessários para o projeto de software que o está utilizando. O SPEM prevê este tipo de adaptação através de um mecanismo de reutilização de processos (Seção 2.1).

Existe, ainda, outra forma de enxergar um PDS, como sendo um Processo de Negócio, onde o negócio é o desenvolvimento de software [Araújo *et al*, 2004]. Ao utilizar esta abordagem, é possível contar com todo o ferramental disponível para a prática de BPM (*Business Process Management*). Na disciplina de Processos de Negócio, existem ferramentas e notações próprias para modelagem de processo. Uma das principais é o BPMN [OMG, 2004] (*Business Process Model and Notation*). Atualmente, também em sua versão 2.0, o BPMN conta com uma notação extremamente rica em termos semânticos e um meta-modelo que a define sendo bastante difundido na prática de modelagem de Processos de Negócio. Não é o seu foco, entretanto, a modelagem de PDS, que tem necessidades específicas.

O objetivo deste trabalho é aliar a riqueza de expressividade do BPMN à uma necessidade específica da modelagem de PDS, adicionando a ele a capacidade de *Process Tailoring*. Isto será feito por meio de uma extensão conservativa do BPMN, batizada de BPMN*t*. Além disso, serão definidas regras que conservarão o PDS modelado através do BPMN*t* válido.

Este trabalho está dividido como segue: a Seção 2 resume o SPEM, detalhando seu mecanismo de *Process Tailoring*; a Seção 3 descreve o BPMN; a Seção 4 apresenta a extensão proposta neste trabalho; a Seção 5 apresenta os trabalhos relacionados e finalmente, na Seção 6, as conclusões e trabalhos futuros são apresentados.

2. SPEM

O SPEM 2.0 é tanto um meta-modelo baseado no MOF 2.0 (*Meta Object Facility*) quanto uma *profile* da UML 2 para a especificação de processos de software. O meta-modelo do SPEM é definido por meio de pacotes (Figura 1) e descreve as estruturas necessárias para expressar formalmente processos de software. O pacote *Core* é o principal e define as estruturas básicas do meta-modelo. Os demais pacotes utilizam estas estruturas através do mecanismo *merge*[1].

[1] *Merge* de pacotes é um relacionamento entre dois pacotes onde o conteúdo do pacote de destino (o que é apontado pela seta) é combinado com o conteúdo do pacote de origem através de especializações e redefinições, quando aplicáveis. [OMG, 2007]

Este trabalho será baseado no pacote *Process Structure* (e, por consequência, no pacote *Core*). Com estes pacotes já é possível modelar minimamente um PDS incluindo *activities*, *work products* e *roles* e organizá-lo em uma hierarquia de WBS[2] estática. Além disto, este pacote apresenta um mecanismo de reutilização de elementos através de herança que permite realizar *tailoring* de processos, no qual será baseada a extensão proposta. Este mecanismo é detalhado na próxima seção.

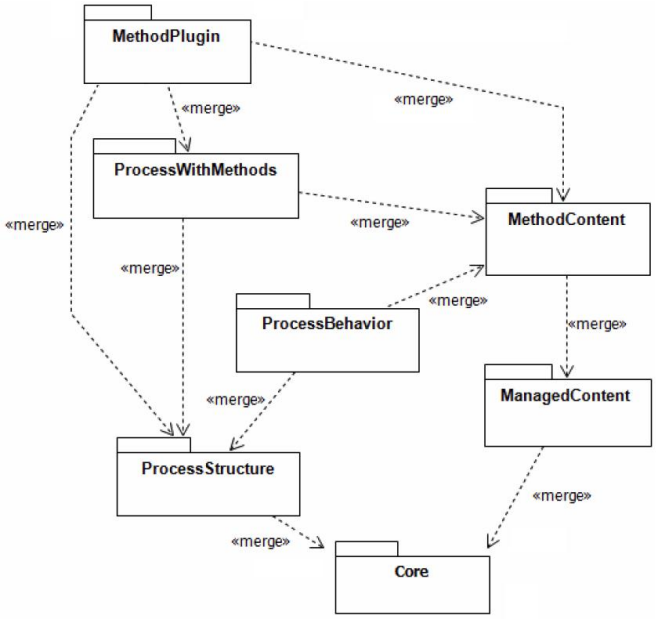


Figura 1: Divisão de pacotes do meta-modelo do SPEM

2.1. Mecanismo de reutilização/*tailoring* do SPEM

A modelagem deste mecanismo se dá diretamente através de relacionamentos da classe `Activity`. Esta classe no SPEM 2.0 é também um `BreakdownElement` e define uma atividade que pode ser desde uma unidade básica de trabalho dentro de um processo até o próprio processo, conforme mostra a Figura 2. Nesta figura, é possível verificar a classe `Activity` e alguns de seus relacionamentos relevantes ao assunto sendo tratado:

