

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
ESCOLA DE INFORMÁTICA APLICADA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Um Estudo da Representação Gráfica de Aspectos em BPM utilizando a
ferramenta ARIS Express 2.2

Nome da Autora:

Clarissa da Silva Correa Romeiro

Nome das Orientadoras:

Flávia Maria Santoro

Claudia Cappelli

Janeiro/2011

Um Estudo da Representação Gráfica de Aspectos em BPM utilizando a
ferramenta ARIS Express 2.2

Projeto de Graduação apresentado à
Escola de Informática Aplicada da
Universidade Federal do Estado do Rio
de Janeiro (UNIRIO) para obtenção do
título de Bacharel em Sistemas de
Informação

Nome da Autora:

Clarissa da Silva Correa Romeiro

Nome das Orientadoras:

Flávia Maria Santoro

Claudia Cappelli

AGRADECIMENTOS

Aos meus pais e irmãos, por todo apoio sempre.

Ao meu namorado Daniel pelo amor e companhia e pelos valiosos esclarecimentos para terminar esta etapa.

Aos meus amigos queridos e eternos.

Às minhas orientadoras Flávia e Claudia pela dedicação, paciência e confiança.

A todos da UNIRIO que participaram desta etapa maravilhosa da minha vida.

RESUMO

Os processos de negócio são muito utilizados pelas organizações atualmente. Seus objetivos englobam facilitar a manutenção e promover o entendimento e reuso dos modelos. A Orientação a Aspectos (OA) quando aplicada à modelagem de processos de negócio propõe auxiliar estes objetivos, no entanto, há poucos trabalhos específicos neste contexto. Este trabalho um destes trabalhos para elaborar uma representação gráfica de aspectos em uma ferramenta gratuita, o que daria suporte à modelagem de processos orientada a aspectos.

SUMÁRIO

1.	Introdução	6
1.1.	Motivação	6
1.2.	Proposta de Solução	8
1.3.	Estrutura do Trabalho	8
2.	Conceitos de OA	9
2.1.	Aspecto	13
2.2.	<i>Joinpoint</i>	15
2.3.	<i>Pointcut</i>	16
2.4.	<i>Advice</i>	17
2.5.	Declarações Inter-tipo	18
2.6.	<i>Weaver</i>	19
2.7.	Aspectos e os Níveis de Abstração de Software	19
3.	Trabalhos Relacionados	21
4.	Representação de Aspectos na Ferramenta ARIS Express 2.2	27
4.1.	Trabalho Base	27
4.2.	Sobre a Ferramenta	37
4.2.1.	Instalação	40
4.3.	Exemplo Utilizando o Processo “Gerir Mudanças”	41
4.4.	Sistemática para Representação Visual de Aspectos na Ferramenta	46
4.4.1.	Criar <i>Pool</i> transversal	46
4.4.2.	Criar raias	47
4.4.3.	Criar relacionamento transversal	49
4.4.4.	Alterar propriedades	51
4.4.5.	Representar modelo final	57
4.5.	Estudo de Caso	58
5.	Conclusão	72
	Bibliografia	74

1. INTRODUÇÃO

1.1. Motivação

A modelagem de processos de negócios [SHARP e MCDERMOTT, 2001] compreende uma série de técnicas e métodos para representação dos processos de negócio de uma organização, na maioria das vezes com o objetivo de análise e melhoria dos processos atuais. Estes processos de negócio podem ser organizados em níveis de detalhamento formando uma hierarquia de processos. Para tal em geral são utilizados os conceitos de: (i) *Macro-processo*: abrange diferentes funções de uma organização, representando os processos maiores e complexos que podem ser subdivididos em diversos processos menores, (ii) *Processo*: conjunto de atividades em uma determinada ordem com um objetivo único e bem definido, (iii) *Sub-processo*: subdivisão de um processo que tem como objetivo simplificar um processo extenso e/ou complexo ou destacar objetivos menores dentro do objetivo final do processo, (iv) *Atividade*: uma tarefa simples que pode ser realizada em um passo, muitas vezes não divisível.

Esta forma de hierarquizar os modelos (modularização) em um projeto de modelagem facilita a organização, manutenção, entendimento e reuso dos modelos. No entanto, apenas estes recursos de modularização não são suficientes para identificar determinados elementos que se repetem e estão espalhados pelo modelo de processos e/ou entrelaçados com outros elementos de um mesmo processo. Estas atividades são denominadas elementos transversais [KICZALES *et al.*, 1997] que misturados a elementos básicos dificultam o entendimento dos modelos de processos. Isso também dificulta o reuso e a manutenção dos modelos, pois os elementos transversais estão espalhados sem rastreamento de suas localizações.

O paradigma de Desenvolvimento de Software Orientado a Aspectos (DSOA) [FILMAN *et al.*, 2005] tem tratado estes problemas com uma nova modularização para estes elementos transversais baseado nos conceitos introduzidos por Kiczales na Programação Orientada a Aspectos [KICZALES *et al.*, 1997]. Esta nova modularização é denominada aspecto.

O paradigma de Desenvolvimento de Software Orientado a Aspectos (DSOA) [FILMAN *et al.*, 2005] também aborda uma nova forma para composição de conceitos transversais e conceitos básicos que permite que ele seja um complemento para outros paradigmas, pois seu objetivo é modularizar interesses transversais aos módulos básicos dos outros métodos.

Há diversos trabalhos sobre o paradigma de orientação a aspectos envolvendo diferentes etapas do processo de desenvolvimento de software. Tendo início nas atividades relacionadas à programação [KICZALES *et al.*, 1997], as pesquisas voltaram a atenção para os níveis de abstração mais altos como na fase de arquitetura [BATISTA *et al.*, 2006] e levantamento de requisitos [SILVA, 2006]. A abordagem de modelagem de processos orientada a aspectos possui alguns trabalhos relacionados [THOMPSON e ODGERS, 1999] [CHARFI e MEZINI, 2007] [CAPPELLI *et al.*, 2009a] [CAPPELLI, 2009] [CAPPELLI *et al.*, 2010]. Esses trabalhos descrevem a necessidade de trabalhar aspectos em modelos de processos com o objetivo de uma melhor modularização, reuso e evolução dos modelos de processo. Além disso, os processos de negócio mapeados são uma fonte de informação valiosa para o levantamento de requisitos, e sendo DSOA uma realidade atual, é relevante que a modelagem de processos de negócio também considere o paradigma de aspectos.

Porém as ferramentas para modelagem de processos de negócio atuais não oferecem suporte a modularização em aspectos, o que dificulta a utilização do paradigma no contexto de projetos de modelagem de processos de negócio. Dentre as ferramentas mais utilizadas atualmente para Gestão de Processos de Negócio, ARIS Platform [ARIS Platform] se encontra entre as líderes de mercado [HILL, 2009] e possui uma versão gratuita para utilização.

Dessa forma, a proposta deste trabalho é apresentar uma representação de aspectos em modelos de processos de negócio na ferramenta ARIS Express 2.2 [ARIS Express 2.2].

1.2. Proposta de Solução

A tese de doutorado de Cappelli [2009] apresenta uma abordagem de integração de aspectos com modelos de processos, apoiando-se na proposta de Silva [2006] de integrar aspectos com engenharia de requisitos. Na abordagem de Cappelli [2009] uma Linguagem de Modelagem de Processos Orientada a Aspectos é proposta seguida de exemplos de como aplicá-la. Estes exemplos foram reutilizados para a aplicação da abordagem no ARIS Express 2.2. Essa aplicação é a representação visual de aspectos na ferramenta, que é mostrada passo a passo.

1.3. Estrutura do Trabalho

A seção 2 apresenta os principais conceitos de Orientação a Aspectos como aspecto, *joinpoint*, *pointcut*, *advice*, declarações inter-tipo e *weaver*.

A seção 3 mostra uma revisão dos trabalhos relacionados com o tema estudado.

A seção 4 apresenta a tese de Cappelli [2009] em que se baseia este trabalho, a proposta e um estudo de caso.

A seção 5 apresenta as conclusões do trabalho e propostas de trabalhos futuros.

2. CONCEITOS DE OA

Com a rápida evolução de tecnologias, as organizações têm buscado cada vez mais automatizar tarefas, e desta forma, os sistemas de informação vêm ganhando mais importância. Além disso, com o crescente aumento de dados e informações, esses sistemas têm ficado mais robustos, difíceis de construir e manter [LADDAD, 2002].

Na Orientação a Objetos, os objetos, representados pelas classes, são abstrações de elementos do mundo real, de forma a refletir a realidade facilitando a compreensão do domínio. As classes possuem atributos e operações que compõem as funcionalidades dos requisitos identificados. Mas nem todos estes requisitos podem ser modularizados em uma única classe [KICZALES *et al.*, 1997].

A Programação Orientada a Aspectos (POA), conceito criado por Gregor Kiczales *et al.* em 1997, é um paradigma recente que surgiu para complementar o paradigma de orientação a objetos, modularizando interesses transversais às funcionalidades básicas representadas pelos objetos. Estes interesses transversais ao sistema ficam espalhados e entrelaçados no código, diminuindo a coesão e aumentando o acoplamento, dificultando reuso e manutenção do código.

Código espalhado pode ser entendido como um interesse que se encontra implantado em diversos módulos, tornando difícil o rastreamento e a realização de mudanças (Figura 1). Um código entrelaçado ocorre quando diversos interesses se encontram implantados em um único módulo, sem uma relação forte de coesão (Figura 2).

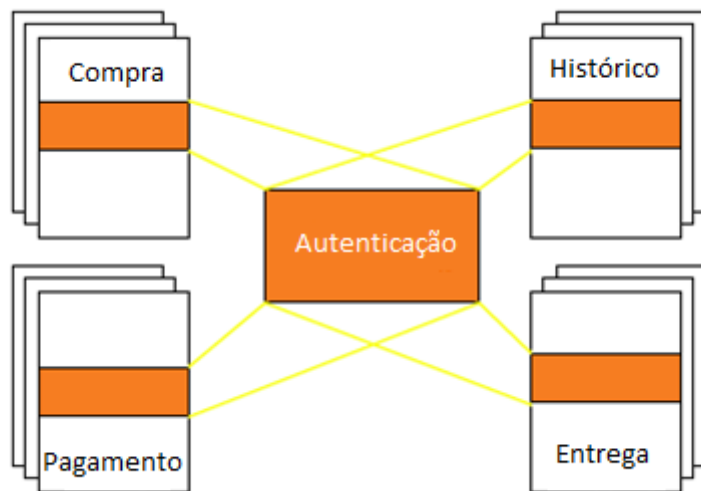


Figura 1 Esta figura ilustra a implementação de um interesse transversal espalhado em múltiplos módulos. [LADDAD, 2002]

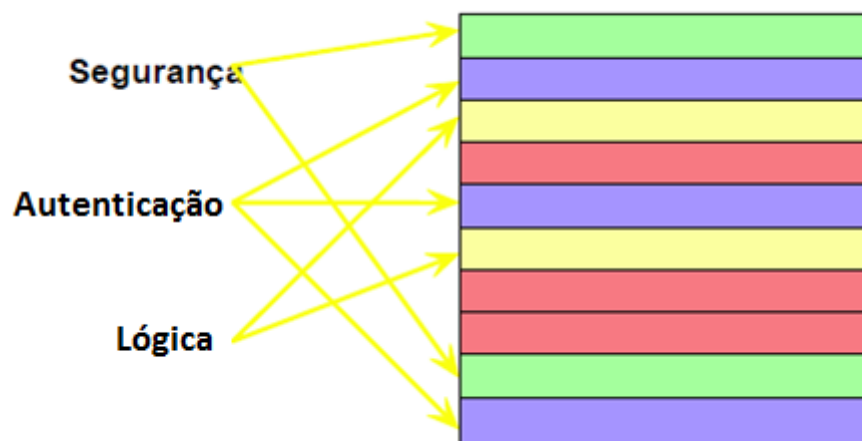


Figura 2 Esta figura ilustra a implementação de múltiplos interesses transversais em um único módulo. [LADDAD, 2002]

Segundo Laddad [2002], as implicações que o espalhamento e o entrelaçamento no design de software acarretam são as listadas a seguir:

- Dificuldade de Rastreamento: A codificação simultânea de vários interesses em um único módulo quebra o caminho que liga o requisito à sua implementação.
- Baixa produtividade: O desenvolvedor acaba despendendo sua atenção em assuntos periféricos ao invés de se concentrar na lógica do negócio.

- Pouco reuso: O problema do código duplicado entre os módulos é um obstáculo para o reuso.
- Dificuldade de refatoração (*refactoring*): Mudança de requisitos implica em modificar múltiplos módulos com o objetivo de mudar um único interesse.

Para exemplificar esses elementos, podemos destacar as atividades relacionadas à segurança que são bastante comuns em diversos processos que envolvem sistemas. Utilizando um modelo de processos hipotético de uma agência de viagens online, atividades de autenticação e *log* de transações se repetem diversas vezes em diferentes processos e em um mesmo processo como, por exemplo, o usuário precisa se autenticar para realizar uma compra; empregados do departamento de transporte e financeiro também precisam realizar autenticação para verificar os pedidos; além disso, *logs* são constantemente armazenados após transações de compra e autorização de cartão de crédito por exemplo.

A linguagem de POA trata o problema de código espalhado e código entrelaçado modularizando os interesses transversais em estruturas definidas por aspectos. A estrutura tradicional de classes da OO não é modificada, a separação por objetos continua sendo realizada, mas estes requisitos que não pertencem a apenas um módulo são colocados em estruturas definidas para isso, permitindo o reuso dessas unidades.

Para encapsular estes interesses transversais em uma única estrutura, é necessário identificar em quais pontos da execução do programa eles devem aparecer. Os *joinpoints* são posições onde o código dos módulos básicos do programa deve interagir com o código dos módulos de aspectos. Pode-se determinar um *pointcut* que define um conjunto de *joinpoints* para que tenham o mesmo tratamento.

Além dos *joinpoints* e *pointcut*, há a estrutura semelhante ao método na programação chamada *advice*, utilizada para declarar o código adicional que deverá ser executado nos *joinpoints*. Os *advices* definem o que deverá ser executado quando os *joinpoints* forem identificados no fluxo de execução do programa.

Por fim, os aspectos são as unidades que representam os interesses transversais, compostos de *joinpoints*, *pointcuts*, *advices* e outros elementos da

linguagem sobre a qual a POA está sendo implementada. Para ocorrer uma correta união da estrutura de aspectos e de objetos, é necessário um componente que processe as duas linguagens e as organize em um único código para execução do programa, é o chamado *weaver*.

O Desenvolvimento de Software Orientado a Aspectos (AOSD) [FILMAN *et al.*, 2005] é a aplicação desta tecnologia em todo o processo de desenvolvimento, ou seja, considerar os aspectos em cada fase de desenvolvimento de um software. Isso permite já identificar e separar os interesses transversais em níveis mais abstratos do desenvolvimento, facilitando o entendimento, assim como acontece na identificação das classes no diagrama de classes da UML.

Durante a leitura desta seção o conceito de Orientação a Aspectos (OA) foi introduzido juntamente com os problemas que ela resolve. Nas próximas subseções serão definidos os conceitos de OA com base na linguagem AspectJ [KICZALES *et al.*, 2001].

AspectJ é uma proposta de código aberto, multi-plataforma e de distribuição gratuita que entende a linguagem Java [JAVA, 1995] implementando as principais estruturas da OA. Essa extensão busca obter compatibilidade com qualquer programa em Java da seguinte forma [KICZALES *et al.*, 2001]:

- Compatibilidade: Programas em Java válidos devem ser programas em AspectJ válidos.
- Compatibilidade de plataforma: Programas em AspectJ válidos devem poder ser executados em uma máquina virtual Java (JVM).
- Compatibilidade de ferramenta: Deve ser possível estender ferramentas existentes para suportar AspectJ.
- Compatibilidade do programador: O programador deve programar em AspectJ como uma extensão de programação em Java.

Existem dois tipos de implementação transversal, a estática e a dinâmica. A primeira complementa as assinaturas estáticas de classes e interfaces em Java [TIRELO *et al.*, 2004] e a segunda define os pontos em que o código dos aspectos será inserido.

Existem algumas linguagens já sendo pesquisadas e criadas como o AspectJ [KICZALES, 2001] e HyperJ [OSSHER e TARR, 2000] para Java, AspectC++ [ASPECTC, 2005] para C e C++. “O poder de expressão e recursos especiais destas linguagens para desenvolvimento de sistemas baseados em aspectos situam-se em um mesmo patamar de equivalência. Entretanto, AspectJ destaca-se por sua grande difusão e alta aderência à proposta inicial do criador da programação orientada por aspectos.” [TIRELO *et al.*, 2004]. Em razão disto, utilizaremos exemplos e definições do AspectJ para detalhar os conceitos apresentados nas próximas subseções.

A subseção 2.7 apresenta os aspectos nos vários níveis de abstração do software, requisitos, arquitetura e programação.

2.1. Aspecto

Aspecto é uma unidade de implementação de um interesse transversal [KICZALES *et al.*, 2001] cuja declaração é muito semelhante à declaração de uma classe. Pode conter declarações de pontos de junção, pontos de corte, comportamentos a adicionar e ainda, outras declarações que são feitas em uma classe como atributos e métodos.

A figura a seguir é um diagrama de classes em UML que será utilizado nas explicações desta e das próximas subseções.

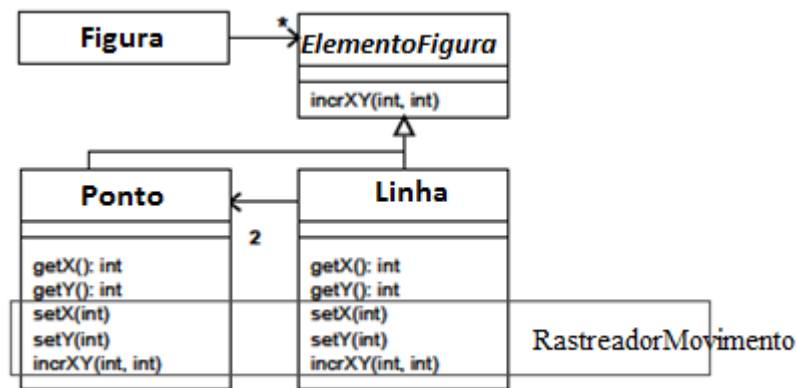


Figura 3 “Descrição em UML de um simples editor de figura. A caixa nomeada “RastreadorMovimento” representa um aspecto transversal às classes Ponto e Linha.”
[KICZALES *et al.*, 2001]

A seguir é apresentada a declaração do aspecto “RastreadorMovimento” que tem como objetivo detectar se um objeto da classe figura se moveu. Este aspecto pode ser utilizado por um mecanismo de atualização de tela para detectar se algum elemento na interface foi modificado desde a última atualização. [KICZALES *et al.*, 2001]

```

aspect RastreadorMovimento {

    static boolean flag = false;
    static boolean testarEResetar() {
        boolean result = flag;
        flag = false;
        return result;
    }
}

```

Como mencionado anteriormente, uma declaração de aspecto pode conter atributos e métodos como uma classe. O aspecto deste exemplo contém uma variável booleana estática denominada *flag* que inicia com o valor *false* e um método com valor de retorno determinado pela *flag*.

Ao contrário da classe, o aspecto não é instanciado [TIRELO *et al.*, 2004], suas declarações são estáticas. Apesar disso, aspectos podem ser abstratos e definir uma hierarquia, mas esses detalhes não fazem parte do escopo do projeto.

2.2. Joinpoint

Como apresentado anteriormente, um *joinpoint* é um ponto bem definido no fluxo de execução de um programa, como por exemplo, a chamada de um método, a instanciação de uma classe, a alteração de um campo. A Figura 4 ilustra uma sequência de *joinpoints* identificados ao ser executada a última linha do código. “As três primeiras linhas de código constroem os objetos representados pelos círculos grandes. Os retângulos representam os métodos. Executando a última linha de código inicia a computação que inclui a sequência de *joinpoints* representados pelos círculos menores numerados.” [KICZALES et al., 2001]

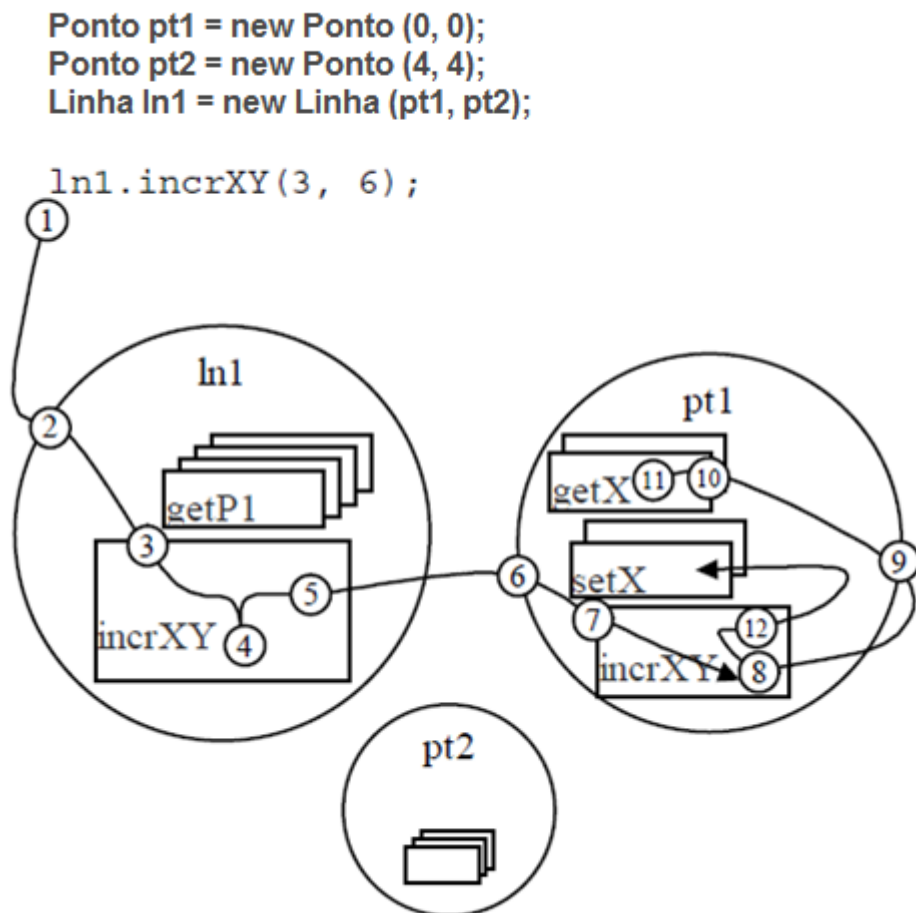


Figura 4 Sequência de *joinpoints* [KICZALES et al., 2001]

- “1) Chamada ao método *incrXY(int, int)* do objeto *ln1*;
- 2) Recepção de *ln1* à chamada ao método *incrXY(int, int)*;
- 3) Início da execução do método *incrXY(int, int)* definido em *ln1*;

4) *Leitura do campo P1 de ln1;*
 5) *Chamada ao método incrXY(int, int) do objeto pt1;*
 ...
 8) *Chamada ao método getX() do objeto pt1;*
 ...
 11) *Leitura do campo X de pt1;*
Controle retorna pelos pontos de junção 11, 10, 9 e 8;
 12) *Chamada ao método setX(int) do objeto pt1;*
 ... e assim por diante, até retornar pelos joinpoints 3, 2 e 1”
 [KICZALES et al., 2001]

Para encapsular os interesses transversais em uma única estrutura, é necessário identificar em quais pontos da execução do programa eles devem aparecer. Os *joinpoints* são pontos bem definidos no fluxo de execução de um programa, onde o código dos módulos básicos do programa deve interagir com o código dos módulos de aspectos, como por exemplo, a chamada de um método, a instanciação de uma classe, a alteração de um campo.

2.3. *Pointcut*

Um *pointcut* é composto por *joinpoints* e os valores de contexto no fluxo de execução, possuindo um nome e o código que deverá ser inserido em cada *joinpoint*. Existem os *pointcuts* primitivos, ou seja, já definidos na linguagem AspectJ e podem-se criar novos *pointcuts* com um nome específico ou anônimo.

Como exemplo, o *pointcut* primitivo abaixo, possui o nome “recepções” e se refere a todas as recepções de chamadas de método cuja assinatura é `void Ponto.setX(int)` em Java [JAVA, 1995].

```
recepções(void Ponto.setX(int))
```

Pointcuts também podem ser combinados por operadores lógicos OR, AND e NOT como em Java como a seguir:

```
recepções(void Ponto.setX(int)) ||
recepções(void Ponto.setY(int))
```


Podemos designar um *pointcut* para o exemplo do aspecto “RastreadorMovimento”. Denominado “movimento”, este *pointcut* detecta quando um elemento figura recebe uma chamada a um método que o modifica graficamente.

```
aspect RastreadorMovimento {  
  
    static boolean flag = false;  
    static boolean testarEResetar() {  
        boolean result = flag;  
        flag = false;  
        return result;  
    }  
    pointcut movimento():  
        receções(void FiguraElemento.incrXY(int, int)) ||  
        receções(void Linha.setP1(Ponto)) ||  
        receções(void Linha.setP2(Ponto)) ||  
        receções(void Ponto.setX(int)) ||  
        receções(void Ponto.setY(int));  
}
```

2.4. Advice

Além dos *joinpoints* e *pointcuts*, há a estrutura semelhante ao método chamada *advice*, utilizada para declarar o código adicional que deverá ser executado nos *joinpoints*. Os *advices* definem o que deverá ser executado quando os *joinpoints* forem identificados no fluxo de execução do programa.

Advice é o trecho de código a ser inserido no fluxo de execução do programa quando identificado um *joinpoint* definido para ele em um *pointcut*. Os *advices* são declarados na estrutura dos aspectos, podendo se referir a um ou mais *pointcuts*. Há três tipos de *advice*, o executado antes do *joinpoint* (*before*), o executado após o *joinpoint* (*after*) e o executado antes e/ou depois do *joinpoint* (*around*). Existem tipos especiais do tipo *after* que são o *after returning* e *after throwing*, especificando as duas formas de se finalizar um *joinpoint*.

Para declarar um *advice*, deve-se associá-lo a um *pointcut*, especificando a forma como ele deve ser executado, *before*, *after* ou *around*. A declaração do código a ser executado é semelhante à declaração de método em Java.

```
aspect RastreadorMovimento {
```

```

static boolean flag = false;
static boolean testarEResetar() {
    boolean result = flag;
    flag = false;
    return result;
}
pointcut movimento():
    receções(void FiguraElemento.incrXY(int, int)) ||
    receções(void Linha.setP1(Ponto)) ||
    receções(void Linha.setP2(Ponto)) ||
    receções(void Ponto.setX(int)) ||
    receções(void Ponto.setY(int));

after(): movimento() {
    flag = true;
}
}

```

No exemplo, é declarado um *advice* para ser executado após o *pointcut* denominado “movimento”. Agora o objetivo do aspecto de exemplo está explicitado, sempre que há um movimento em um elemento figura, o atributo *flag* é verdadeiro, indicando que houve uma mudança.

2.5. Declarações Inter-tipo

Declarações inter-tipo é a forma como o aspecto adiciona métodos e atributos em uma classe ou em um objeto existente. Utilizando os exemplos de Kiczales *et al.* [2001] apresentados nas subseções anteriores, adicionamos dentro do aspecto “RastreadorMovimento” a declaração abaixo:

```

aspect RastreadorMovimento {
    private void Ponto.aspectMethod(int x){
        ...
    };
    ...
}

```

Isto adiciona o método “aspectMethod(intx)” na classe “Ponto” implicitamente e de forma que apenas o aspecto “RastreadorMovimento” terá acesso devido à declaração de acesso “private”.

2.6. *Weaver*

Após a implementação das funcionalidades básicas e dos aspectos transversais, é necessário um compilador que entrelace os dois códigos formando um único código de execução. Este compilador é chamado *weaver*, responsável por identificar os *joinpoints* e inserir o código necessário para que tanto aspectos quanto os módulos básicos executem de forma coordenada e correta.

A figura a seguir ilustra o funcionamento do *weaver*.

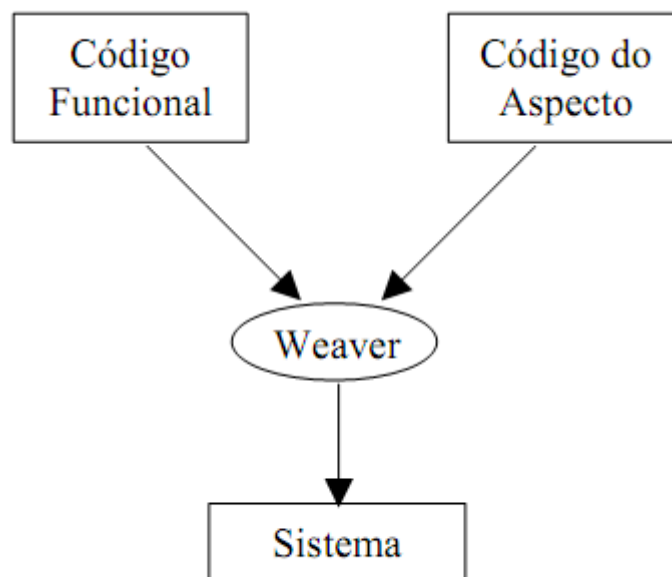


Figura 5 Ilustração do funcionamento de um Weaver. [FERNANDES e BATISTA 2004]

2.7. Aspectos e os Níveis de Abstração de Software

Após a introdução do conceito de aspectos por [KICZALES *et al.*, 1997], diversos trabalhos foram realizados em cima de estudos sobre aspectos nos diversos níveis de abstração de software. Baseando-se no conceito de características

transversais, aspectos em níveis mais abstratos são considerados aspectos candidatos que poderão ou não ser implementados como aspectos na fase de codificação.

A Engenharia de Requisitos Orientada a Aspectos (EROA) representa a identificação, análise e tratamento de requisitos transversais durante a engenharia de requisitos. Com isso o modelo de requisitos passa a ter um padrão modular adequado, evitando que em características transversais fiquem escondidas, o que facilitará a implementação em ambientes orientados a aspectos. Separando conceitos transversais no levantamento de requisitos, facilita o rastreamento entre documentos de requisitos e arquitetura e o código, principalmente quando se utiliza OA [SILVA, 2006].

A arquitetura de software compreende a análise e definição dos componentes que farão parte do software, seus relacionamentos e interfaces entre si e externamente. Esta área tem sido estudada para integração com OA principalmente as linguagens de descrição arquitetural (ADLs). A partir das ADLs, foram criadas extensões das abstrações existentes da linguagem para tratar aspectos já nessa fase do processo de desenvolvimento de software [BATISTA *et al.* 2006].

3. TRABALHOS RELACIONADOS

Diversos trabalhos relacionados foram estudados. Dentre eles, grande parte refere-se a aspectos na etapa da modelagem de requisitos.

Thompson *et al.* [1999] apresenta uma proposta de utilização das abstrações de aspectos para customização de processos. O artigo não utiliza processos em linguagens BPM e sim, implementações de processos em AspectJ.

Rashid *et al.* [2003] fala em seu artigo da importância de se identificar e tratar aspectos já na fase de análise de requisitos, separando requisitos aspectuais, requisitos não-aspectuais e regras de composição. Os autores afirmam que essa modularização no início do processo de engenharia de software permite mudanças em aspectos mais cedo, facilitando a tomada de decisão pelos *stakeholders* e a negociação. A definição de requisitos transversais também facilita o mapeamento e a determinação dos componentes aspectuais nos demais estágios do processo de desenvolvimento.

O artigo de Batista *et al.* [2006] faz uma reflexão sobre sete questões importantes que surgem quando estudamos a integração de características transversais e linguagens de descrição (ADL) arquitetural. O objetivo é apresentar o ponto de vista dos autores sobre se a modularização desses aspectos transversais exigem extensões das ADLs existentes.

O desenvolvimento se concentra em dois capítulos, o capítulo 2 que faz uma revisão das ADLs existentes, inclusive as que adotaram a abordagem orientada a aspectos; e o capítulo 3 apresenta as sete questões sobre aspectos e ADLs. A seguir, um resumo das sete questões discutidas no artigo:

Questão 1: Quais elementos em ADL podem ter características transversais - Discute sobre quais elementos da descrição arquitetural (componentes, conectores, configurações e interfaces) podem ser afetados por características transversais.

Questão 2: Composição - Discute a conexão entre dois componentes básicos e a conexão entre um componente básico e um componente transversal e como essa segunda deve ser representada em uma especificação arquitetural. A proposta do

artigo é estender a interface dos conectores de forma a definir papéis (*roles*) diferentes para a conexão com componentes aspectuais e não aspectuais.

Questão 3: Quantificação - Trata do problema de querer referenciar vários pontos de junção em uma declaração, sendo necessário um mecanismo de quantificação da ADL que possa unificar os pontos de junção afetados por um aspecto sem a necessidade de citar um a um.

Questão 4: Interfaces de aspectos - Discute se as questões 2 e 3 são suficientes para representar os aspectos ou se as interfaces dos componentes aspectuais também precisam de informações extras para suportar essas conexões.

Questão 5: Exposição de pontos de junção - Discute se as interfaces dos componentes afetados por aspectos necessitam ser estendidas para explicitar os *joinpoints* que permitem a conexão com componentes aspectuais.

Questão 6: Mudança em interfaces - Trata da adoção do conceito de *inter-type declarations* (declarações inter-tipo) no nível da especificação arquitetural, se é necessário especificar no nível de arquitetura atributos e serviços que um aspecto pode introduzir nos elementos básicos do sistema.

Questão 7: Como representar aspectos - Discute a questão da abstração de um aspecto, se é necessária uma nova abstração pra representar um aspecto transversal ou se este aspecto pode ser representado pelas estruturas existentes para abstração dos componentes básicos na ADL em questão.

Como conclusão, o artigo produz uma tabela comparativa entre linguagens existentes e esboça um modelo de que não introduz complexidade na especificação da arquitetura, utilizando as mesmas abstrações utilizadas na descrição arquitetural e apenas estendendo alguns elementos para tratar os aspectos transversais. O trabalho de graduação de Soares [2008] utiliza este artigo para traçar ligação os interesses transversais na modelagem de processos de negócio e o conceito da orientação a aspecto com base nos sete pontos críticos do artigo.

Na tese de doutorado de Silva [2006], foi apresentando um modelo conceitual que define um novo tipo de relacionamento, denominado relacionamento transversal, que poderá ser utilizado por diversas linguagens de modelagem de requisitos para

envolver os problemas de espalhamento e entrelaçamento de características transversais. Para a representação destas características transversais, foi criada uma linguagem de modelagem de requisitos orientada a aspectos (LMROA) que pode ser em outras linguagens de modelagem de requisitos existentes para estendê-las.

Vanderfeesten *et al.* [2006] analisa a importância da elaboração de “bons” modelos de processos, utilizando um framework cognitivo para identificar e entender a relação entre um conjunto de elementos de um modelo e o processo mental de conhecimento do mesmo. Para isso, eles definem uma métrica de inter-conectividade que mede a força da ligação entre os elementos do processo, baseando-se na hipótese de que modelos são mais facilmente entendidos e contêm menor número de erros com alta inter-conectividade.

Em sua tese Charfi e Mezini [2007] apresentam um estudo de modularização em workflows, mais especificamente, sobre aspectos transversais e mudanças em fluxos de trabalho. O objetivo principal é a elaboração de uma linguagem de workflow que apóie a modularização dos aspectos transversais e mudanças. O trabalho identifica limitações nas linguagens de workflow correntes para modularização de aspectos transversais e mudanças e propõe uma linguagem de workflow orientada a aspectos.

Este trabalho apresenta uma solução interessante para o problema dos aspectos transversais em processos. Nos exemplos de linguagens gráficas independentes de outras linguagens, os conceitos da OA foram bem identificados e definidos, podendo ser adaptado para processos de negócio.

Interessante observar que nos exemplos, outros elementos do processo, além das atividades, como participantes, sistemas e dados de entrada e saída, são considerados na separação dos conceitos transversais. Estes outros elementos também são objeto de estudo na identificação e elaboração de proposta para aspectos transversais em processos de negócio. No entanto, em um workflow, eles não são tratados separadamente das atividades.

A figura abaixo ilustra os aspectos espalhados entre os processos de workflow e os processos entrelaçados de diferentes aspectos. Ao lado os aspectos se apresentam modularizados em estruturas próprias.

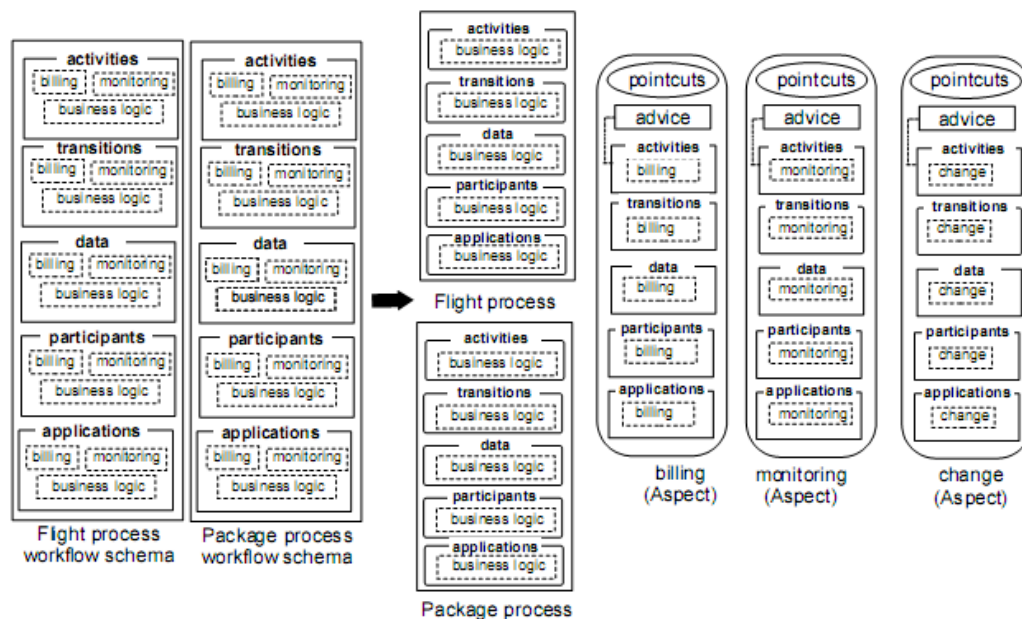


Figura 6 Exemplo de processos de workflow e aspectos espalhados e entrelaçados [Charfi e MEZINI, 2007]

Cabe aqui introduzir conceitos relacionados à OA no contexto de processos de workflow. Pontos de junção são pontos determinados no fluxo de execução do processo, onde um determinado aspecto deve ser contemplado. Pontos de corte é um agrupamento de pontos de junção que devem ser tratados de uma mesma forma. Por fim, comportamento a adicionar nesse contexto representa as atividades relacionadas ao aspecto e em que momento elas devem ser executadas no fluxo.

No exemplo da Figura 7, o aspecto de coleta de dados para pagamento é invocado pelo ponto de junção da atividade “Flight search Berlin Air” e o adendo representado pela atividade “Increment counter” deve ser executado após aquela atividade.

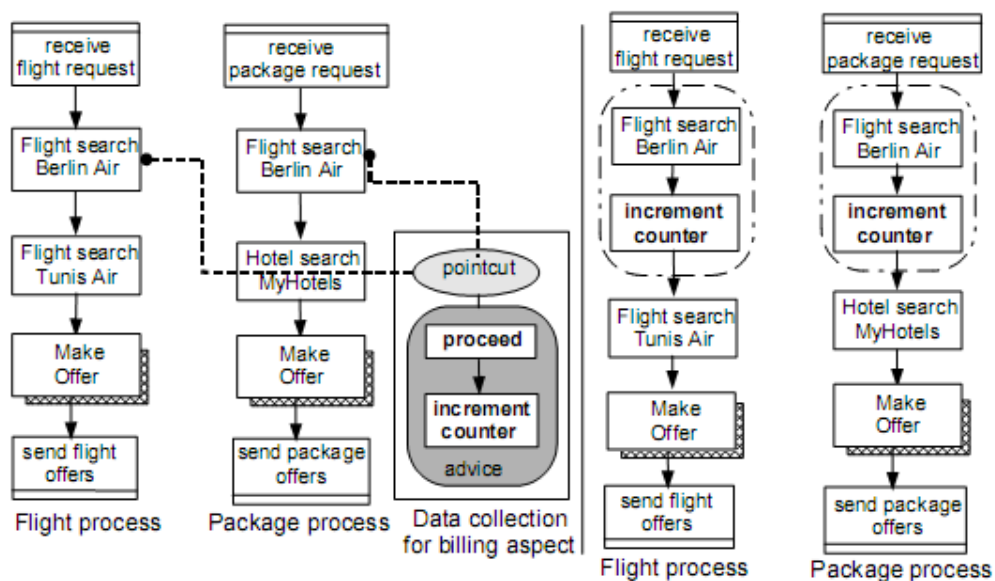


Figura 7 Exemplo do aspecto de coleta de dados [Charfi e MEZINI, 2007]

Novamente na Figura 7, é identificado um ponto de junção para o aspecto de medição de tempo de execução de atividade. Neste exemplo a atividade “Flight search Berlin Air” é executada ainda na execução do adendo do aspecto.

Park *et al.* [2007] foca na modularização de regras de negócio como características transversais. Os autores definem um framework de Programação Orientada a Aspectos (AOP) baseado em regras para separação e execução de regras de negócio. Além disso, um método é definido para representação e análise de regras de negócio em um *rule engine* separado do processo base. Apesar de a proposta tratar modularização em implementações de processos de negócio, o artigo contribui para este trabalho destacando as regras de negócio como um elemento de aspectualização.

Wada *et al.* [2008] propõe uma linguagem orientada a aspectos para separar propriedades funcionais e não-funcionais de processos de negócio apoiados por uma Arquitetura Orientada a Serviços (SOA). Cada aspecto encapsula propriedades não-funcionais que permeiam múltiplos serviços. A linguagem permite a separação dessas propriedades, diminuindo o custo e o trabalho de desenvolvimento e manutenção de serviços.