- nestedBreakdownElement: Este atributo de relacionamento representa o aninhamento de atividades. Como a classe `BreakdownElement` é pai de `Activity` isto define uma estrutura hierárquica onde uma `Activity` pode representar um processo inteiro, com seus elementos filhos.
- usedActivity: Esta associação define uma reutilização de uma atividade e é usado em conjunto com o atributo *useKind* de acordo com a semântica definida na enumeração *ActivityUseKind* cujos valores podem ser:
  - na: Este é o valor padrão para atividades que não utilizam reutilização, ou seja, não possuem associação do tipo *usedActivity*.
  - extension: É o mecanismo básico do processo de reutilização. Uma determinada atividade, ao utilizar o relacionamento *usedActivity* escolhendo

[2] WBS significa *Work Breakdown Structure*, e é utilizada para agrupar hierarquicamente atividades de um processo de maneira a melhorar sua organização e visualização. [OMG, 2007]

useKind igual a *extension* faz com que todos os elementos que façam parte da hierarquia abaixo da atividade apontada sejam herdados pelo processo reutilizador.

- localContribution: Define um mecanismo para adicionar atividades à uma estrutura hierárquica reutilizada através de uma extensão (em uma atividade pai). O processo resultante fica, então, com as atividades herdadas do processo que se está reutilizando, além das novas atividades adicionadas, filhas da atividade em que se define o *localContribution*.
- localReplacement: Define um mecanismo para redefinir algumas partes do processo que está sendo reutilizado através de uma extensão (em uma atividade pai). As atividades herdadas são substituídas hierarquia definida pela atividade de origem do relacionamento onde se define o *localReplacement*.

- supressedBreakdownElement: Este atributo permite esconder qualquer `BreakdownElement` da estrutura do processo reutilizado. Para isto, basta apontá-lo para o elemento a ser suprimido, desde que seja filho do processo reutilizado.

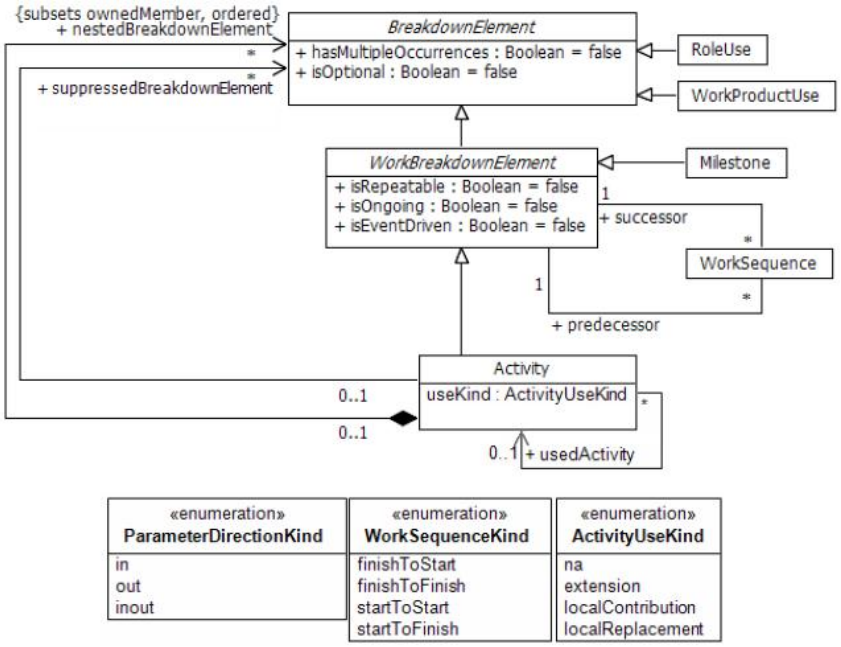


Figura 2: Modelo do pacote *Process Structure*

A Figura 3 ilustra o mecanismo de reutilização. Neste exemplo é definido um Processo 2 que reutiliza um Processo 1 realizando algumas adaptações, o que pode ser visto como um *tailoring*. A primeira atividade é a de mais alto nível hierárquico, que é o próprio *Process 2*. Esta atividade está relacionada à atividade *Process 1* através do relacionamento *usedActivity* estando o atributo *useKind* definido como *extension*, o que provoca a herança de toda a sua estrutura, que é definida através do relacionamento *nestedBreakdownElement*. Neste exemplo, no entanto, a estrutura é modificada através de operações de *supression*, *localContribution* e *localReplacement*.