O artigo de Cappelli *et al.* [2009a] apresenta uma abordagem orientada a aspectos que objetiva modularizar especificamente a modelagem de processo de

negócio. Essa abordagem inclui uma Linguagem de Modelagem de Processo de Negócios Orientada a Aspectos (AOPML), independente de qualquer linguagem específica de processo de negócio. Adicionalmente, o trabalho mostra a aplicabilidade da proposta utilizando uma notação de modelos de processo de negócio (BPMN [BPMN, 1997]) em um estudo de caso.

[CAPPELLI *et al.*, 2010] explora o assunto de integração de OA com processos de negócio iniciando com um extensa revisão bibliográfica e reutilizando as sete questões apresentadas por Batista *et al* [2006], para fazer a análise de como os conceitos de OA podem ser integrados em modelagem de processos de negócio. Além disso o trabalho apresenta uma implementação em ferramenta de código aberto para modelagem de processos com OA. Esta ferramenta é disponibilizada como CrossOryx Editor [CrossOryx] por estender a ferramenta Oryx Editor [Oryx].

Os trabalhos analisados apresentam um estudo sobre características transversais em outros níveis de desenvolvimento de software, antecipando a identificação e o tratamento de aspectos já nas fases de análise. Os exemplos de trabalho de Cappelli [2006] servirão como base para a implementação no ARIS Express 2.2 da LMPOA apresentada na tese de doutorado.

É importante destacar que qualquer proposta de actualização em processos de negócio deve trazer pouca ou nenhuma complexidade tanto para analistas e modeladores de processos quanto para pessoas do negócio que já conhecem a modelagem tradicional. Portanto, isto será refletido na aplicação da linguagem na ferramenta escolhida.

4. REPRESENTAÇÃO DE ASPECTOS NA FERRAMENTA ARIS EXPRESS 2.2

4.1. Trabalho Base

A proposta é baseada na tese de doutorado de Cappelli [2009] que propõe utilizar processos de negócio para inserção de transparência nas organizações. Transparência pode ser definida no contexto organizacional como “Condição de abertura total dos canais de comunicação de uma organização (empresa, instituição, governo) para o público, sem qualquer cerceamento de informações” [CAPPELLI, 2009 *apud* BARBOSA e RABAÇA, 2002]. Para introduzir transparência nos modelos de processos, uma abordagem orientada a aspectos é utilizada.

Cappelli afirma que a presença ou a ausência de “transparência não impacta a essência do que o processo organizacional executa” e baseando-se no conceito de interesse transversal em processos de negócio que pode ser separado e reutilizado no modelo de processo, a autora propõe transparência como característica transversal.

O ponto de partida da abordagem é a tese de doutorado de Silva [2006] descrita anteriormente, que define um metamodelo para integração de características transversais. Esse metamodelo foi adaptado para o contexto de modelos de processos organizacionais seguindo a estratégia de *Separar, Compor e Visualizar* definida por Silva [2006]. Desta forma, os aspectos são “identificados e separados, depois combinados novamente aos processos e por fim visualizados da forma mais interessante para a organização” [CAPPELLI, 2009]. A estratégia é ilustrada abaixo:

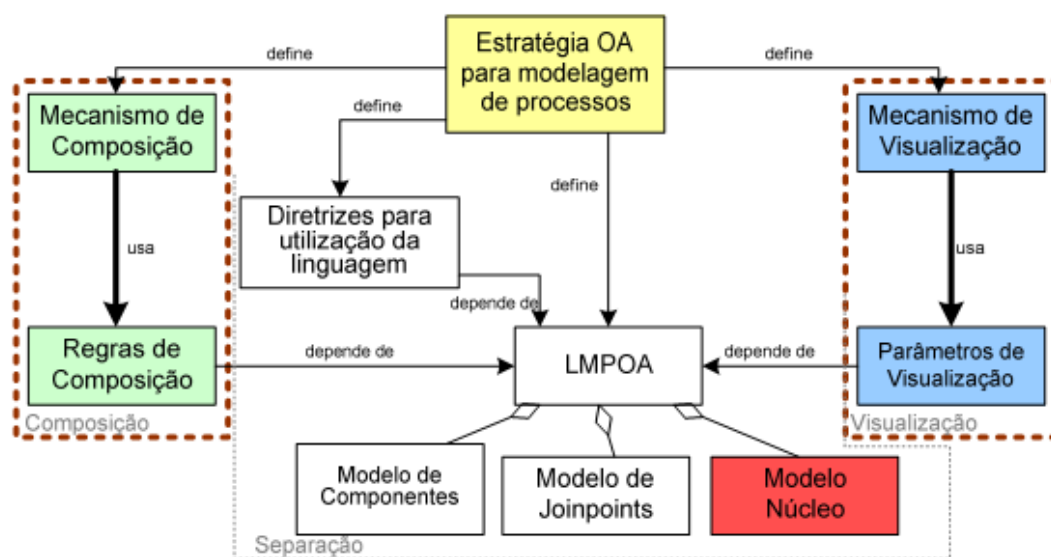


Figura 8 Componentes usados para integrar relacionamentos transversais em modelos de processos [CAPPELLI. 2009]. Adaptado de Silva [2006].

A Linguagem de Modelagem de Processos Orientada a Aspectos (LMPOA) pode ser instanciada para qualquer linguagem BPM e é composta pelos elementos básicos de uma linguagem BPM (*Modelo de Componentes*), o que envolve geralmente processos, atividades, eventos, atores, sistemas e informações; pelo relacionamento transversal (*Modelo Núcleo*); e pelos elementos da linguagem BPM que podem ser afetados pelo relacionamento transversal (*Modelo de Joinpoints*).

O relacionamento transversal é um novo tipo de relacionamento entre elementos transversais e elementos do processo base, podendo compará-lo ao aspecto em AspectJ. É composto pelo elemento de origem (elemento transversal); elementos de destino (*pointcut*); e pelo *advice* que determina como o elemento de origem afeta os elementos de destino. Abaixo a sintaxe do relacionamento transversal traduzida de Cappelli [2009].

```

1.<interesseTransversal> := Aspecto <nomeOrigem>
2.<tipoOrigem> := Tipo <tipoOrigem>
3.<responsávelOrigem> := Responsável <responsávelOrigem>
4.<declaração transversalidade> := Declaração de
  Transversalidade {
5.   <relacionamentoTransversal> := Relacionamento
  <legendaRelTransversal>
6.   <pointcut> := Onde: {<expressãoRegular> :=
7.     <tipoAdvice> := after|before|around
8.     <valor>; "<joinpoint>" |
9.     NOT <valor>; "<joinpoint>" |
  
```

```

10.      AND <valor>; "<joinpoint>" |
11.      OR <valor>; "<joinpoint>" }
12.  <primitiva> := Ação include
    <legendaRelTransversal>}

```

Na abordagem de Cappelli a semântica dos elementos *pointcut* e *advice* da sintaxe foram adaptados dos conceitos já existentes nos níveis de implementação e arquitetura para o contexto da modelagem de processos de negócio. *Pointcut* seleciona os elementos que serão afetados pelo aspecto, utilizando operadores para agrupamento de *joinpoints* (NOT, AND e OR). *Advice* define como os elementos transversais afetam o *pointcut*, utilizando os tipos de *advice* (*after*, *before* e *around*) e a primitiva *include*. Apenas essa é utilizada para a modelagem de processos.

Na definição das diretrizes para separação de características transversais a autora considera que qualquer característica pode ou não "estar" transversal, de forma que todas são modeladas da mesma maneira e a decisão de separá-las ou não se baseia nos seguintes pontos:

- Ela está repetida muitas vezes no modelo;
- Ela pode ou não estar presente no processo, sem impedir a realização de seu objetivo global;
- Ela não faz parte do contexto do processo, é apenas uma qualidade do modelo;
- Ela pode ser reutilizada em outros domínios;
- Ela é muito complexa, estando associada a muitos elementos do processo;
- Ela não se repete no modelo, mas se repete em outros modelos;
- Ela é independente das demais, representando um conceito bem delimitado." [CAPPELLI, 2009]

Esses pontos são recomendações e as características devem ser avaliadas durante o processo de modelagem, pois a separação depende do domínio do negócio.

É necessário que as características transversais separadas do modelo possam ser integradas novamente e para isso definiram-se regras de composição. Estas regras são apresentadas abaixo e definem como os operandos (elementos afetados pelas características transversais) são agrupados e onde devem ser inseridos os elementos transversais, utilizando o tipo *advice*.

Regras de Composição	
Operador	Semântica
<i>AND</i>	Afeta todos os operandos
<i>OR</i>	Afeta apenas um dos operandos
<i>NOT</i>	Exclui o operando do conjunto de pontos a serem afetados
Tipo Advice	Semântica
<i>Before</i>	Inclui os elementos definidos no corpo do <i>advice</i> ANTES do <i>pointcut</i>
<i>After</i>	Inclui os elementos definidos no corpo do <i>advice</i> DEPOIS do <i>pointcut</i>
<i>Around</i>	Inclui os elementos definidos no corpo do <i>advice</i> como filho do <i>pointcut</i>

Figura 9 Regras de composição do relacionamento transversal [CAPPELLI, 2009]

A etapa de visualização objetiva definir diferentes visões do modelo de processos. Essas visões atendem diferentes necessidades da gestão de processos de negócio, pois apresentam diferentes faces do modelo com os pontos mais importantes para solucionar o problema a ser tratado, e ocultando informações em excesso.

Parâmetros de Visualização	
Parâmetros	Descrição
Modelo(s)	Define o(s) modelo(s) de processo que se quer visualizar
Componente(s) do(s) Modelo(s)	Define os componentes da linguagem que se quer visualizar nos modelos definidos (pointcuts ou não)
Características Transversais	Define as características transversais que se quer visualizar (uma ou mais de uma)
Apresentação Relacionamento Transversal	Define como se quer visualizar os relacionamentos (ex: formato texto ou formato gráfico)

Figura 10 Parâmetros de visualização [CAPPELLI, 2009]

Cappelli utiliza um exemplo de processo dentro do contexto de desenvolvimento de software modelado em BPMN [BPMN, 1997] para ilustrar a abordagem proposta. Neste processo (Figura 11) as atividades “Obter Formulário Solicitação Mudança”, “Enviar Solicitação Mudança”, o documento “Formulário Solicitação Mudanças” (“Formulário padrão Requisição” no diagrama) e os dados “Dados Requisição” se repetem. Esses elementos estão destacados na Figura 12

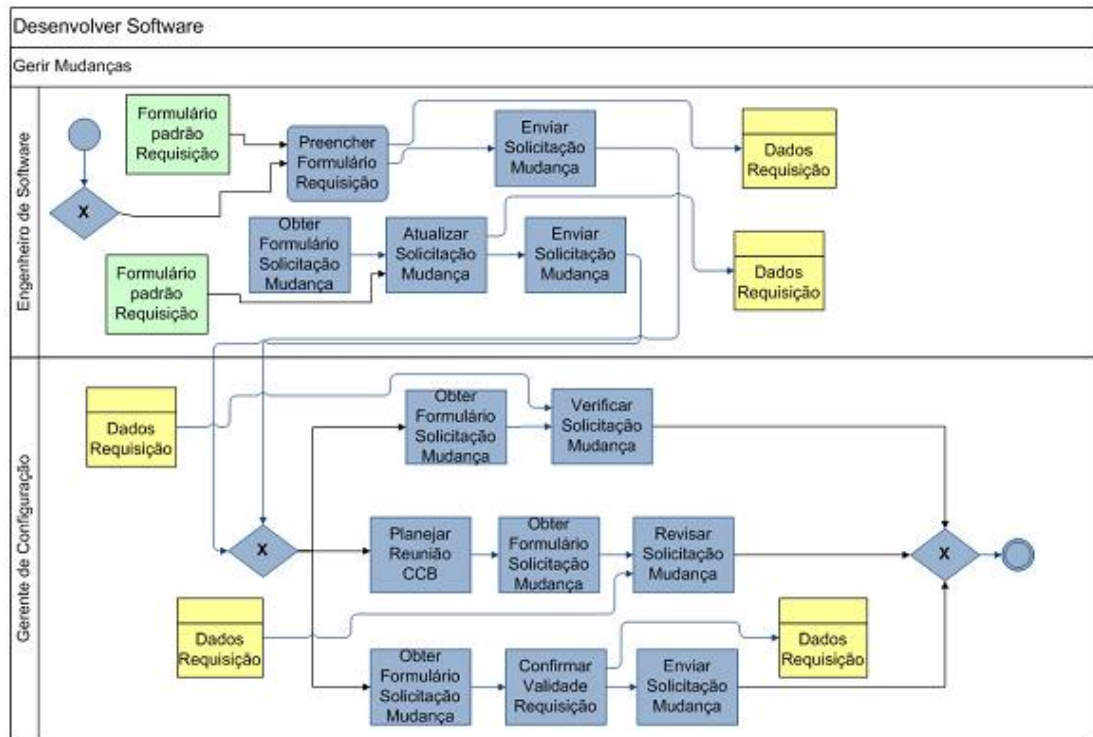


Figura 11 Processo “Gerir Mudanças”. Abordagem tradicional de modelagem [CAPPELLI, 2009]

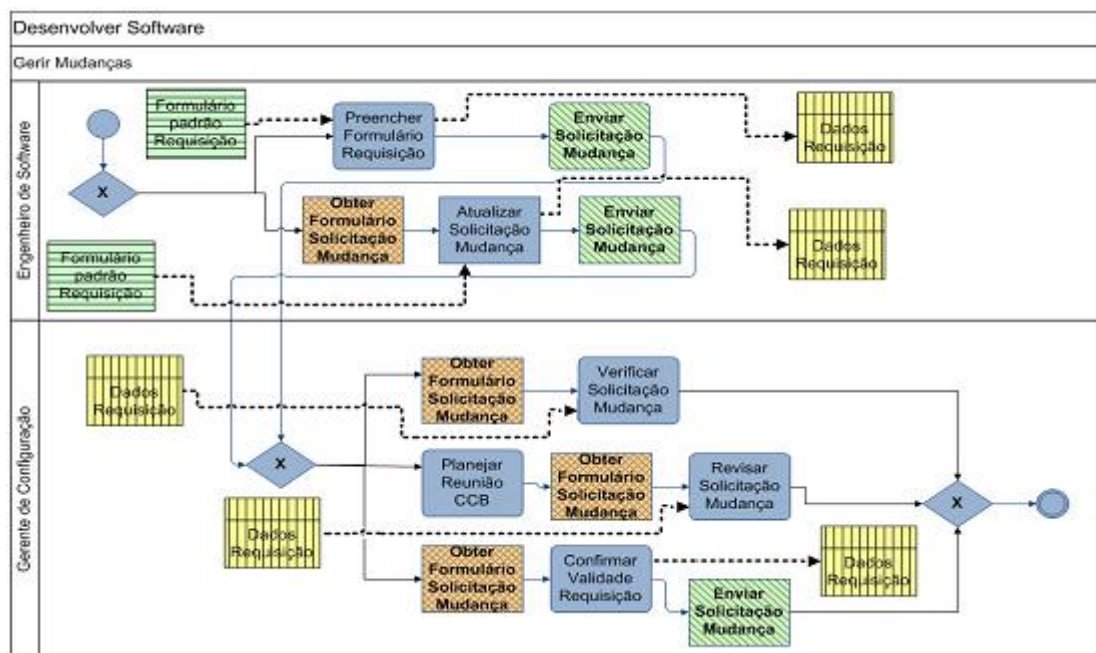


Figura 12 Processo “Gerir Mudanças”. Características transversais e seus relacionamentos com demais elementos [CAPPELLI, 2009]

Para cada elemento foi criado um relacionamento transversal, apresentados em Figura 13, Figura 14, Figura 15 e Figura 16. As figuras mostram o nome do

aspecto a ser inserido no processo (nome do elemento repetido), o tipo do elemento, o nome do relacionamento, onde o elemento transversal será inserido (*before*, *around* ou *after*) e o *advice*, que contém a primitiva *include*, operadores e operandos. Os responsáveis dos aspectos serão representados por atores nas raias ortogonais.

Aspecto: Obter Formulário Solicitação Mudança
Tipo: Activity
Declaração Transversalidade:
Relacionamento: ObtForSolMud
Onde: Before [AllActivities [Atualizar Solicitação Mudança] AND
[Verificar Solicitação Mudança] AND
[Revisar Solicitação Mudança] AND
[Confirmar Validade Requisição]]
Ação: Include [Obter Formulário Solicitação Mudança]

Figura 13 Exemplo de aspecto – Obter Formulário Solicitação Mudança [CAPPELLI, 2009]

Aspecto: Enviar Solicitação Mudança
Tipo: Activity
Declaração Transversalidade:
Relacionamento: EnvSolMud
Onde: After [AllActivities [Preencher Formulário Requisição] AND
[Atualizar Solicitação Mudança] | AND
[Confirmar Validade Requisição]]
Ação: Include [Enviar Solicitação Mudança]

Figura 14 Exemplo de aspecto – Enviar Solicitação de Mudança [CAPPELLI, 2009]

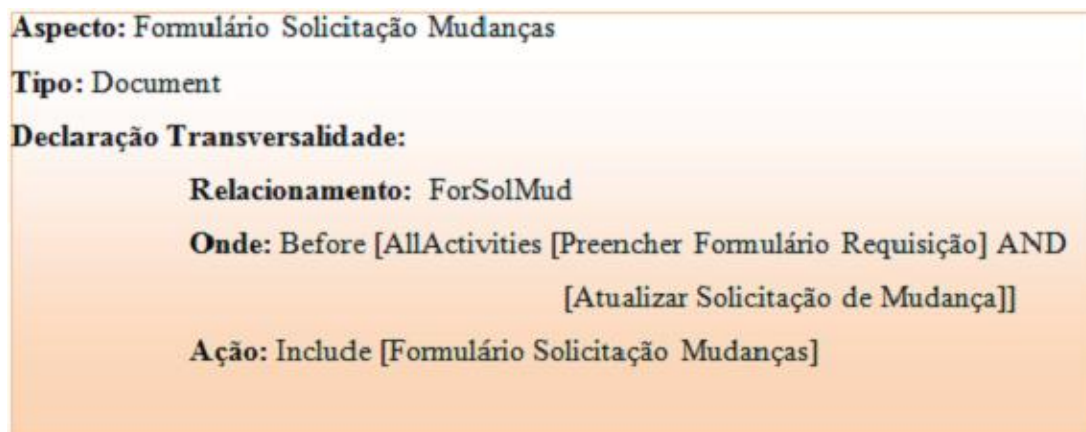


Figura 15 Exemplo de aspecto – Formulário Solicitação Mudanças [CAPPELLI, 2009]

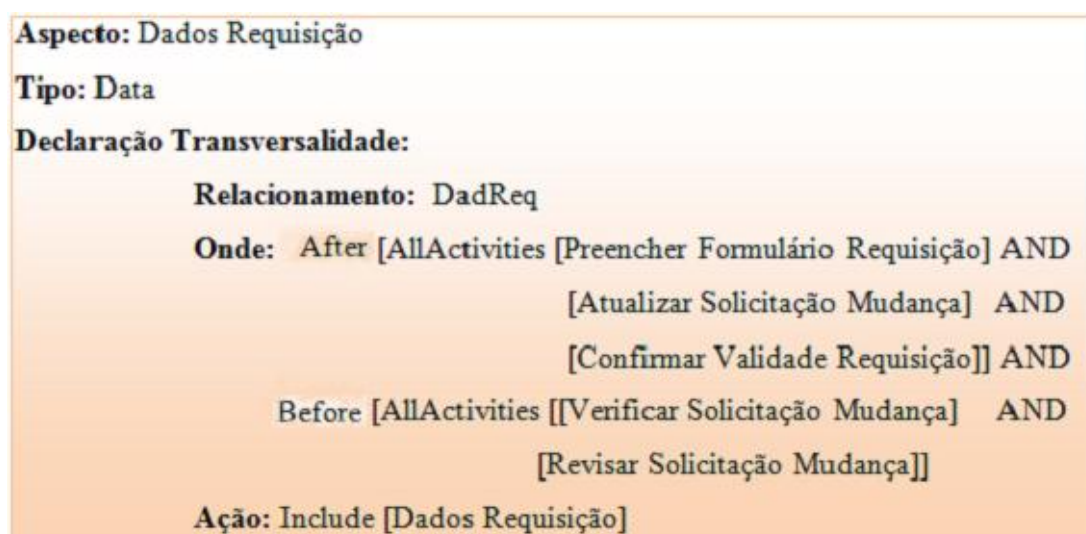


Figura 16 Exemplo de aspecto – Dados Requisição [CAPPELLI, 2009]

Após a identificação dos elementos transversais, esses são retirados do modelo, inclusive suas repetições, e são colocados em raias ortogonais, sem repetição, junto com o nome do responsável pelo aspecto. Estas raias são ortogonais ao processo base justamente por se tratarem de características transversais (ou ortogonais) ao modelo e desta forma, o aspecto visual fica alinhado ao conceito.

Utilizando a semântica dos exemplos acima, um modelo é construído baseado nas regras de composição (Figura 9) e relacionamentos transversais representados por linhas tracejadas. A figura abaixo ilustra o modelo composto de características transversais em raias ortogonais e do processo base.

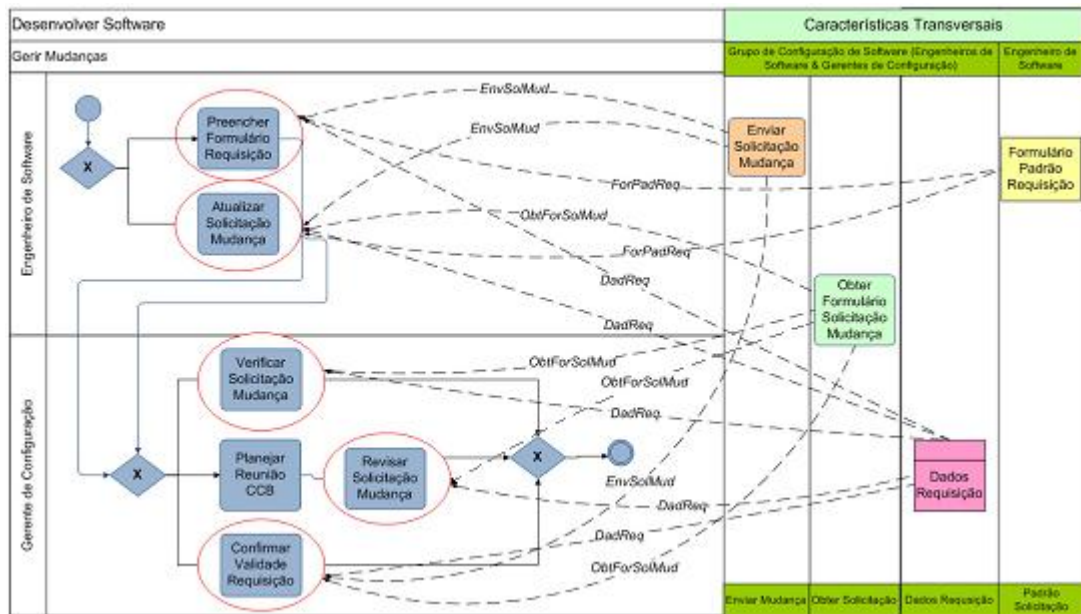


Figura 17 Composição das características transversais [Cappelli, 2009 *apud* CAPPELLI *et al.*, 2009b]

O mesmo modelo pode ser representado também em formato texto, como a figura a seguir. Neste caso não há necessidade da representação gráfica do relacionamento transversal já que a descrição está nos textos.

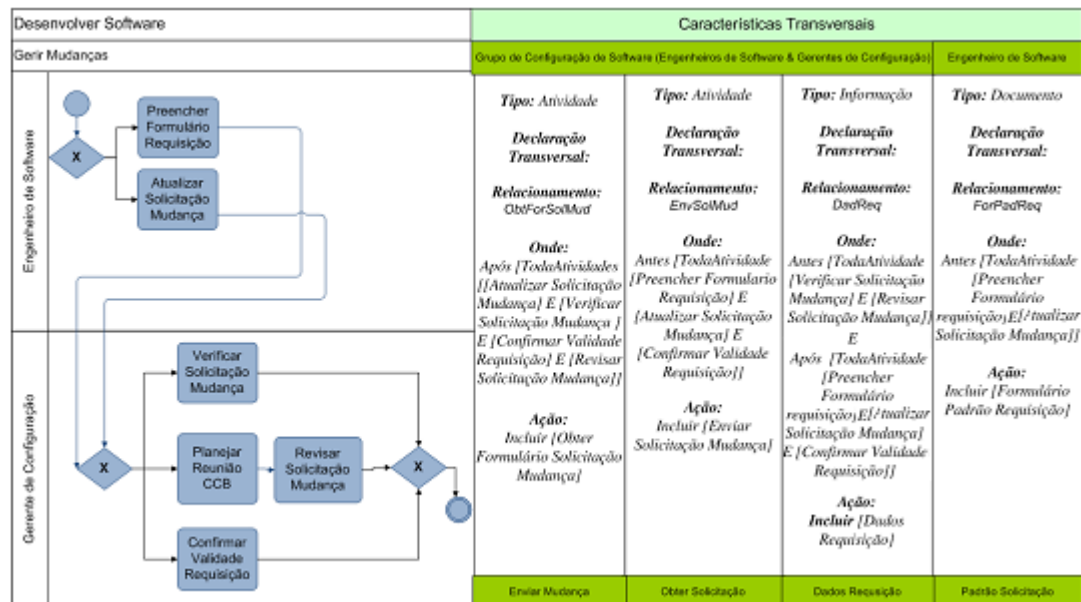


Figura 18 Composição das características transversais – Formato Texto Cappelli, 2009 *apud* CAPPELLI *et al.*, 2009b]

Por último, alguns exemplos de visões do modelo são apresentados. Pode ser de interesse para a organização visualizar quais atividades utilizam o mesmo

documento ou os mesmos dados(Figura 19 e Figura 20). Ou um processo que contenha apenas atividades (Figura 21). A Figura 22 apresenta outra visão, a de atividades básicas ou principais.

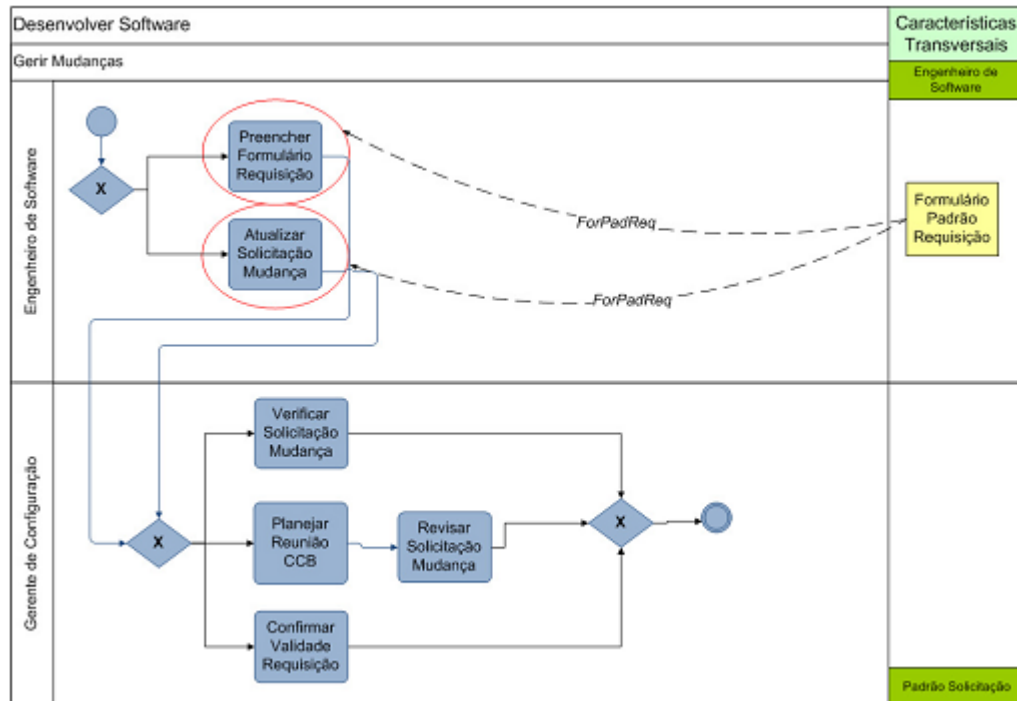


Figura 19 Visão de Documentos – As atividades que utilizam o documento [CAPPELLI, 2009]

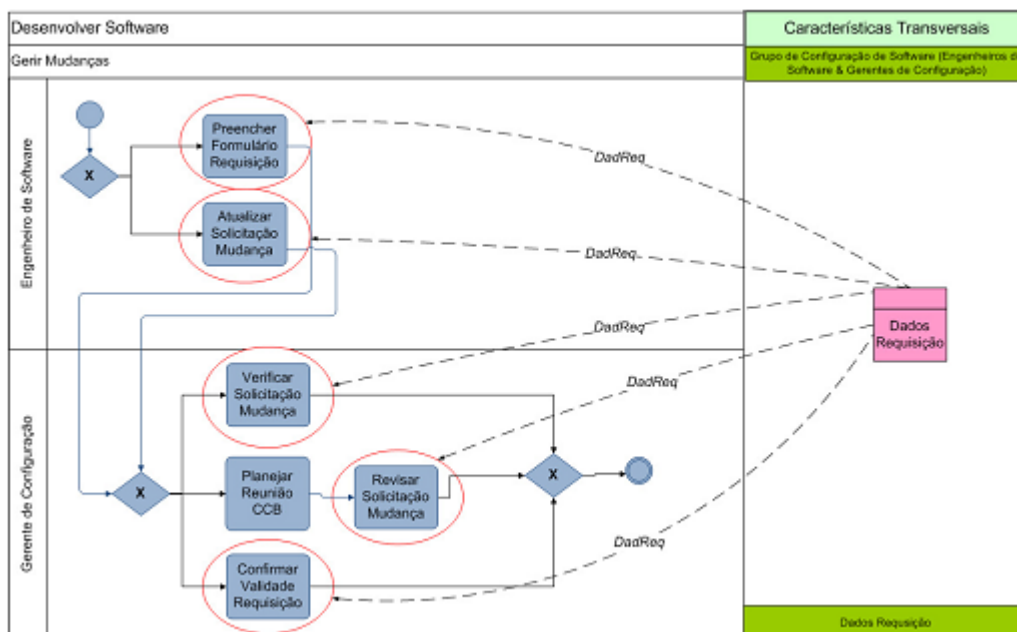


Figura 20 Visão de Dados – As atividades que utilizam um conjunto de informações [CAPPELLI, 2009]

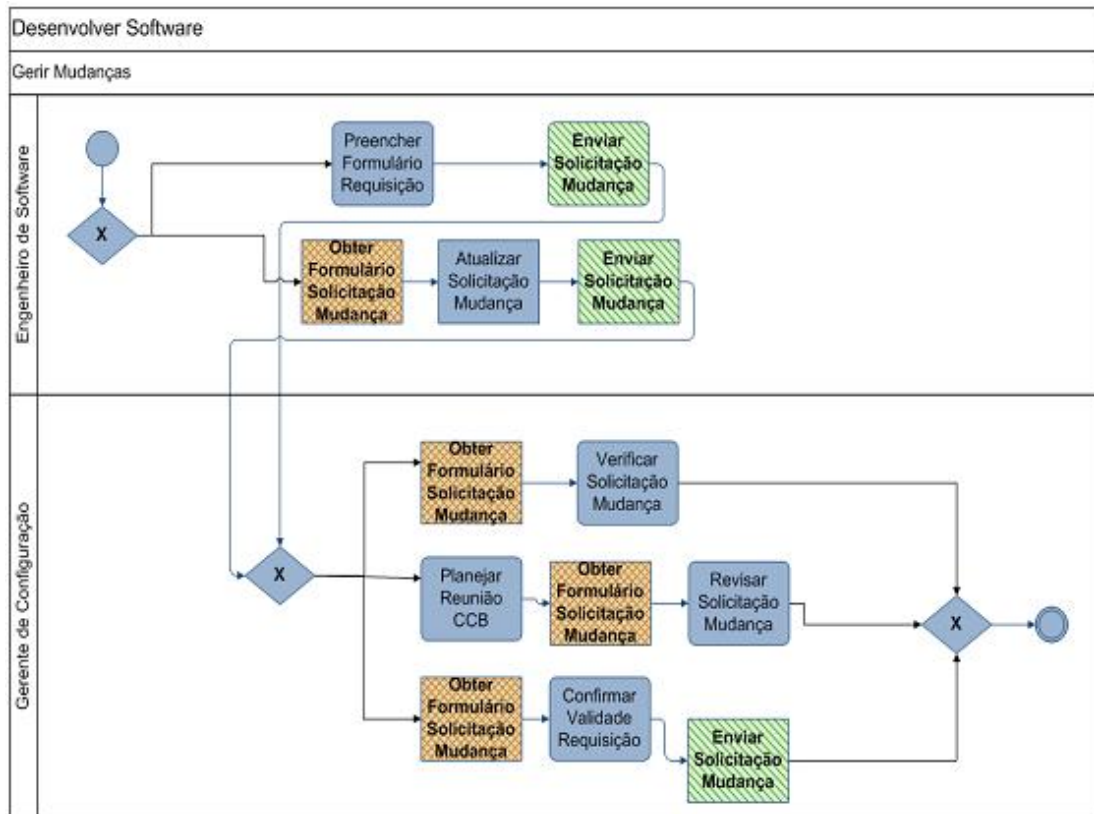


Figura 21 Visão de Atividades – Repetição [CAPPELLI, 2009]

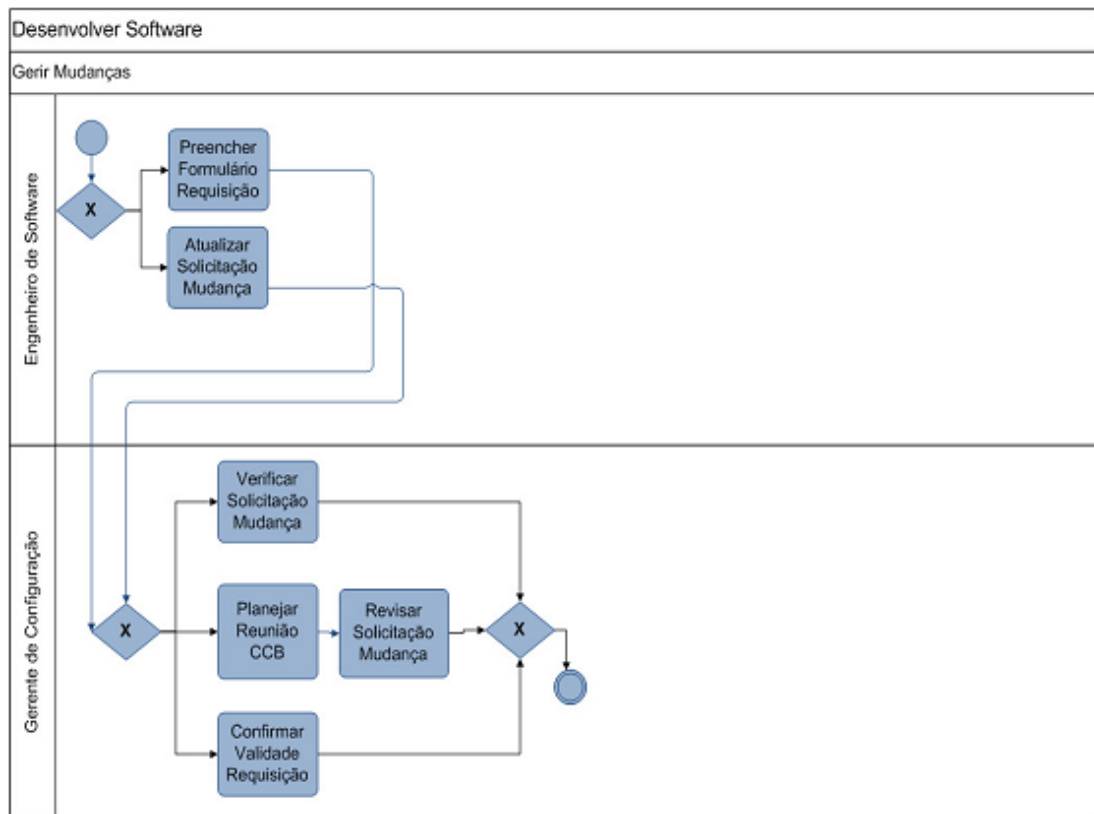


Figura 22 Visão de Atividades Principais [CAPPELLI, 2009]

A tese em seguida faz uma aplicação do modelo proposto para a inserção de características de transparência em modelos de processos organizacionais.

4.2. Sobre a Ferramenta

ARIS Express 2.2 é uma ferramenta de código fechado de uso gratuito que serve como ponto de partida para a Gestão de Processos de Negócio [ARIS Express 2.2]. É uma versão mais leve e simples que a ARIS Platform [ARIS Platform] e é destinada a usuários ocasionais e iniciantes no processo de gestão. Além disso, os diagramas criados podem ser importados no ARIS Platform.

O programa é composto pela página inicial (Figura 23), com as opções dos tipos diagramas e uma área com os diagramas usados recentemente, e pela área de trabalho para modelagem dos diagramas. Os nove tipos de diagramas são [ARIS Community]:

- *Organizational chart* (Gráfico organizacional): fornece ferramentas para modelar a estrutura da organização mostrando o relacionamento entre unidades, papéis e pessoas, assim como localização destes.
- *Data model* (Modelo de dados): modela a estrutura dos dados na notação entidade-relacionamento, inclusive suas propriedades.
- *BPMN diagram* (Diagrama BPMN): permite a modelagem de processos utilizando a notação BPMN 2.0 [BPMN, 2010], inclusive diagramas de colaboração.
- *Whiteboard*: usado para memória e estruturação de idéias e tarefas como por exemplo o resultado de uma sessão de *brainstorming*.
- *Process landscape* (Panorama de processos): para estruturação da cadeia de valor da organização e também para representar hierarquia entre processos.
- *System landscape* (Panorama de sistemas): para mostrar os sistemas utilizados e suas relações como os domínios em que são utilizados, assim como suas propriedades (sistema operacional e banco de dados por exemplo).

- *IT infrastructure* (Infraestrutura de TI): permite a representação de como os componentes de TI estão organizados, quais sistemas estão disponibilizados em quais hardware e as redes que os interligam.
- *Business process* (Processo de negócio): representa processos como uma sequência de atividades e eventos, na notação EPC (*Event-driven Process Chain*) [KELLER *et al.*, 1992].
- *General diagram* (Diagrama genérico): fornece elementos gráficos livres de semântica para a criação de outros tipos de diagramas.

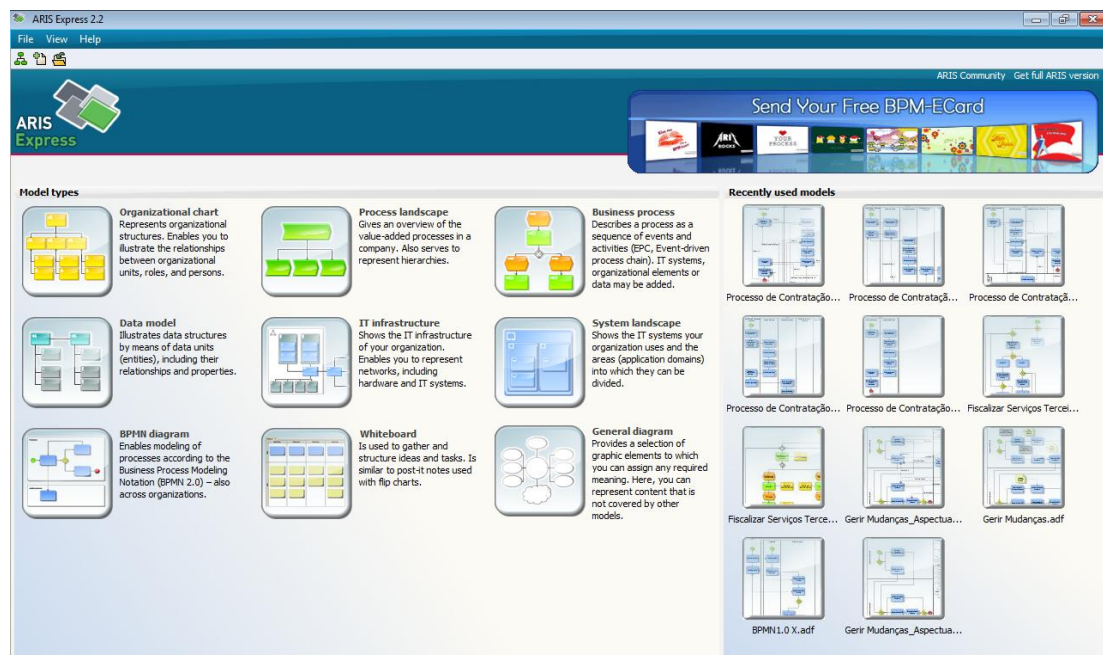


Figura 23 Página inicial da ferramenta

A Figura 24 mostra a área de trabalho do tipo *BPMN diagram*. A área de trabalho dos outros tipos de diagrama são basicamente as mesmas, o que muda são os símbolos da janela *Symbols* que fica à direita no topo, destacada na Figura 25, e os fragmentos da janela *Fragments* que fica à direita embaixo, destacada na Figura 26.

A opção de criação de *Fragments* é um recurso interessante da ferramenta. Os *Fragments* são fragmentos de processos contendo um ou mais objetos, que podem ser reutilizados em outros processos. A janela *Fragments* contém os fragmentos disponibilizados pela própria ferramenta ou criados pelo usuário.