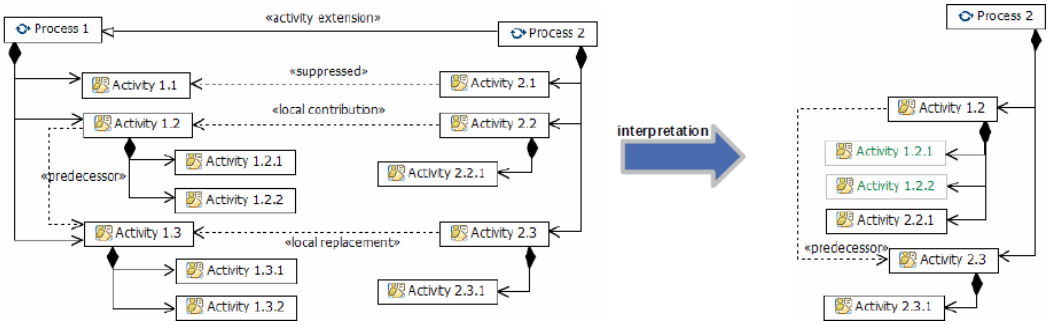


Figura 3: Ilustração do processo de reutilização previsto no SPEM 2.0

A primeira modificação é a supressão da *Activity 1.1*. Apesar de na Figura 3 existir uma atividade de nome *Activity 2.1* ligada à *Activity 1.1*, de fato o que acontece é que a própria atividade *Process 2* aponta para a *Activity 1.1* através do relacionamento *suppressedBreakdownElement*. Com isto, no momento da interpretação do *Process 2* para a geração da sua forma final, a *Activity 1.1* é removida da hierarquia.

Já a *Activity 2.2* se relaciona diretamente com a *Activity 1.2* através do relacionamento *usedActivity*. Neste caso o atributo *useKind* é definido como *localContribution*, o que provoca a adição dos seus filhos à *Activity 1.2*, que prevalece no processo final.

Finalmente, a *Activity 2.3* se relaciona com a *Activity 1.3* com o atributo *useKind* definido como *localReplacement*. Neste caso, a *Activity 2.3* e seus filhos se sobrepõem completamente à *Activity 1.3*, redefinindo o processo reutilizado.

Assim, através da reutilização obtém-se um novo processo adaptando um processo existente. A implementação desta capacidade de reutilização/*tailoring* no BPMN 2.0 será abordada nas próximas seções.

3. BPMN

Segundo a sua especificação [OMG, 2010], a versão 2.0 do BPMN tem como principal objetivo definir uma notação que seja facilmente compreensível por todos os usuários de negócio, desde o analista de negócios que cria as versões iniciais do processo, passando pelos desenvolvedores responsáveis por implementar a tecnologia que executará estes processos até, finalmente, as pessoas de negócio que vão mantê-los e monitorá-los. Com isso, o BPMN 2.0 preenche uma lacuna existente entre o desenho do processo de negócio e sua implementação, sendo utilizável no dia a dia por humanos além de ser facilmente mapeado para uma linguagem de execução como o WSBPEL.

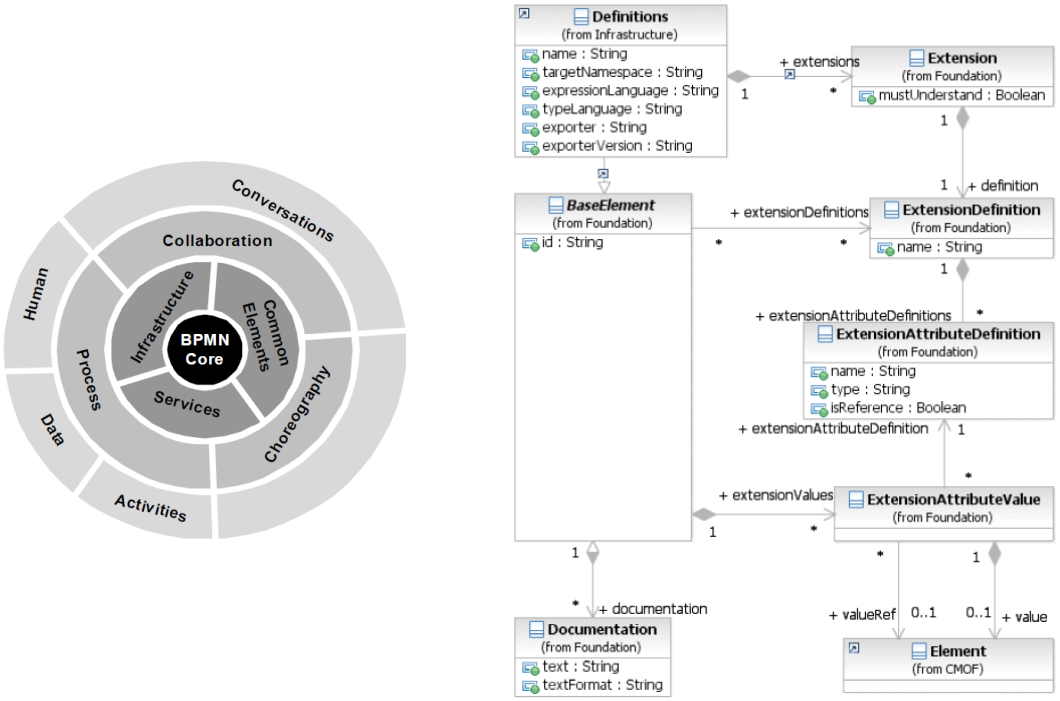
Para isso, o BPMN 2.0 define um meta-modelo e uma notação para representação de Processos de Negócio. Não faz parte do seu escopo, entretanto, a definição de modelos organizacionais, modelo de regras de negócio, modelos de estratégia e modelos de dados e informações, por exemplo.