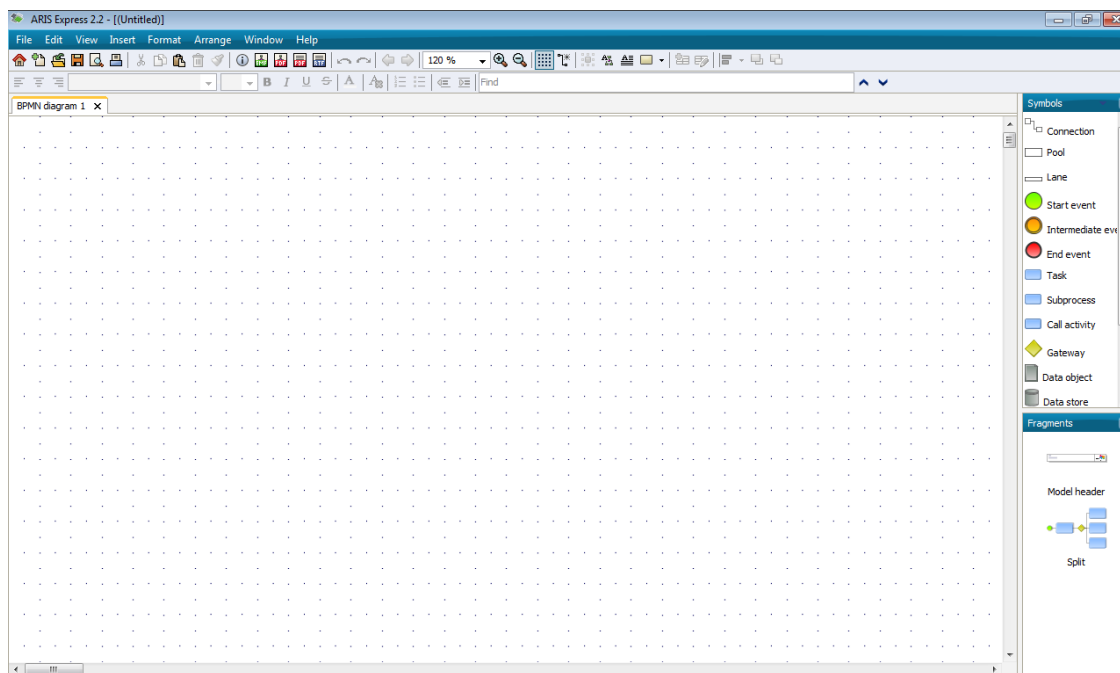


Figura 24 Área de trabalho para modelos BPMN da ferramenta ARIS Express 2.2

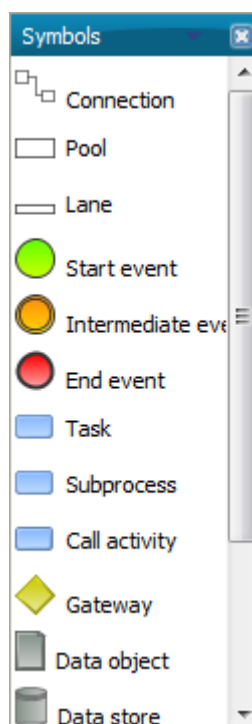


Figura 25 Janela de símbolos BPMN para utilizar nos diagramas BPMN

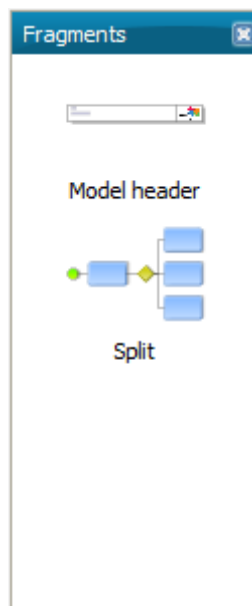


Figura 26 Janela de fragmentos em BPMN para utilizar nos diagramas BPMN

Além dos fragmentos, outro recurso interessante da ferramenta é o *SmartDesign* que permite a criação automática de fluxos de processo ou estruturas dos diagramas através de uma planilha. Esta opção só não está disponível para os tipos *BPMN diagram* e *Whiteboard*.

Além disso, outra facilidade é a mini barra de ferramentas que aparece ao selecionar um elemento qualquer, que mostra as ações disponíveis mais utilizadas, acelerando a modelagem.

Os formatos disponíveis de exportação dos diagramas são PDF, imagens (JPEG, PNG e EMF) e ADF. Este último é o formato específico de diagramas modelados no ARIS Express 2.2. Diferentemente do ARIS Platform, o ARIS Express não usa banco de dados para armazenar os modelos e sim arquivos do formato ADF [ARIS Community].

4.2.1. Instalação

Para instalação da ferramenta, deve-se fazer uma conta na ARIS Community [ARIS Community] gratuita onde também é possível obter suporte para a ferramenta. No primeiro acesso à ferramenta é necessário inserir uma conta da ARIS Community que é validada *online*. Nos próximos acessos não é necessário estar conectado na

Internet. Por ser uma aplicação baseada em Java [JAVA, 1995], ela pode ser utilizada em qualquer plataforma que suporte Java e por isso, deve-se instalar o Java Runtime Environment antes. Apesar disso, ARIS Express 2.2 oficialmente só suporta a plataforma Microsoft Windows [ARIS Community].


4.3. Exemplo Utilizando o Processo “Gerir Mudanças”


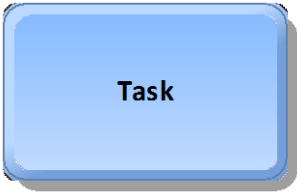
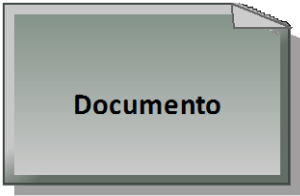
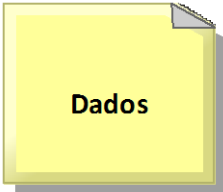



O processo “Gerir Mudanças” foi modelado novamente em BPMN [BPMN, 1997] na ferramenta, como mostra a Figura 27. Este processo como dito na seção anterior está dentro do contexto de desenvolvimento de software e é responsável pelo controle de requisições de mudanças. Apenas foram feitas duas pequenas mudanças no modelo da tese para padronização dos nomes dos elementos, todas as palavras “Requisição” foram modificadas para “Solicitação Mudança” e onde havia “Solicitação” apenas adicionou-se “Mudança”.


O processo inicia quando o Engenheiro de Software necessita fazer uma nova solicitação de mudança ou atualizar uma existente. Após o preenchimento ou atualização, o engenheiro envia o formulário para o Gerente de Configuração. O gerente pode então realizar três ações, apenas verificar a solicitação de mudança; planejar uma reunião com o grupo de controle de configuração (*Configuration Control Boarding*) e depois revisar a solicitação de mudança; ou confirmar a validade da solicitação de mudança e enviar a solicitação de mudança.

A semântica dos símbolos utilizados nos modelos é apresentada na Tabela 1 abaixo.

Tabela 1 Sintaxe da linguagem BPMN dos símbolos utilizados nos modelos

Símbolo	Descrição
	Representa um evento que inicia o processo.

Símbolo	Descrição
	Representa o operador lógico “XOR” que quando divide o fluxo do processo, apenas um caminho é seguido. Quando mais de um fluxo converge neste elemento, apenas um é considerado.
	Representa uma atividade.
	Representa um documento consumido ou gerado pela atividade.
	Representa dados, informação consumida ou gerada pela atividade.
	Representa um evento final do processo.
	<i>Pools</i> (piscinas) representam unidades organizacionais independentes. <i>Lanes</i> (raias) representam papéis na organização.
	Associação entre atividades e documentos ou entre atividades e dados. O sentido da seta define se a atividade consome ou gera o documento ou os dados.

Símbolo	Descrição
	Associação entre atividades, eventos e operadores lógicos, indicando o fluxo do processo.

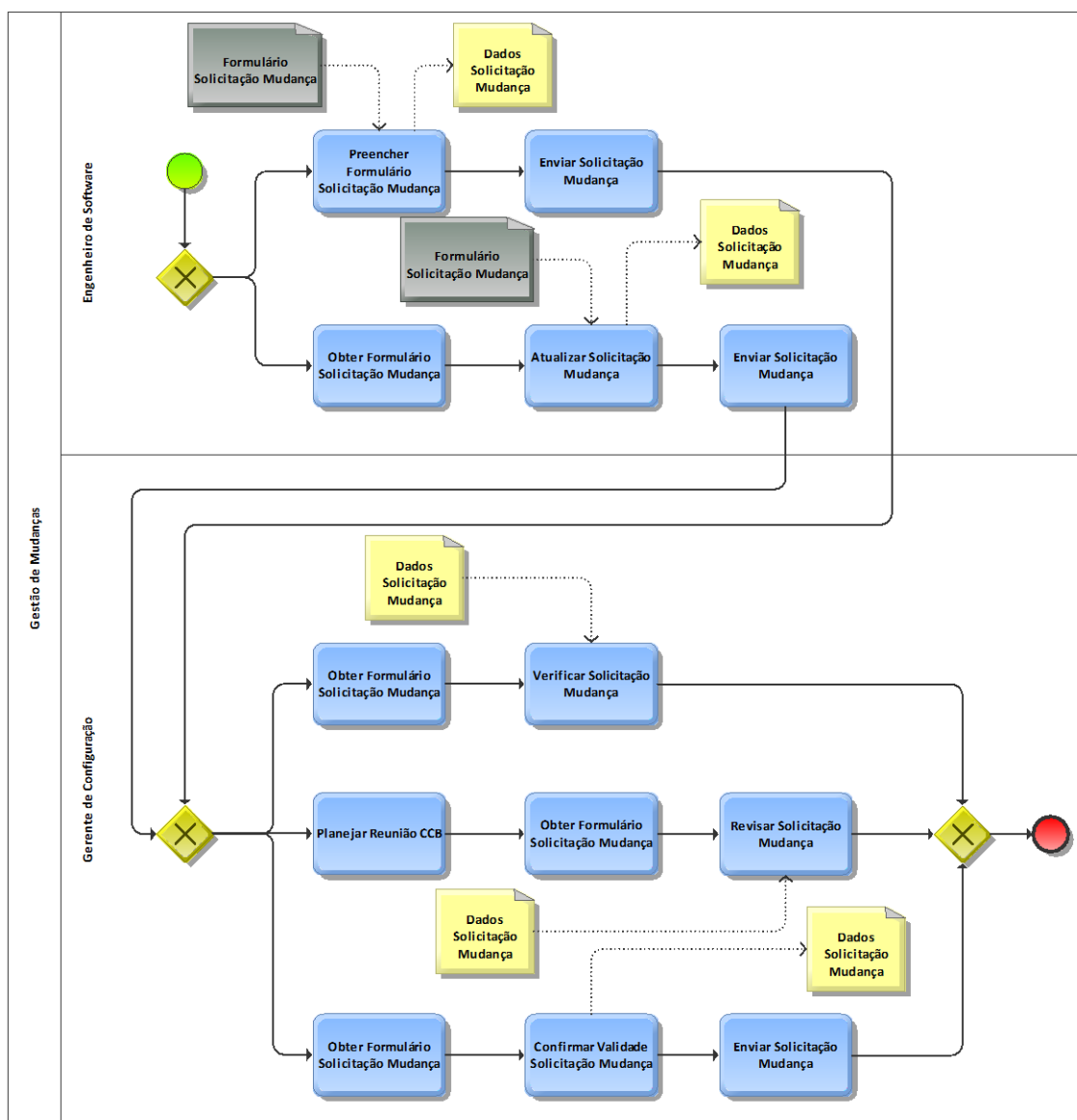


Figura 27 Processo “Gerir Mudanças”. Adaptado de Cappelli [2009] e modelado no ARIS Express 2.2

Como apresentado na tese de Cappelli [2009], as atividades “Obter Formulário Solicitação Mudança”, “Enviar Solicitação Mudança”, o documento

“Formulário Solicitação Mudança” e os dados “Dados Solicitação Mudança” se repetem. Esses elementos são então separados do modelo e compostos no modelo aspectualizado apresentado na Figura 28. Os relacionamentos transversais são apresentados abaixo:

Aspecto: Obter Formulário Solicitação Mudança

Tipo: Atividade

Declaração Transversalidade:

Relacionamento: ObtForSolMud

Onde: Before

[TodasAtividades [Atualizar Solicitação Mudança]
AND [Verificar Solicitação Mudança] AND [Revisar
Solicitação Mudança] AND [Confirmar Validade
Solicitação Mudança]]

Ação: Include

[Obter Formulário Solicitação Mudança]

Aspecto: Enviar Solicitação Mudança

Tipo: Atividade

Declaração Transversalidade:

Relacionamento: EnvSolMud

Onde: After

[TodasAtividades [Preencher Formulário Solicitação
Mudança] AND [Atualizar Solicitação Mudança] AND
[Confirmar Validade Solicitação Mudança]]

Ação: Include

[Enviar Solicitação Mudança]

Aspecto: Formulário Solicitação Mudança

Tipo: Documento

Declaração Transversalidade:

Relacionamento: ForSolMud

Onde: Before

[TodasAtividades [Preencher Formulário Solicitação
Mudança] AND [Atualizar Solicitação Mudança]]

Ação: Include

[Formulário Solicitação Mudança]

Aspecto: Dados Solicitação Mudança

Tipo: Dados

Declaração Transversalidade:

Relacionamento: DadSolMud

Onde: Before

[TodasAtividades [Verificar Solicitação Mudança]
AND [Revisar Solicitação Mudança]] AND

After

[TodasAtividades [Preencher Formulário Solicitação
Mudança] AND [Atualizar Solicitação Mudança] AND
[Confirmar Validade Solicitação Mudança]]

Ação: Include

[Dados Solicitação Mudança]

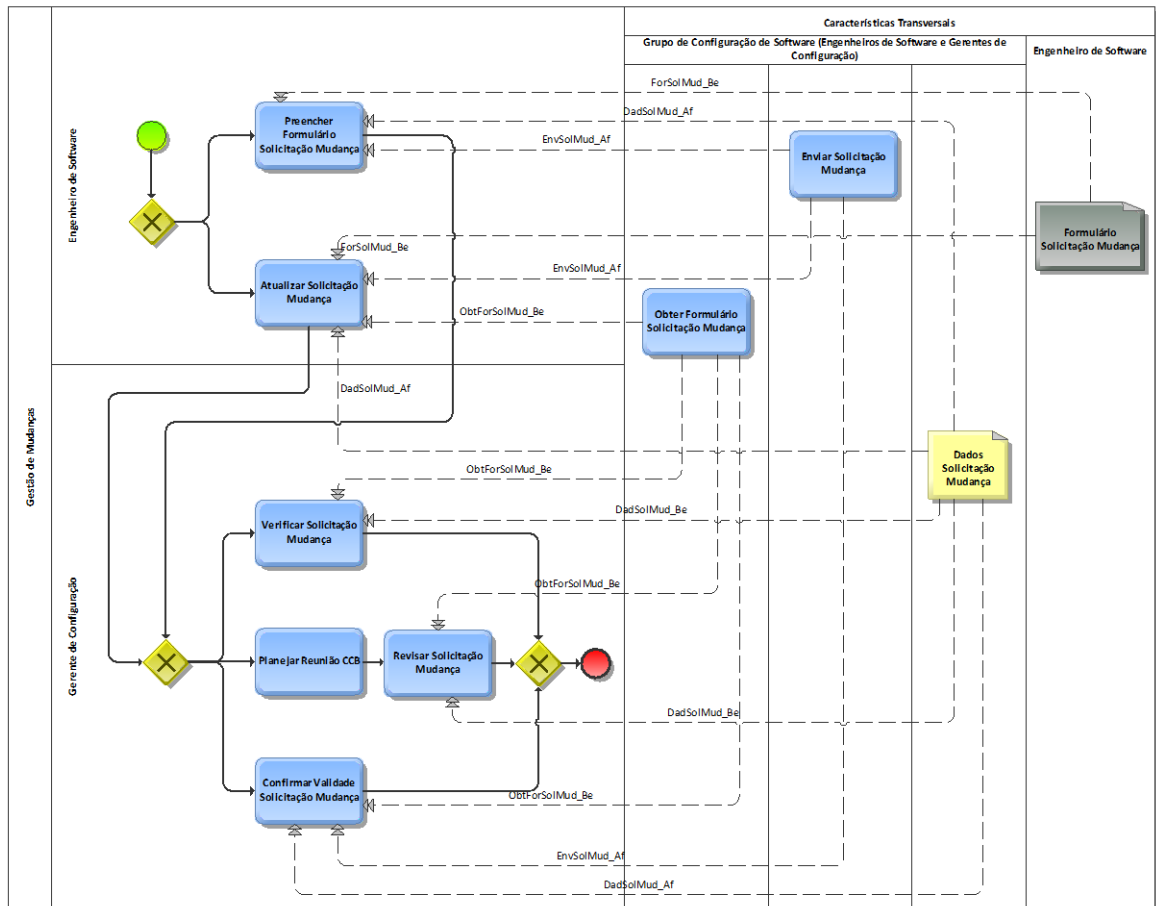


Figura 28 Processo “Gerir Mudanças” aspectualizado. Modelado no ARIS Express 2.2, opção *BPMN diagram*

O passo a passo para criar diagramas aspectualizados no ARIS Express 2.2 é detalhado na subsecção 4.4.

4.4. Sistemática para Representação Visual de Aspectos na Ferramenta

A sistemática para a representação visual de aspectos está organizada em passos resumidos abaixo e descritos a seguir.

- 4.3.1. Criar *pool* transversal
- 4.3.2. Criar raias
- 4.3.3. Criar relacionamento transversal
- 4.3.4. Alterar propriedades
- 4.3.5. Representar modelo final

4.4.1. Criar *Pool* transversal

Com o processo modelado (na direção horizontal neste exemplo), deve-se criar uma nova *pool*, selecionando esta opção na janela de símbolos. Ao clicar na opção, basta clicar na área de modelagem, posicionando o cursor ao lado direito do diagrama pronto e clicar novamente. A nova *pool* será criada automaticamente na direção ortogonal ao do processo básico. No caso de um diagrama na direção vertical, a nova *pool* deve ser posicionada abaixo, para que fique ortogonal à do processo básico. A Figura 29 ilustra o resultado, a nova *pool* aparece com o nome padrão *Pool* e foi renomeada para “Características Transversais”.

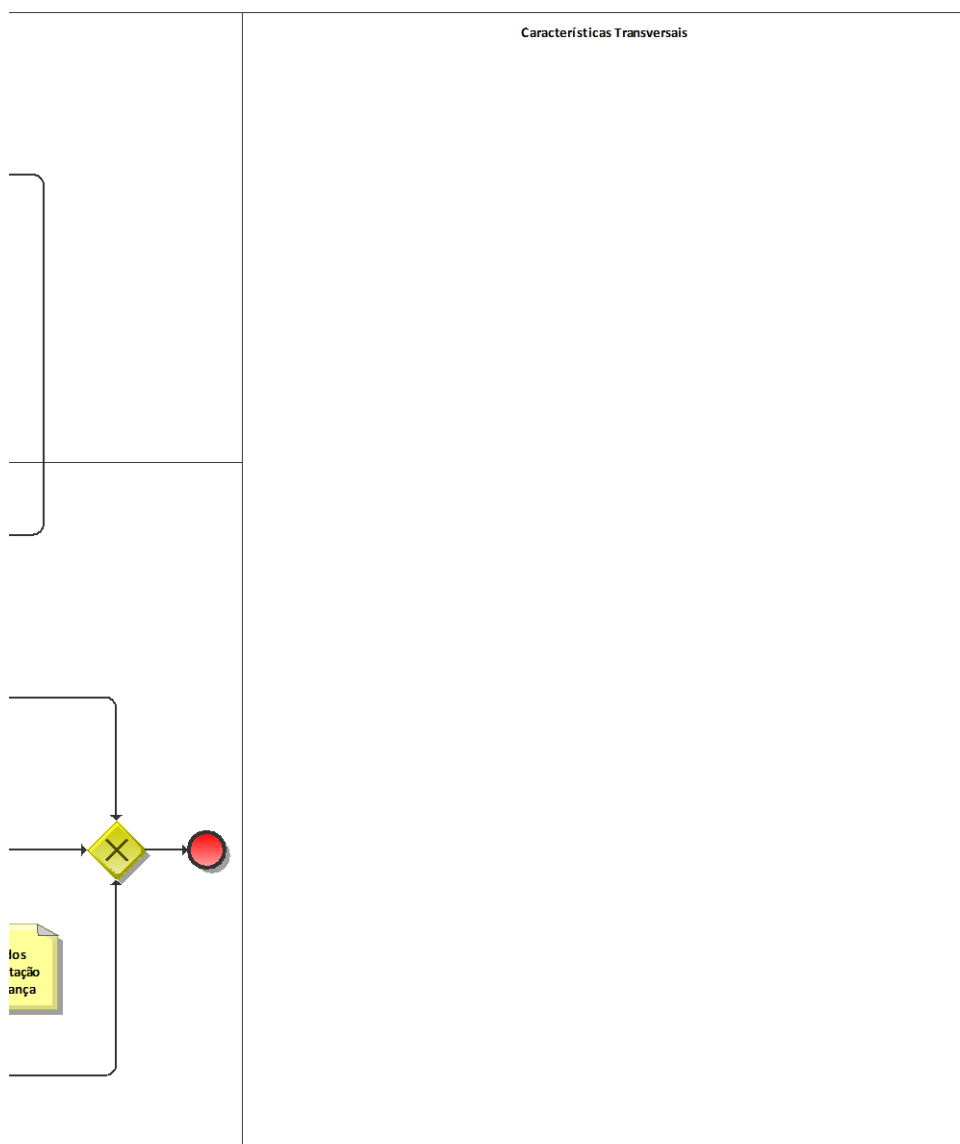


Figura 29 Nova *pool* para as características transversais ao processo

4.4.2. Criar raias

A seguir devem-se criar as raias (*lanes*), de acordo com os papéis responsáveis pelos aspectos. Deve-se selecionar a opção raia da janela de símbolos e clicar em cima da nova *pool* ou de outras raias para criar uma hierarquia de papéis e uma nova raia será criada. Também é possível criar novas raias clicando com o botão direito e selecionando a opção “BPMN” (Figura 30). O resultado é apresentado na Figura 31, já com os aspectos posicionados nas respectivas raias.

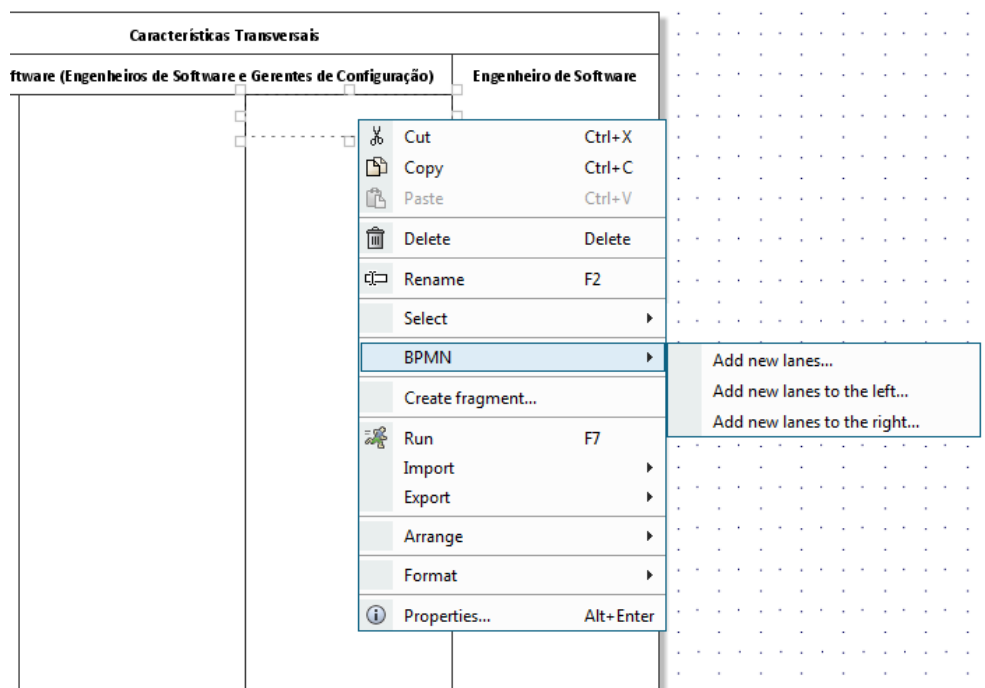


Figura 30 Criando novas raia através do menu ao clicar com o botão direito do mouse na *pool* ou raia

aparecem automaticamente em uma mini barra de ferramentas, como mostra a Figura 32.

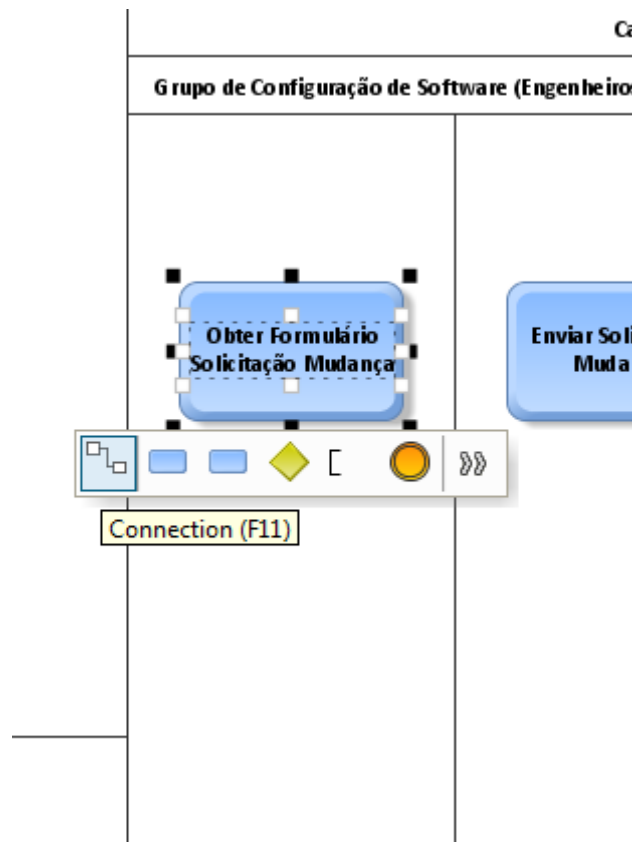


Figura 32 Selecionando a opção *Connection* para criar a associação entre o aspecto e o elemento afetado pelo aspecto

O tipo da associação é criado automaticamente. A associação entre atividades de *pools* diferentes (*message flow*) possui semântica própria e representação gráfica na modelagem convencional de processos. Entre atividade e documentos ou dados, a associação também possui semântica própria, junto com outra representação gráfica. Por isso, é necessário diferenciar essas associações do relacionamento transversal, criando uma nova representação gráfica para ele. No exemplo, ao criar a associação entre a o aspecto “Obter Formulário Solicitação Mudança” e a atividade “Atualizar Solicitação Mudança”, o tipo *message flow* foi automaticamente utilizado, como mostra a Figura 33.

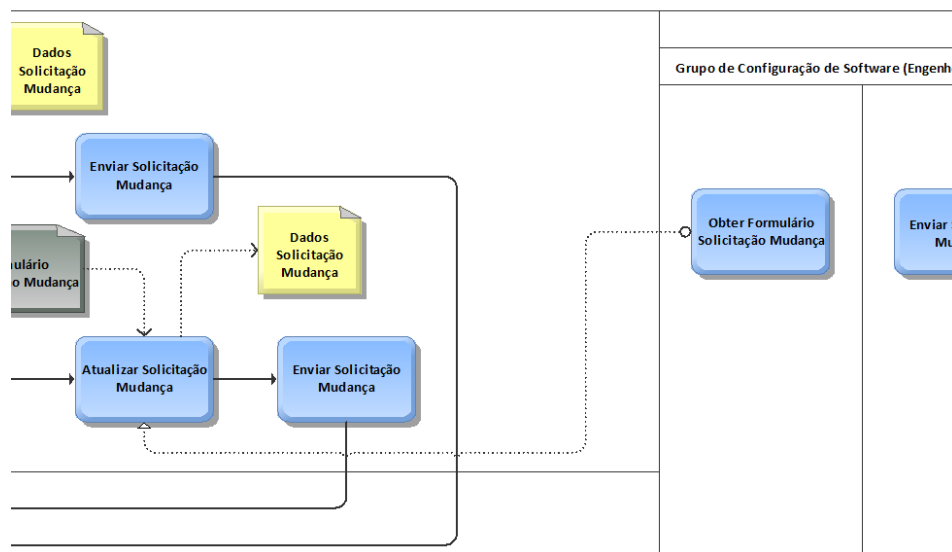


Figura 33 Associação criada

4.4.4. Alterar propriedades

Para modificar graficamente a associação, deve-se dar dois cliques na linha ou clicar com o botão direito na linha e selecionar *Properties...*

Na seleção *Attributes* das propriedades da associação, deve-se dar o nome do relacionamento transversal para o atributo *Name* (Figura 34). Além do nome do relacionamento transversal, foram colocadas as iniciais do tipo *advice* para facilitar na visualização, se o aspecto é inserido *after* (_Af), *before* (_Be) ou *around* (_Ar) do elemento afetado.

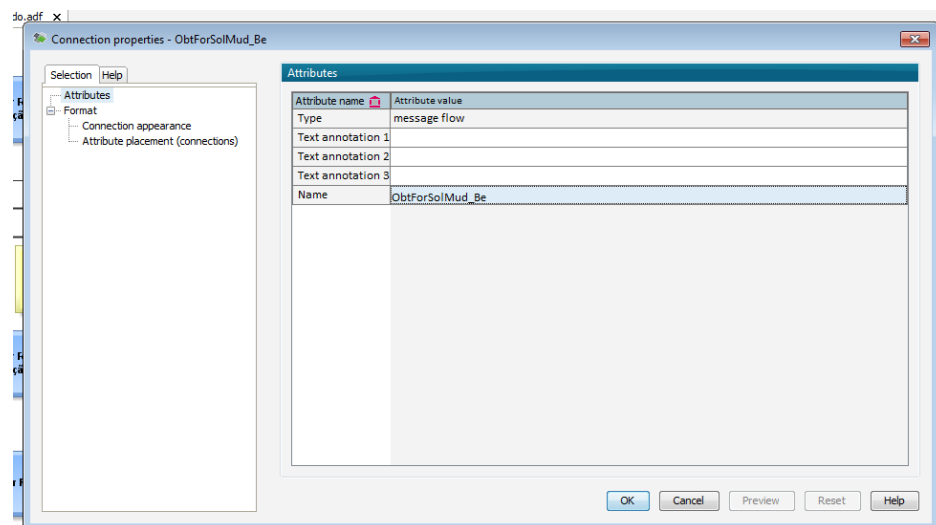


Figura 34 Nomeando a associação com o nome do relacionamento transversal

Na seleção *Connection appearance*, o estilo da linha é modificado para *Dashed* (Figura 35).

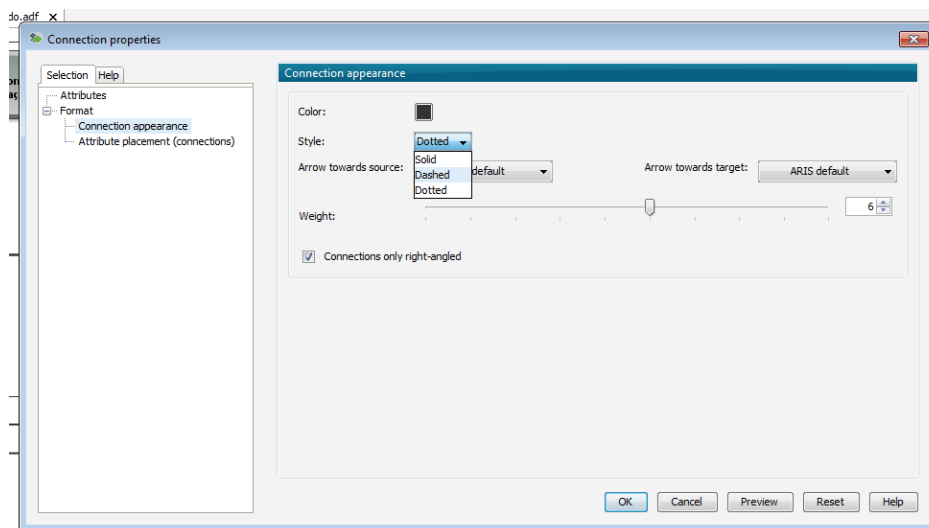


Figura 35 Selecionando linha no estilo *Dashed*

Ainda na seleção *Connection appearance*, as opções marcadas com um círculo vermelho na Figura 36 também forma modificadas.

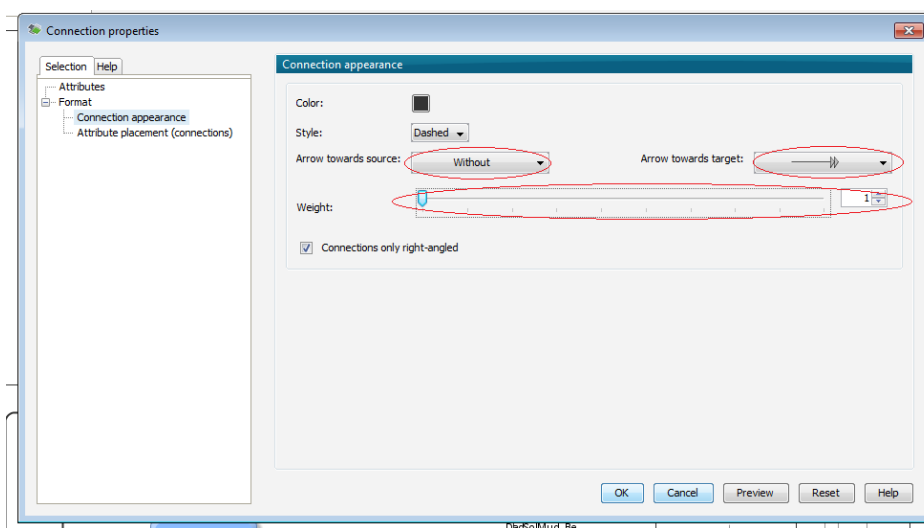


Figura 36 Aparência das setas e densidade da linha

E por fim, na seleção *Attribute placement (connections)*, deve-se clicar no botão *Add...* à direita e selecionar o atributo *Name* (Figura 37), clicar em OK e finalizar, clicando em OK novamente.

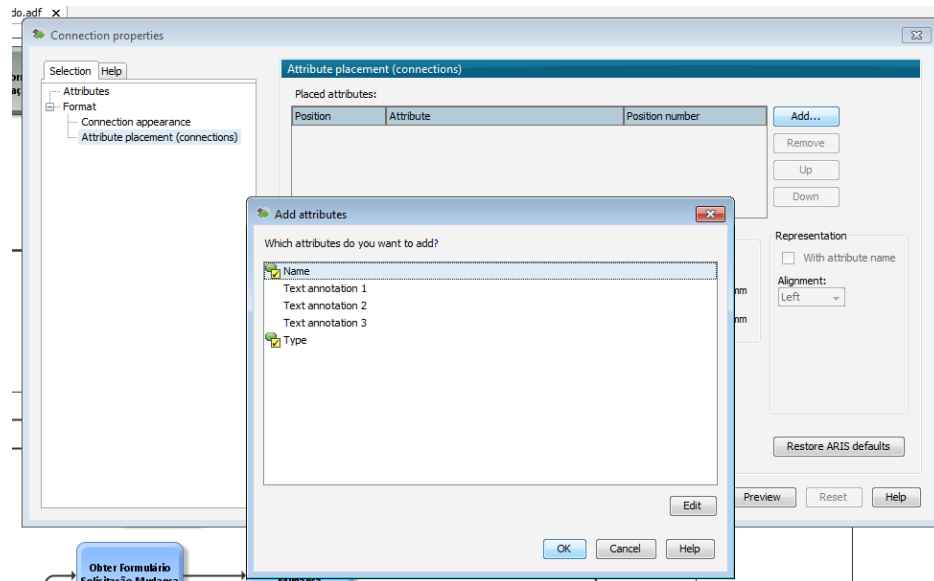


Figura 37 Adicionando atributo *Name* para aparecer no diagrama junto à linha

O relacionamento transversal terá então a aparência final da associação na Figura 38 (exemplo ObtForSolMud_Be).

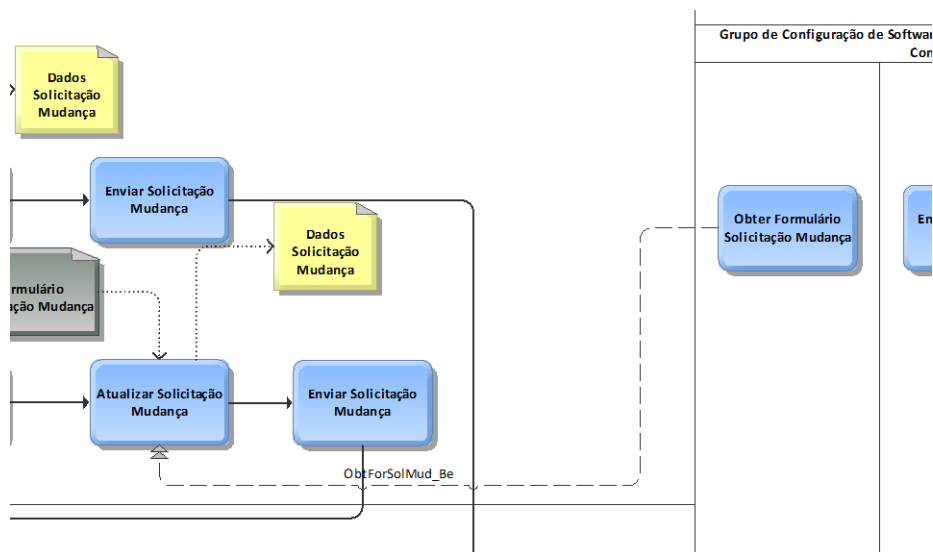


Figura 38 Aparência do relacionamento transversal com o exemplo ObtForSolMud_Be.

Para facilitar a criação do relacionamento transversal, pode-se utilizar os *Fragments*. Através deste recurso, é possível criar um fragmento contendo o relacionamento transversal com todas as características visuais, evitando repetir para cada relacionamento a sequência de passos.

Para a criação de um fragmento com o relacionamento transversal, deve-se conectá-lo entre dois elementos pois não é possível criar fragmento apenas de uma

conexão. A Figura 39 apresenta um exemplo de um fragmento para reutilizar o relacionamento transversal e a opção para criação do fragmento. No exemplo o relacionamento liga duas atividades genéricas e não possui o atributo *Name* preenchido (o nome depende do aspecto que representa). Desta forma, apenas a Figura 35 e a Figura 36 foram utilizadas para a aparência visual da associação. Apenas os objetos selecionados farão parte do fragmento.

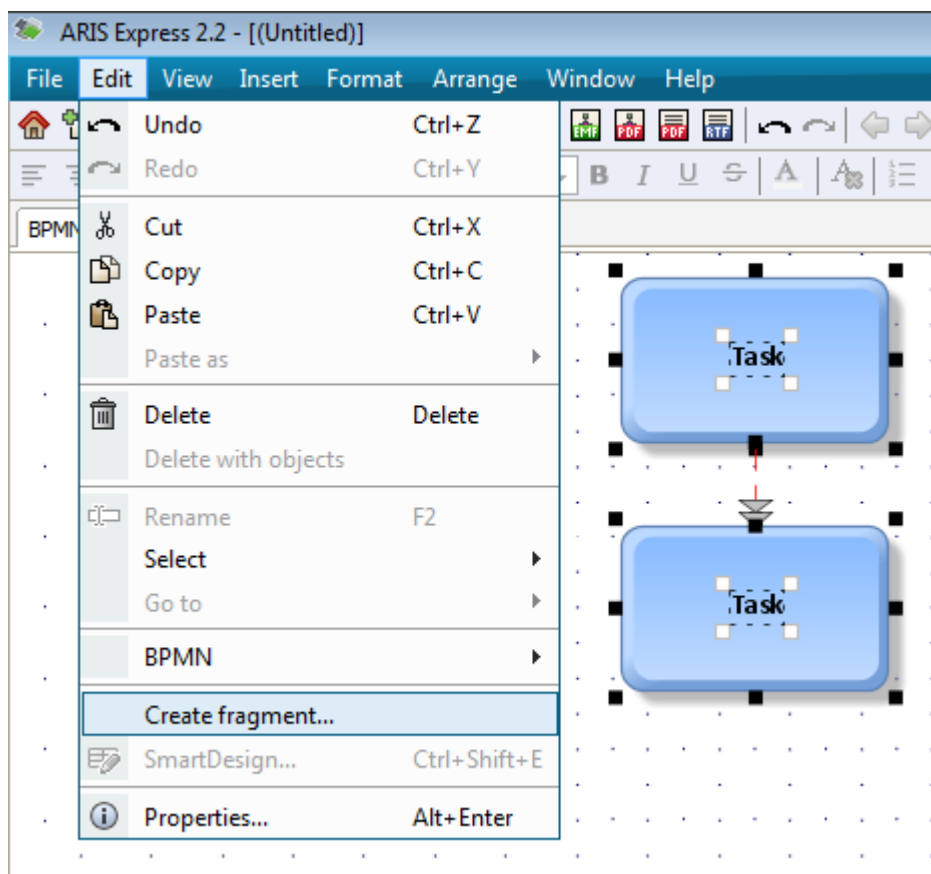


Figura 39 Selecionando opção *Create fragment...* com os objetos selecionados

Após a seleção da opção *Create fragment...*, é necessário nomear o fragmento. A Figura 40 ilustra a ação.