Para representar processos de negócio, a notação BPMN 2.0 é bastante rica semanticamente. Em [Thom e Iochpe, 2009] é possível visualizar toda a grande variedade de elementos gráficos, utilizados para representar cada aspecto de um processo, incluindo fluxos de exceção e alternativos.

O meta-modelo do BPMN 2.0 é organizado em camadas em uma estrutura parecida com a de pacotes do SPEM. Cada camada mais externa depende das camadas mais centrais para adicionar mais capacidades ao modelo, como mostra a Figura 4.a.

3.1. Classes de extensão do meta-modelo do BPMN 2.0

O meta-modelo que define o BPMN 2.0 contém um conjunto de classes que já preveem a capacidade de extensibilidade. Na Figura 4.b é possível visualizá-las em um diagrama.



(a) (b)

Figura 4: (a) Modelo de camadas do BPMN 2.0 (b) Conjunto de classes que definem o mecanismo de extensibilidade do BPMN 2.0

Uma extensão BPMN 2.0 consiste basicamente em 4 classes: `Extension`, `ExtensionDefinition`, `ExtensionAttributeDefinition` e `ExtensionAttribute-Value`. A classe `ExtensionAttributeDefinition` define uma lista de atributos que podem ser adicionados a qualquer elemento filho de `BaseElement`. Esta lista define o nome e o tipo do novo atributo. Isto permite que os analistas possam integrar qualquer meta-modelo existente ao BPMN 2.0, reutilizando seus elementos. A classe `ExtensionDefinition` por si só pode ser criada de maneira independente de qualquer elemento ou definição BPMN. Para utilizar uma `ExtensionDefinition` em um modelo esta deve estar associada à classe `Extension` que faz a sua ligação com a definição de um modelo BPMN específico (classe `Definition`). A classe `Extension` é um atributo da classe `Definition` e, através dela, podemos associar um `ExtensionDefinition` a qualquer elemento filho de `BaseElement`. Além disso, qualquer elemento BPMN "estendido" contém a classe `ExtensionAttributeValue`, que representa o valor do atributo. Esta classe é do tipo `Element` e tem uma associação à sua correspondente definição `ExtensionAttributeDefinition`.

Desta maneira é possível estender qualquer `BaseElement` e quaisquer das suas classes filhas. Na Figura 5 é possível visualizar algumas classes filhas de `BaseElement`. Para este trabalho, interessam particularmente as classes `FlowElement` e `FlowElement-Container` através das quais pode-se representar um processo e seus sub-processos (ou coreografias) em uma estrutura hierárquica semelhante à utilizada no SPEM.

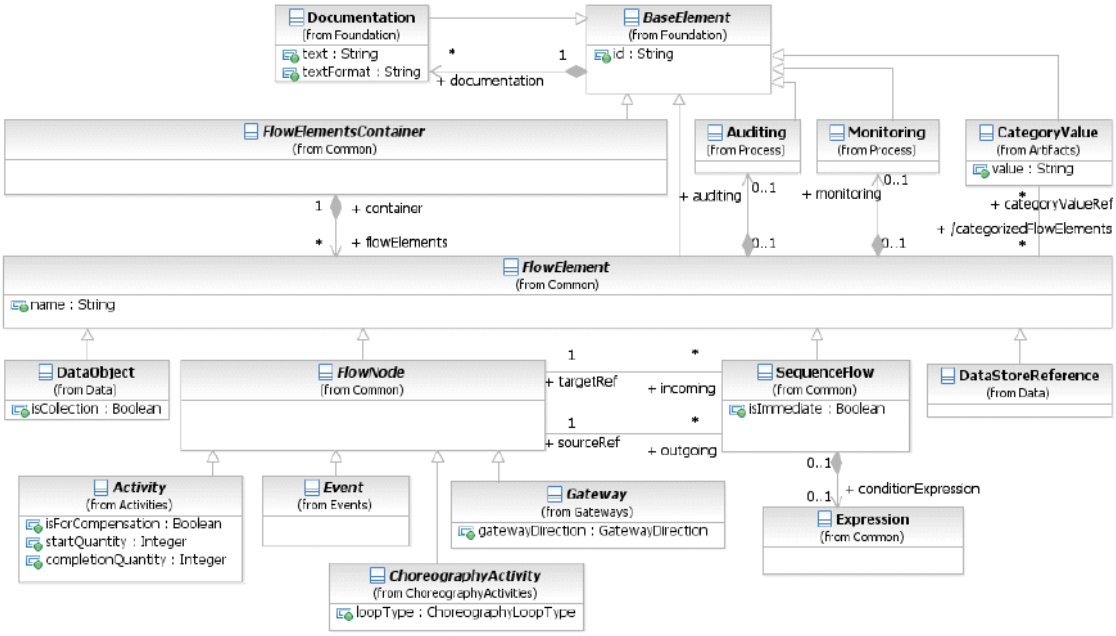


Figura 5: Diagrama de classes exibindo o relacionamento entre a classe `BaseElement` e as demais classes que compõem um Processo ou Coreografia

4. Estendendo o BPMN 2.0 (BPMN*t*)

Para incluir no BPMN a capacidade de realizar *Process Tailoring* e assim ser capaz de utilizá-lo na prática para representar PDSs, foi realizada uma extensão conservativa utilizando as classes descritas na Seção 3.1. Através desta extensão, foi incluída nestes elementos a capacidade de serem suprimidos, substituídos e complementados, da mesma maneira que é feito no SPEM, através da funcionalidade de reutilização de processos. Na Figura 6 pode-se visualizar uma representação da classe `BaseElement-Instanciado` mostrando somente os atributos adicionados pela extensão proposta, bem como as classes da extensão que originaram cada um deles. Nesta Figura o relacionamento de dependência é utilizado para associar o atributo à sua origem e não representa semanticamente uma dependência. Os três novos atributos serão usados nas classes `FlowElementsContainer` e `FlowElement` e têm a mesma semântica do SPEM:

- usedBaseElement: Da mesmo modo que no SPEM existe a associação determinada pelo atributo *usedActivity*, na classe instanciada a partir de `FlowElements-Container` ou `FlowElement` estendidas foi incluído o atributo *usedBaseElement* que deve se relacionar com a classe a ser reutilizada. Ele tem a sua semântica definida a partir da enumeração *ElementUseKind* que pode assumir os mesmos valores da enumeração análoda do SPEM: *na*, que é o valor assumido quando o relacionamento *usedBaseElement* não for utilizado, *extension* para definir a herança de uma estrutura hierárquica a partir de uma instância da classe `FlowElementsContainer`; *localContribution* para definir uma adição a uma hierarquia herdada e *localReplacement*, que define uma substituição de um `FlowElementsContainer` herdado por outro definido localmente. Todos estes relacionamentos são realizados entre duas instâncias de classes filhas de `FlowElementsContainer`.

- supressedBaseElement: Este atributo de relacionamento permite esconder qualquer `BaseElement` da estrutura do processo. É usado no contexto da reutilização de um processo através do atributo *usedBaseElement*.

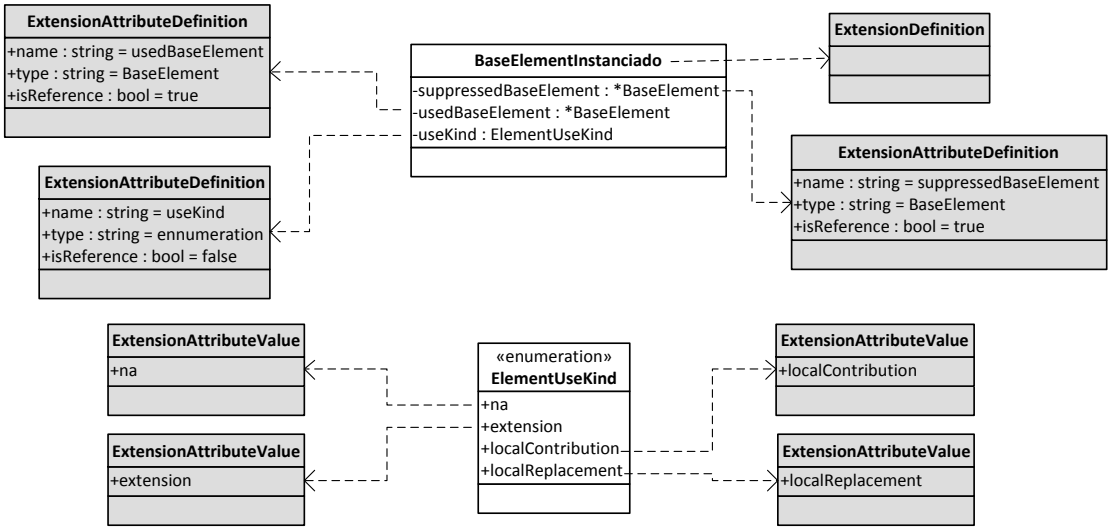


Figura 6: Representação da extensão proposta

4.1. Características do BPMN*t*

Uma extensão conservativa, por definição, não invalida nenhuma das regras já presentes no modelo original, mas pode adicionar novas regras que valem no contexto da extensão. Porém, o BPMN*t* herda algumas características do SPEM, já que a proposta é a de reproduzir no BPMN a sua funcionalidade de reutilização/*tailoring*.

Uma das características do BPMN*t* é a necessidade de existência de níveis hierárquicos no processo modelado. Qualquer das operações descritas deve ser realizada sempre em pelo menos dois níveis hierárquicos. Uma *extension*, onde o processo reutilizador herda elementos do processo reutilizado, é realizada no nível de uma subclasse de `FlowElementsContainer` e provoca a herança de todos os elementos filhos, ficando o processo reutilizador também com dois níveis hierárquicos.