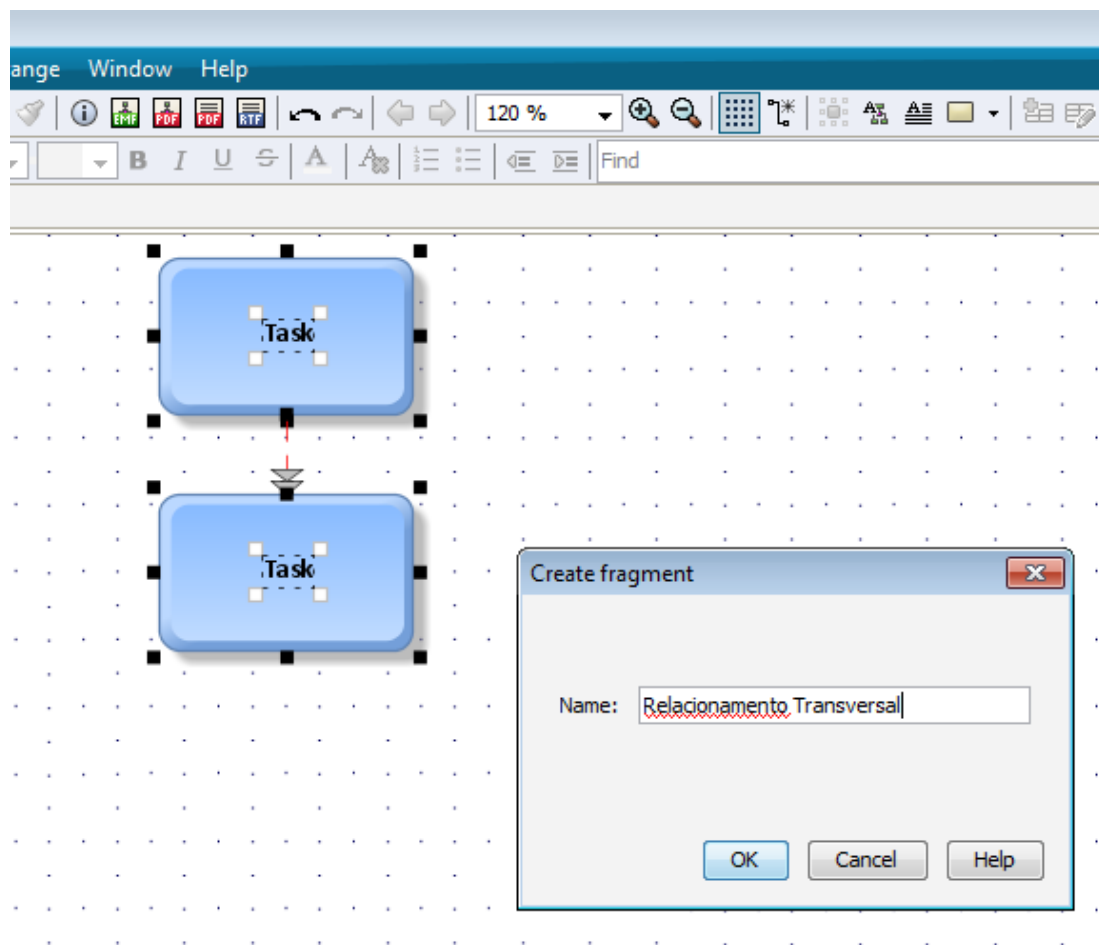


Figura 40 Nomeando o fragmento

Com o fragmento criado, ele aparecerá na janela *Fragments* disponível para uso no mesmo ou em outros diagramas do mesmo tipo (Figura 41).

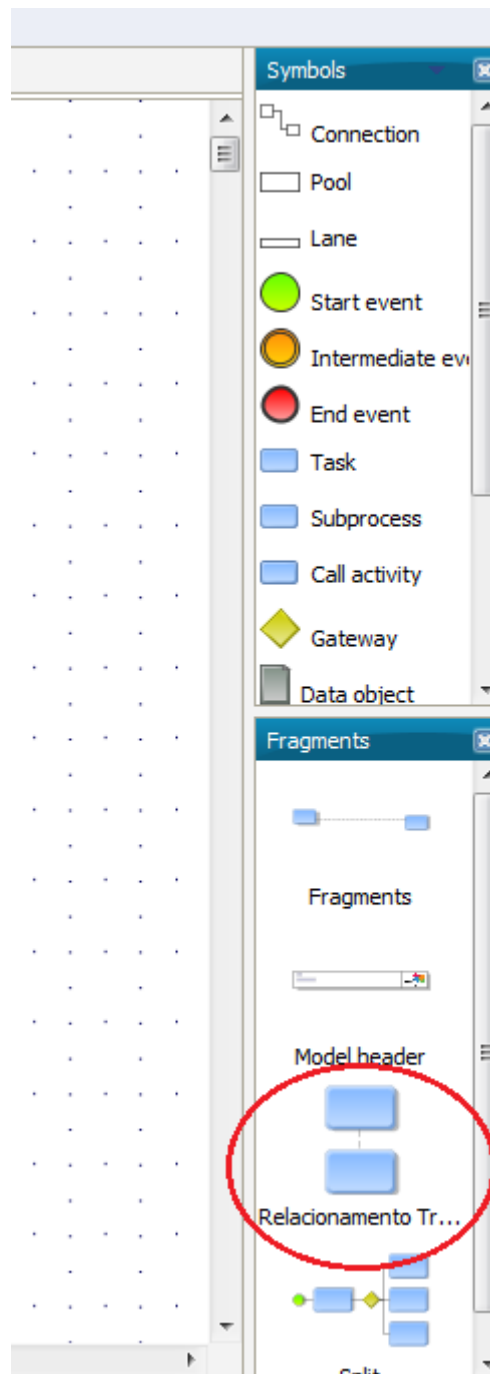


Figura 41 Fragmento criado aparecendo na janela *Fragments*

Para utilizar o fragmento, basta clicar nele com o botão esquerdo do mouse e clicar com o mesmo botão onde quiser posicioná-lo no diagrama, como é feito com os elementos da janela *Symbols*. Os elementos ligados pelo relacionamento transversal no fragmento podem ser reutilizados (se o aspecto for uma atividade afetando outra atividade por exemplo) renomeando-os. Para utilizar apenas o relacionamento transversal, basta selecioná-lo e clicar com o botão esquerdo do

mouse no pequeno quadrado preto que fica na origem da associação (quando selecionada) e mover até a borda do elemento aspectual e clicar novamente. O destino (onde fica a seta da associação e também um pequeno quadrado preto quando selecionada) também pode ser reposicionado no elemento afetado. Depois é só deletar os objetos que sobraram.

4.4.5. Representar modelo final

Repetindo esses passos para todos os aspectos, deve-se chegar na Figura 28, após algumas mudanças de posicionamento para tornar o diagrama o mais limpo possível.

A partir do modelo aspectualizado, é possível criar a representação em formato texto, substituindo os elementos transversais e os relacionamentos por caixas de texto na opção *Insert free-form text* na barra de ferramentas no topo da área de trabalho da ferramenta (Figura 42).

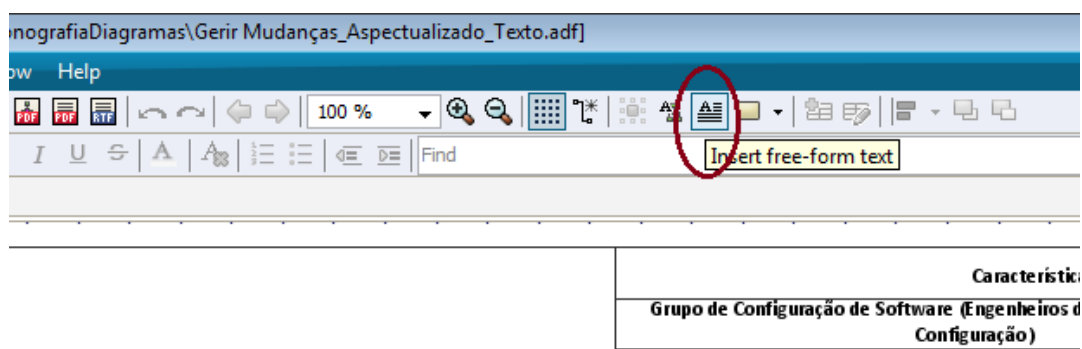


Figura 42 Selecionando a opção caixa de texto *Insert free-form text*.

O diagrama aspectualizado em formato texto é apresentado na Figura 43.

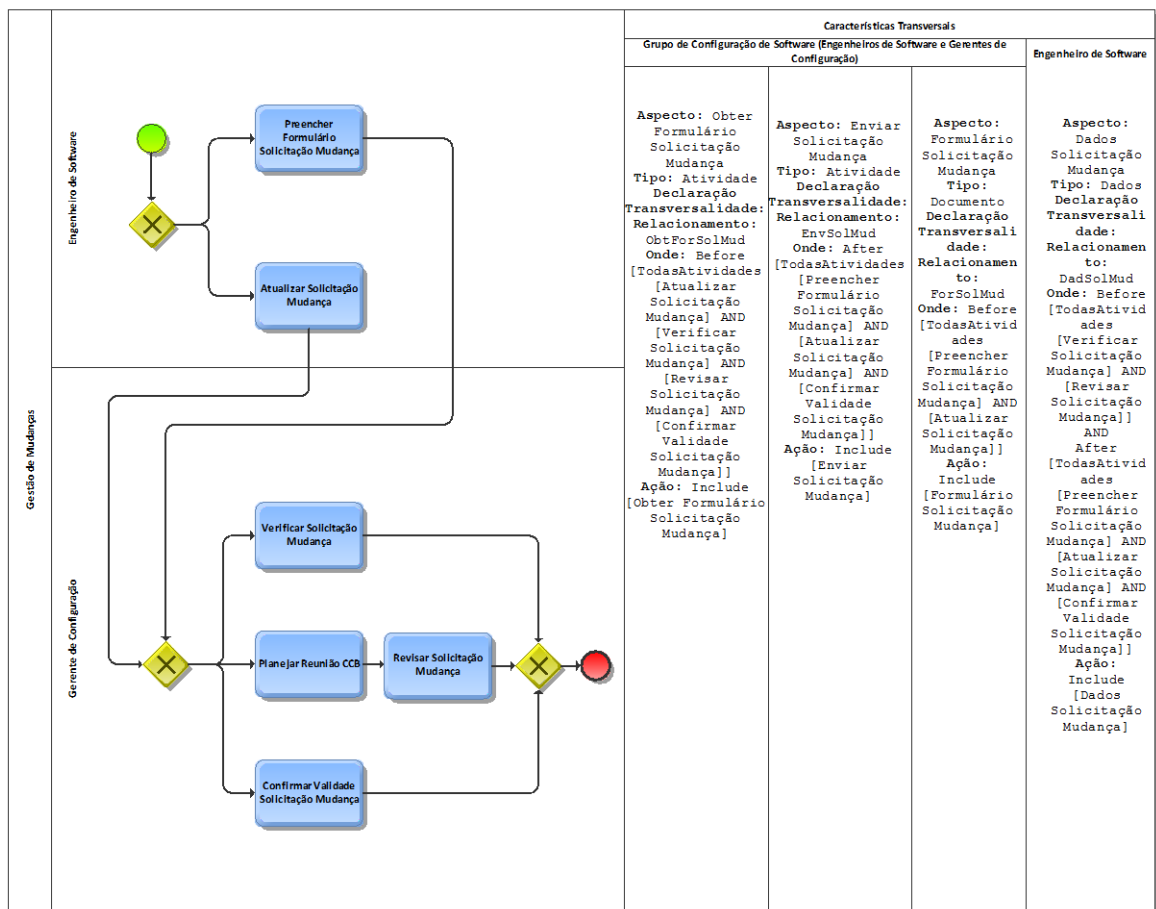


Figura 43 Processo “Gerir Mudanças” aspectualizado – Formato Texto. Modelado no ARIS Express 2.2.

Para as diferentes visões, o procedimento é o mesmo da criação gráfica do relacionamento transversal, o que muda são os elementos que aparecem no processo base e os elementos transversais mostrados. Basta definir os parâmetro de visualização (Figura 10) e modelar de acordo.

4.5. Estudo de Caso

Para o estudo de caso foi utilizada a dissertação de mestrado de Magdaleno [2006] que propõe utilizar processos de negócio para planejamento e explicitação de como a colaboração deve funcionar dentro nas organizações. O modelo de maturidade em colaboração (ColabMM) proposto é um guia para melhoria e avaliação de aspectos de colaboração em processos de negócio. A partir deste modelo, foi desenvolvido um método para introdução de práticas relacionadas à

colaboração nos modelos de processo. Como isto representa a introdução de elementos novos nos processos existentes ou explicitação de elementos que estão espalhados e/ou entrelaçados no modelo de processos, a colaboração pode ser representada como uma característica transversal. Além disso, colaboração não faz parte da essência do que o processo executa, o que também contribui para defini-la como uma característica transversal.

A Figura 44 ilustra o ColabMM, apresentando os quatro níveis de maturidade de colaboração, junto com as práticas específicas de cada nível que são também realizadas nos níveis subsequentes. O nível casual não possui práticas, pois não apresenta formalização de práticas colaborativas no processo.

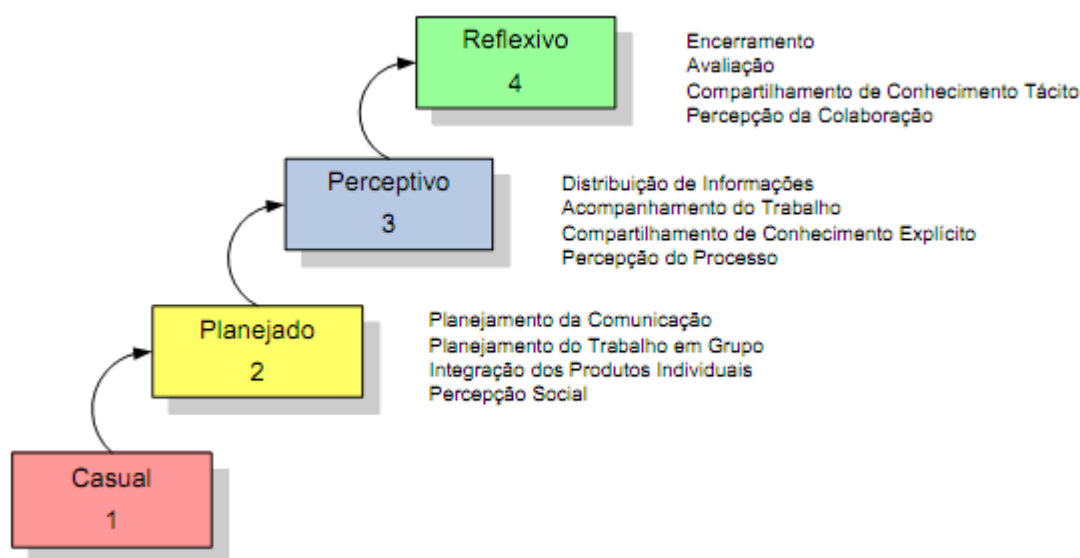


Figura 44 Modelo ColabMM [MAGDALENO, 2006]

O método proposto para explicitar a colaboração durante a modelagem define etapas baseadas nos níveis e práticas do ColabMM. Para ilustração do método, a autora utilizou como exemplo um processo genérico de contratação de software utilizado pelas organizações que desejam adquirir serviços de desenvolvimento de software.

O processo, apresentado na Figura 45 (modelado na direção vertical neste exemplo), inicia com o gerente de projeto elaborando a requisição de software que contem as especificações do software que se deseja contratar. Depois é feita a pré-seleção dos candidatos pelo analista de contratos com base em uma pesquisa de

mercado e a requisição de software é enviada a cada candidato. Depois de recebidas as propostas, os candidatos são avaliados e um é selecionado. Após a elaboração e assinatura do contrato, o trabalho da empresa contratada é iniciado na data determinada sob a supervisão do gerente de projetos que faz o controle de mudanças Documento de Requisição de Software (DRS) durante o desenvolvimento.

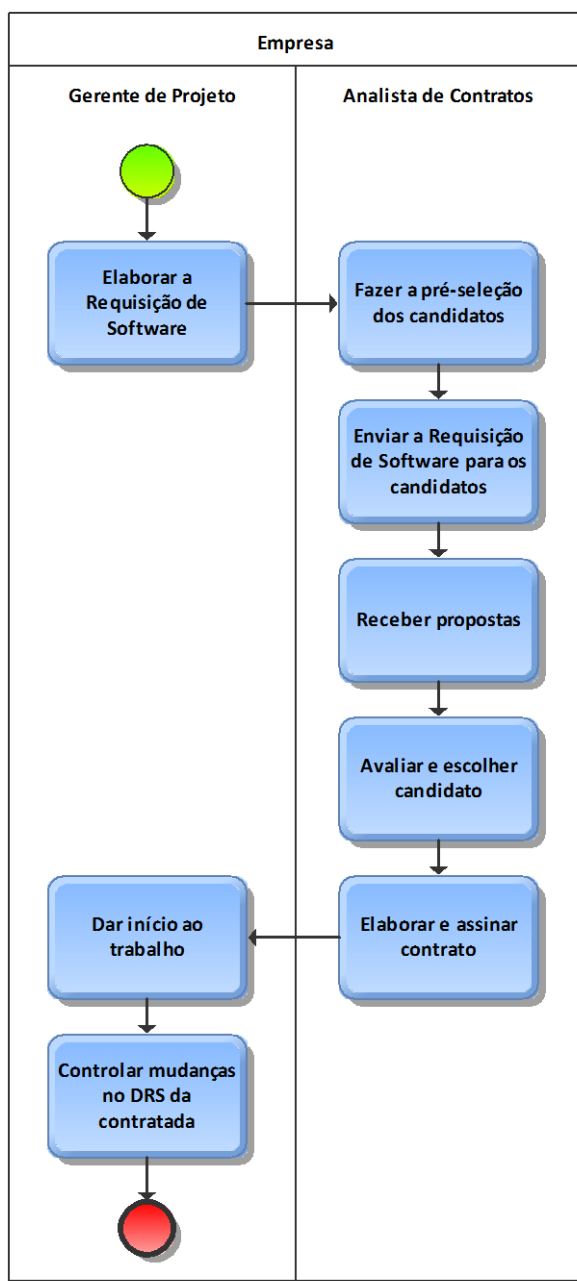


Figura 45 “Processo de Contratação de Software”. Adaptado de Magdaleno [2006]

Seguindo o método proposto, resumido e ilustrado na Figura 46, o “Processo de Contratação de Software” sofreu algumas alterações para se adequar à proposta.

As etapas que definem a inclusão de uma atividade (“Incluir Atividade de Socialização” e “Incluir Atividade de Encerramento” por exemplo), a etapa “Elaborar Modelo de Interação entre Papéis do Processo” e a etapa “Detalhar Atividades de Elaboração Conjunta de Artefatos” especificamente modificaram o processo no nível de detalhamento apresentado na Figura 45.

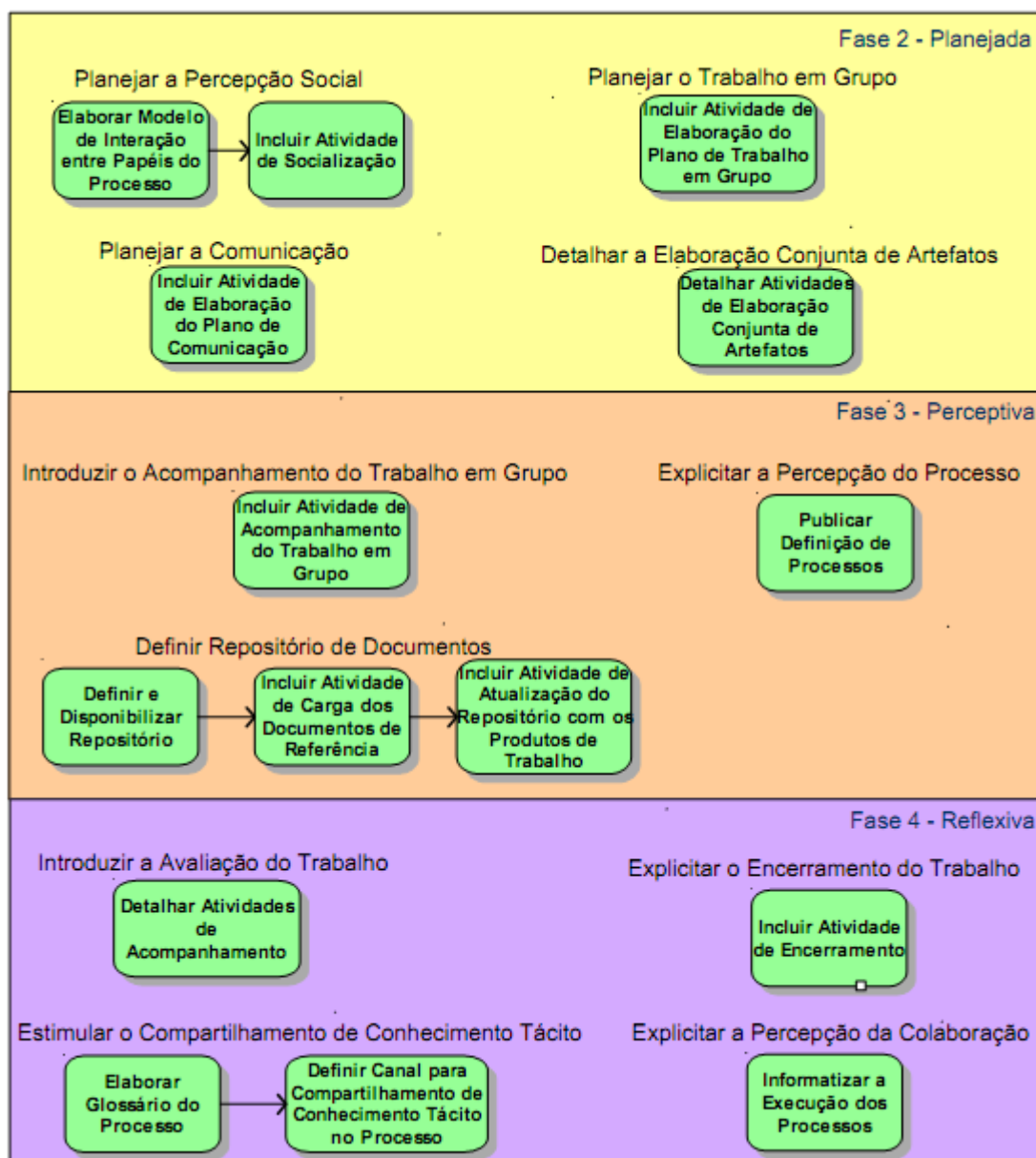


Figura 46 Resumo do método de inserção de características de colaboração nos processos de negócio [MAGDALENO, 2006]

A etapa “Elaborar Modelo de Interação entre Papéis do Processo” explicitou o papel da “Contratada” que antes não aparecia e definiu novas combinações de

interação entre os papéis para execução colaborativa das atividades, criando novas raia. A atividade “Elaborar requisição de software” passou a ser executado pelo gerente de projeto e o analista de contratos pela definição da etapa “Detalhar Atividades de Elaboração Conjunta de Artefatos”.