Da mesma maneira, uma operação de *localContribution* ou *localReplacement* deve ser realizada em algum elemento herdado do processo reutilizado. Ou seja, no segundo nível hierárquico, conforme colocado acima. Logo deve existir em ambos os processos (reutilizado e reutilizador) pelo menos 3 níveis, sendo o primeiro o nível da *extension*, o segundo o nível do *localContribution* ou *localReplacement* e o terceiro, onde ficam os elementos que serão adicionados ou substituídos. Uma operação de *supress* necessita apenas de uma *extension* onde a mesma instância de `FlowElementsContainer` realiza a *extension* e já aponta para o elemento que deve ser suprimido, podendo ser realizada, então com apenas dois níveis hierárquicos.

No SPEM, as operações acima são realizadas no nível da instância da classe `Activity` onde em uma *extension* todos as instâncias de `BreakdownElement` filhas são herdadas pelo processo reutilizador. No BPMN a herança acontece no nível da classe `FlowElementsContainer` com filhos do tipo `BaseElement`. Logo, neste sentido, as adaptações no processo devem respeitar algumas regras de consistência para não invalidar o seu fluxo de execução, descritas na Tabela 1.

Tabela 1: Regras de consistência do BPMN*t*

Operação	#	Regra
Supress	1	O fluxo que de chegada do elemento suprimido passa a ser o fluxo de chegada do elemento imediatamente seguinte ao suprimido.
Supress	2	Não é possível suprimir um *gateway*[3] que divide um fluxo do processo em mais de um fluxo de saída sem remover completamente pelo menos um dos fluxos, de forma que só exista no final um fluxo linear.
Supress	3	Não é possível suprimir um *gateway* que unifique mais de um fluxo sem remover inteiramente pelo menos mais um fluxo de entrada, de modo que no final só exista um fluxo chegando ao elemento imediatamente posterior ao suprimido.
Supress	4	Artefatos ou *Data Objects*[4] ligados ao elemento suprimido também devem ser suprimidos.
Supress	5	Fluxos de compensação, erro, ou qualquer outro ligado ao elemento suprimido também devem ser suprimidos inteiramente.
Supress	6	Ao suprimir um fluxo inteiro de saída ou entrada de um *gateway*, caso só reste apenas um fluxo entrando ou saindo deste *gateway*, este também será removido e o último elemento do fluxo restante imediatamente anterior ao *gateway* será ligado ao elemento imediatamente posterior.
local-Contribution	7	Não podem ser adicionados *gateways* que tenham fluxos de entrada ou saída não utilizados ou *Data Objects* não ligados a nenhuma atividade, respeitando as regras já existentes no BPMN
local-Replacement	8	O lugar da instância de `FlowElementsContainer` substituída deverá ser ocupado por outra instância com o mesmo número de fluxos de entrada e saída.

4.2. Exemplo de Uso do BPMN*t*

Para ilustrar a utilização do BPMN*t* é apresentado um exemplo de *tailoring* na Iteração de Concepção do *Delivery Process* do OpenUP[5], um PDS baseado no RUP.

A Figura 7 mostra as atividades da Iteração de Concepção do OpenUp modelada utilizando BPMN. Além disso, a Figura detalha algumas tarefas e os produtos de trabalho que entram (consumidos) e saem (produzidos e/ou modificados) em cada uma das atividades. Para modelar as atividades e tarefas do OpenUP foram utilizados, respectivamente, os sub-processos e atividades do BPMN. Já para representar os produtos de trabalho foi utilizado o elemento *Data Object*.

Para este exemplo foram consideradas algumas operações de exclusão (*supress*) e modificação (*localContribution*) de atividades da Iteração Concepção. Inicialmente, a exclusão do sub-processo Identificar e Refinar Requisitos foi realizada, pois, neste exemplo, os requisitos do projeto sendo adaptado já estariam prontos (provenientes de

[3] *Gateways* são usados para controlar como o fluxo do Processo converge e diverge ao longo do diagrama

[4] *Data Objects* são elementos do BPMN que representam dados

[5] Disponível em http://www.eclipse.org/epf/openup_component/openup_vision.php

outro projeto realizado anteriormente). Outra operação realizada foi uma contribuição no sub-processo Iniciar Projeto com a inclusão da nova atividade Apresentar Plano do Projeto. O objetivo da nova atividade é a aprovação do plano de projeto pelos *stakeholders* visando o aumento do comprometimento destes *stakeholders* no projeto. Foi definido como entrada para a nova atividade o *Data Object* Plano de Projeto.

Para realizar a criação do processo adaptado, além da criação de um sub-processo que representa a Iteração de Concepção original do OpenUP (Figura 7), foi necessária a criação de um novo sub-processo (para ser adaptado) que é mostrado na Figura 8. Neste novo processo foi criado um relacionamento de *extend* com o sub-processo Iteração de Concepção para herdar os seus elementos. Além disso, como era necessário realizar as operações de exclusão e contribuição, foram incluídas duas novas relações no sub-processo adaptado (Figura 8): (1) um relacionamento do sub-processo criado com o sub-processo Identificar e Refinar Requisitos através do atributo *supressedBaseElement* (para exclusão) e (2) um relacionamento do tipo *local-Contribution* do sub-processo Iniciar Projeto Adaptado (que possui a nova atividade Apresentar Plano de Projeto) com o sub-processo Iniciar Projeto (para contribuição).

É importante considerar que, com a exclusão do sub-processo Identificar e Refinar Requisitos, o *gateway* imediatamente anterior ao sub-processo suprimido ficou com apenas uma saída. De acordo com a Regra 6 da Tabela 1, este *gateway* também foi suprimido, sendo estabelecida a ligação entre as atividades Iniciar Projeto Adaptado e Obter Acordo Sobre Abordagem Técnica.

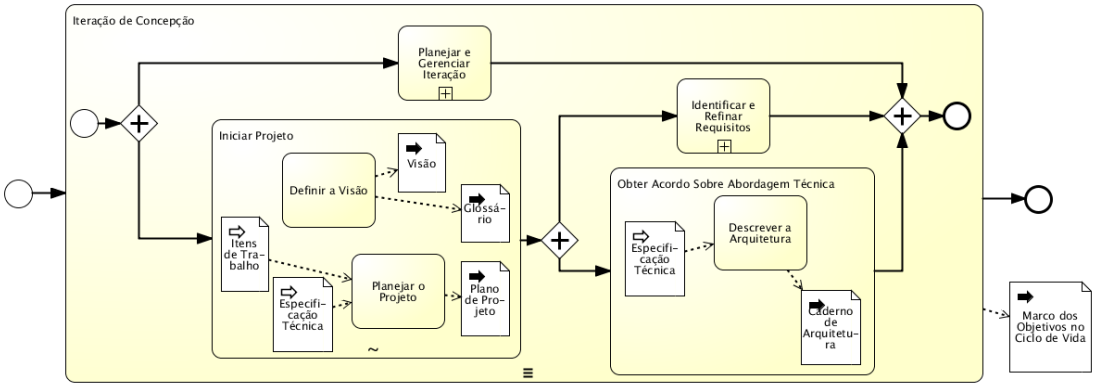


Figura 7: A fase de Concepção do *Delivery Process* do OpenUp

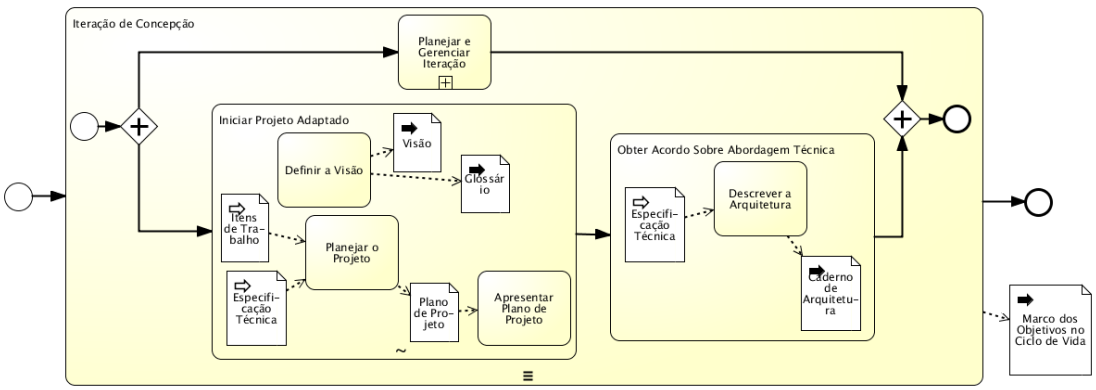


Figura 8: Processo final depois de realizado o *tailoring*

5. Trabalhos relacionados

Em seu trabalho sobre uma extensão conservativa do BPMN para melhorar sua representação de elementos de dados, [Magnani & Montesi, 2009] discutem sobre uma deficiência na modelagem de dados do BPMN e propõem o BPDMN (*Business Process and Data Modeling Notation*), baseado no BPMN 1.2. Alguns dos conceitos propostos aparecem incluídos na versão 2.0 do BPMN como, por exemplo, o elemento *Data Store*.

Em [Gagné & Trudel, 2009] mais uma extensão para o BPMN é proposta, desta vez com o objetivo de se adicionar a representação de tempo decorrido nos diagramas modelados. O trabalho propõe uma representação das atividades ao longo do tempo introduzindo restrições temporais como, por exemplo, *Must Start On*, *Start No Earlier Than* e *Finish No Later Than*. Além disso, cria um novo relacionamento entre atividades representando graficamente restrições do tipo *Finish-to-Start* e *Start-to-Start* que são também encontradas no SPEM. No entanto, este trabalho também utiliza a versão 1.2 do BPMN e não representa estes elementos temporais como uma extensão do seu meta-modelo.

Na linha de trabalhos que abordam a área sobre process *tailoring* em PDSs, podemos citar [Pereira *et al*, 2008]. Este trabalho propõe uma extensão ao meta-modelo do RUP incluindo elementos e relacionamentos relacionados com *process tailoring*. Além disso, os autores apresentam um conjunto de regras de boa-formação para conduzir operações de inclusão e exclusão de elementos em processos de software baseados no RUP. O principal objetivo do meta-modelo e regras de boa-formação é garantir a conformidade dos processos adaptados com o processo original.

6. Conclusão e Trabalhos futuros

Neste trabalho foi apresentada uma extensão conservativa para o BPMN 2.0, o BPMN*t*. Esta extensão tem o objetivo de adicionar ao BPMN 2.0 a capacidade de *Process Tailoring*, inspirada no mecanismo de reutilização do SPEM 2.0. Com isso, é possível utilizar toda a capacidade semântica do BPMN 2.0 para modelar um PDS.

Como a extensão foi baseada no SPEM 2.0, algumas das suas características foram herdadas. Uma delas foi a necessidade de se modularizar o processo estruturando-o em níveis hierárquicos, já que o SPEM é utiliza fortemente o conceito de WBS. A estratégia foi representar este conceito no BPMN através do aninhamento de sub-processos e sub-coreografias, o que é um pouco diferente da proposta do SPEM, mas que atendeu completamente ao objetivo proposto.

O BPMN 2.0 já prevê em seu meta-modelo classes próprias para que sejam realizadas extensões. Com estas classes foi possível adicionar novos atributos às classes já existentes e através deste mecanismo o BPMN*t* foi criado. Para que os processos gerados com o BPMN*t* ficassem íntegros e consistentes, foi necessário definir algumas regras de validação que garantem que o processo final adaptado seja executável. Estas regras devem ser aplicadas no momento da interpretação do processo reutilizador para a geração da sua versão final. Uma ferramenta como o EPF Composer[6], por exemplo, realiza esta interpretação interativamente no momento da adequação do processo reutilizador, o que também é uma estratégia válida.

[6] Disponível em http://www.eclipse.org/epf/

A partir deste trabalho, é possível visualizar diversas futuras frentes para desenvolvimento e pesquisa. A primeira delas é modelar detalhadamente um PDS real utilizando BPMN*t* incluindo as dependências entre as tarefas e os fluxos de atividades executados quando algo dá errado. Normalmente estas decisões a respeito do que fazer ficam a cargo do Gerente do Projeto que teria o seu trabalho simplificado por um caminho alternativo já previamente definido, em um momento onde as pressões do projeto ainda não o estão obrigando a tomar decisões rápidas e, muitas vezes, impensadas, que podem vir a comprometer o projeto. A partir deste modelo realizar experimentos de utilização para colher evidências reais da utilidade do trabalho.

Outra frente de trabalho possível é a criação de uma ferramenta (ou adequação de uma ferramenta já existente) para facilitar o processo de *tailoring* utilizando BPMN*t*.

Por fim, como trabalho futuro, a questão da herança da estrutura hierárquica do SPEM pode ser analisada sob outra ótica, fazendo com que o BPMN*t* seja adaptado para se tornar mais flexível e sem as restrições impostas pela estratégia adotada.

7. Referências

Araujo, R., Capelli, C., Gomes, A., Pereira, M., Iendrike, H., Ielpo, D., Tovar, J. (2004) "A Definição de Processos de Software sob o ponto de vista da Gestão de Processos de Negócio". VI Simpósio Internacional de Melhoria de Processos de Software (SIMPROS). Departamento de Informática Aplicada/UNIRIO, RJ, Brasil.

Fuggetta, A. (2000), "Software process: a roadmap," *Proceedings of the Conference on the Future of Software*, ICSE'00, pp. 25 - 34.

Lindvall, M., Rus, I. (2000) "Process diversity in software development," *Software, IEEE*, vol. 17, 2000, pp. 14-18.

Magnani, M. e Montesi, D. (2009) "BPDMN: A conservative extension of BPMN with enhanced data representation capabilities." In Proceedings of CoRR, 2009.

OMG (2010) "Business Process Model and Notation (BPMN), Version 2.0 (2010)", disponível em <http://www.omg.org/spec/BPMN/2.0>.

OMG (2006) "Meta Object Facility (MOF) Core Specification, Version 2.0", disponível em <http://www.omg.org/spec/MOF/2.0/PDF>.

OMG (2004) "Software Process Engineering Metamodel (SPEM) 2.0" RFP, Nov. 2004

OMG (2007) "OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2" disponível em <http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF>

Pereira, E., Bastos, R. e Oliveira, T. (2008) "Process tailoring based on well-formedness rules," *SEKE'08: Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering*, 2008, pp. 185-190.

Thom, L., Iochpe, C. (2009) "Poster da BPMN 2.0 - Notação e Modelo de Processo de Negócio", Disponível em <http://www.bpmb.de/images/BPMN2_0_Poster_PT.pdf>, acessado em: 03 de setembro de 2010.

Gagné, D. e Trudel, A. (2009) "Time-BPMN", *IEEE Conference on Commerce and Enterprise Computing, CEC'09*, IEEE, pp. 361–367.