A etapa “Incluir Atividade de Acompanhamento do trabalho em grupo” foi a única que no caso deste processo não incluiu uma nova atividade pois foi identificada que suas características já estavam presentes na atividade “Controlar mudanças no DRS da contratada”. A Figura 47 mostra o processo após a execução do método.

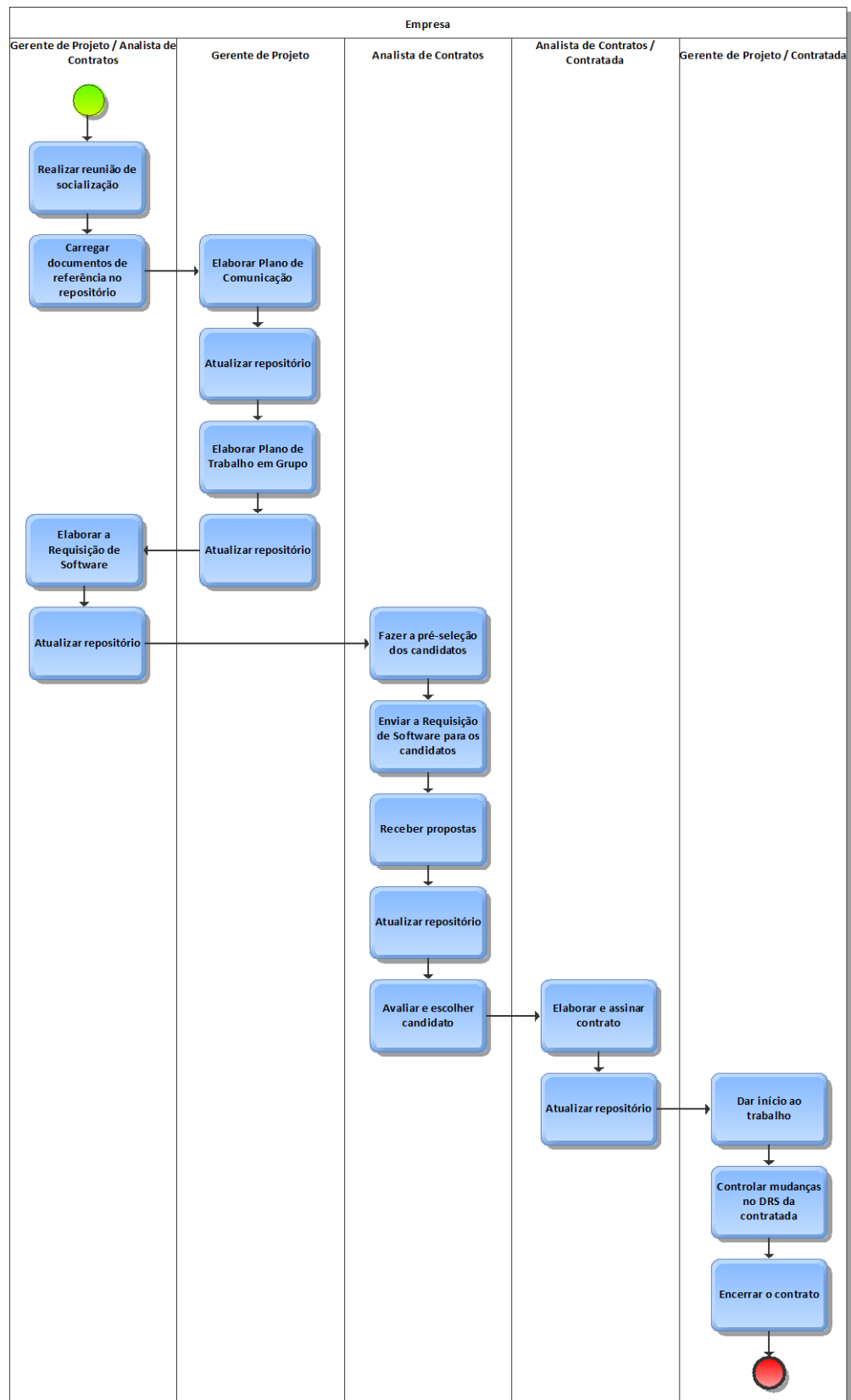


Figura 47 “Processo de Contratação de Software” com elementos de colaboração. Adaptado de Magdaleno [2006].

Como dito anteriormente, colaboração pode ser tida como um aspecto para os processos de uma empresa. No caso deste processo, para a introdução da colaboração, algumas atividades e papéis foram incluídos. Diferentemente do processo “Gerir Mudanças”, este processo apresenta papéis como aspectos e também duas sequências de atividades, além de atividades como no exemplo do “Gerir Mudanças”.

O modelo atualizado é apresentado na Figura 49. Os papéis “Analista de Contratos” e “Contratada” (em amarelo com o símbolo de um boneco) estão em raiais sem nome, já que não faz sentido colocar o nome do executor de um papel. Para representar graficamente estes elementos, foi utilizado o mesmo elemento *Task* das atividades, mas com um símbolo específico denominado *User task* em *Symbol* e a cor em *Fill color*: alterada para amarelo (RGB: 255, 255, 102). Tudo isso nas propriedades do objeto em *Properties...*, *Object appearance* como na Figura 48.

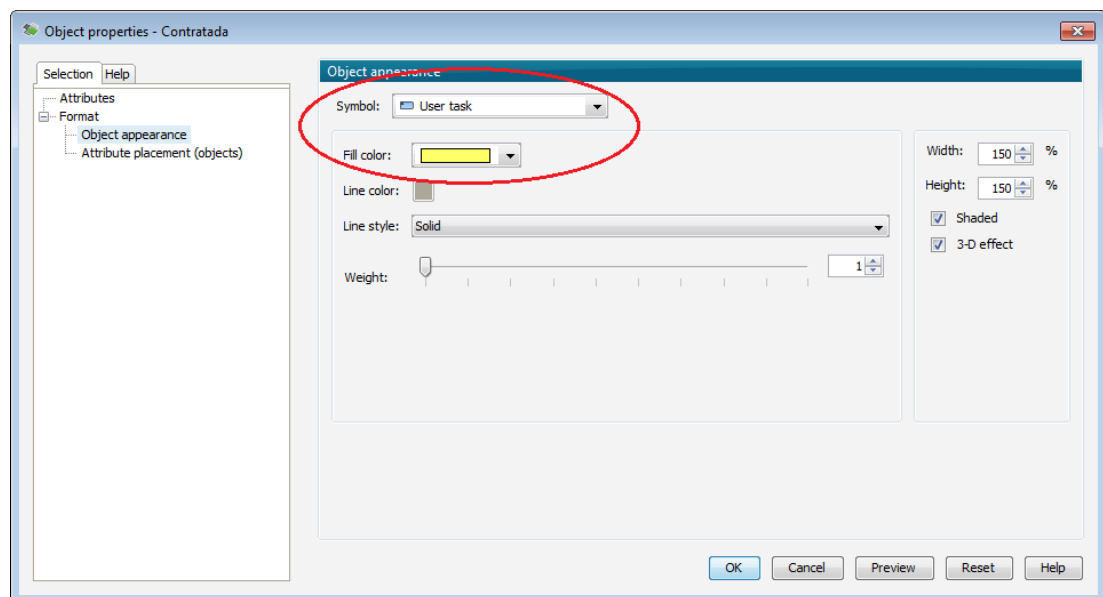


Figura 48 Aparência do elemento para representar um papel

Também para os papéis utilizou-se para nomeação do relacionamento transversal o “_Ar” de *Around* (durante) já que um papel atua durante a atividade e não antes (*Before*) ou após (*After*).

Para as sequências de atividades, dependendo de onde são inseridas (*Before* ou *After*), foi utilizada a lógica de que se uma sequência é inserida antes (*Before*), então o relacionamento transversal deve partir do último elemento da sequência já

que no modelo com o processo base e os elementos aspectuais misturados, após o último elemento da sequência entra o elemento afetado. Assim, quando a sequência é inserida após (*After*) o relacionamento transversal deve partir do primeiro elemento da sequência.

A escolha por sequências ao invés de separar as atividades foi para manter a granularidade dos aspectos mais grossa, simplificando a visualização e também, por se tratarem de atividades relacionadas a um mesmo conceito (colaboração), faz sentido que permaneçam juntas. De qualquer forma, esta resolução depende de cada caso e cabe na modelagem a definição das diretrizes para granularidade dos aspectos.

Já atividades no meio de sequências não podem ser separadas pois os elementos antes e após também são aspectos e por isso deve-se mantê-las unidas como o caso da atividade “Elaborar Plano de Trabalho em Grupo”. No caso da segunda atividade “Atualizar repositório” da sequência mais abaixo no diagrama (Figura 49), poderia não fazer parte da sequência e apenas incluir mais um relacionamento transversal partindo do aspecto “Atualizar repositório” ligando-o à atividade “Fazer a pré-seleção dos candidatos” como *Before*. No entanto a semântica seria afetada pois essa atividade está fortemente ligada à anterior da sequência (“Elaborar Plano de Trabalho em Grupo”) em que o repositório é atualizado com o plano elaborado anteriormente.

Os relacionamentos transversais são apresentados abaixo:

Aspecto: Analista de Contratos

Tipo: Papel

Declaração Transversalidade:

Relacionamento: AnContr

Onde: Around

[TodasAtividades [Elaborar a Requisição de Software]]

Ação: Include

[Analista de Contratos]

Aspecto: Contratada

Tipo: Papel

Declaração Transversalidade:

Relacionamento: Contrat

Onde: Around

[TodasAtividades [Elaborar e assinar contrato] AND
[Dar início ao trabalho] AND [Controlar mudanças
no DRS da contratada]]

Ação: Include

[Contratada]

Aspecto: Atualizar repositório

Tipo: Atividade

Declaração Transversalidade:

Relacionamento: AtRep

Onde: After

[TodasAtividades [Elaborar a Requisição de
Software] AND [Receber propostas] AND [Elaborar e
assinar contrato]]

Ação: Include

[Atualizar repositório]

Aspecto: Encerrar contrato

Tipo: Atividade

Declaração Transversalidade:

Relacionamento: EncContr

Onde: After

[TodasAtividades [Controlar mudanças no DRS da
contratada]]

Ação: Include

[Encerrar contrato]

Aspecto: Realizar reunião de socialização / Carregar
documentos de referência no repositório

Tipo: Sequência de Atividades

Declaração Transversalidade:

Relacionamento: RealSocial_CarregDocsRefRep

Onde: Before

[TodasAtividades [Elaborar a Requisição de
Software]]

Ação: Include

[Realizar reunião de socialização] AND [Carregar
documentos de referência no repositório]

Aspecto: Elaborar Plano de Comunicação / Atualizar
repositório / Elaborar Plano de Trabalho em Grupo /
Atualizar repositório

Tipo: Sequência de Atividades

Declaração Transversalidade:

Relacionamento: ElabPlanCom_AtRep_ElabPlanTrab_AtRep

Onde: Before

[TodasAtividades [Fazer a pré-seleção dos
candidatos]]

Ação: Include

[Elaborar Plano de Comunicação] AND [Atualizar
repositório] AND [Elaborar Plano de Trabalho em
Grupo] AND [Atualizar repositório]

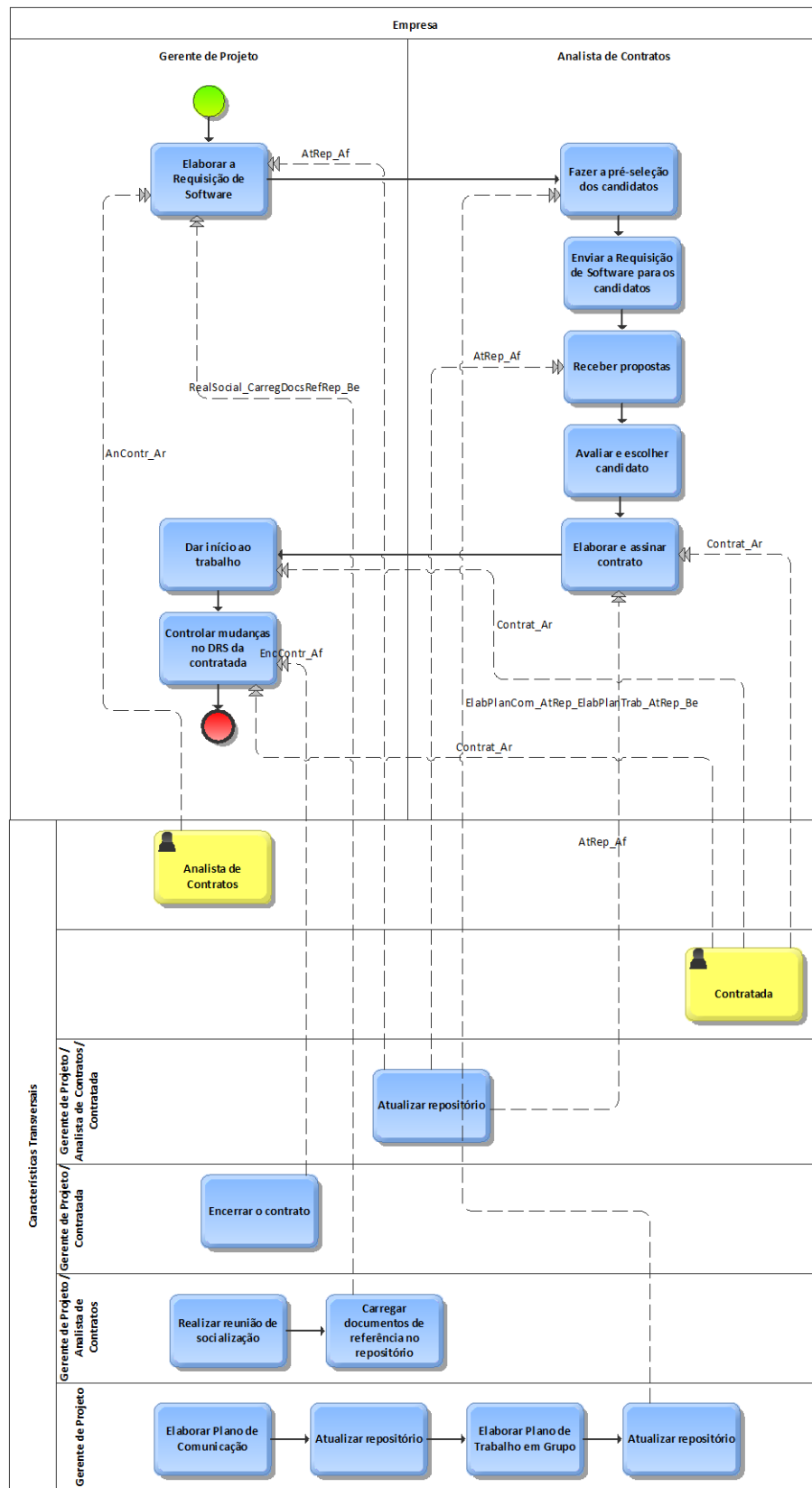


Figura 49 “Processo de Contratação de Software” com elementos de colaboração aspectualizados

Outra forma de visualização poderia ser colocando os novos papéis nas raiais do processo base e apenas representar as atividades como aspectos (Figura 50). Neste caso os relacionamentos transversais são os mesmos, apenas não tem os papéis que já aparecem nas raiais do processo base.

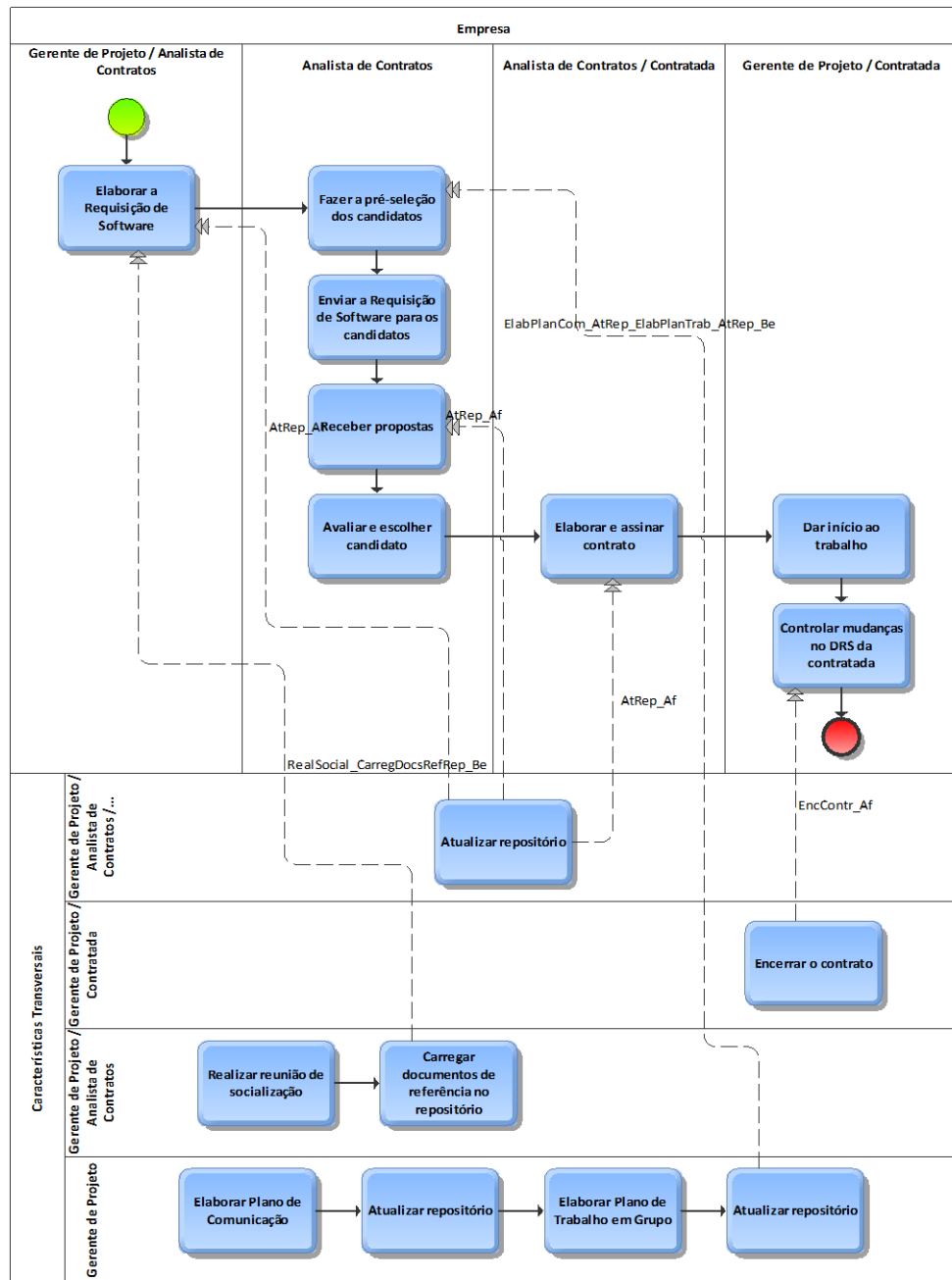


Figura 50 “Processo de Contratação de Software” com elementos de colaboração e atividades de colaboração aspectualizadas

Uma outra visualização do processo poderia ser aspectualizando apenas as atividades que acessam o repositório, ou seja, as atividades “Atualizar repositório” e

“Carregar documentos de referência no repositório” (Figura 51). Desta forma, ter-se-ia uma visão do processo sem atividades apenas registram armazenam informação, mas não alteram ou criam nova informação. Neste caso os relacionamentos transversais são representados como:

Aspecto: Atualizar repositório

Tipo: Atividade

Declaração Transversalidade:

Relacionamento: AtRep

Onde: After

[TodasAtividades [Elaborar a Plano de Comunicação]
AND [Elaborar de Trabalho em Grupo] AND [Elaborar
a Requisição de Software] AND [Receber propostas]
AND [Elaborar e assinar contrato]]

Ação: Include

[Atualizar repositório]

Aspecto: Carregar documentos de referência no repositório

Tipo: Atividade

Declaração Transversalidade:

Relacionamento: CarregDocsRefRep

Onde: After

[TodasAtividades [Realizar reunião de
socialização]]

Ação: Include

[Carregar documentos de referência no repositório]

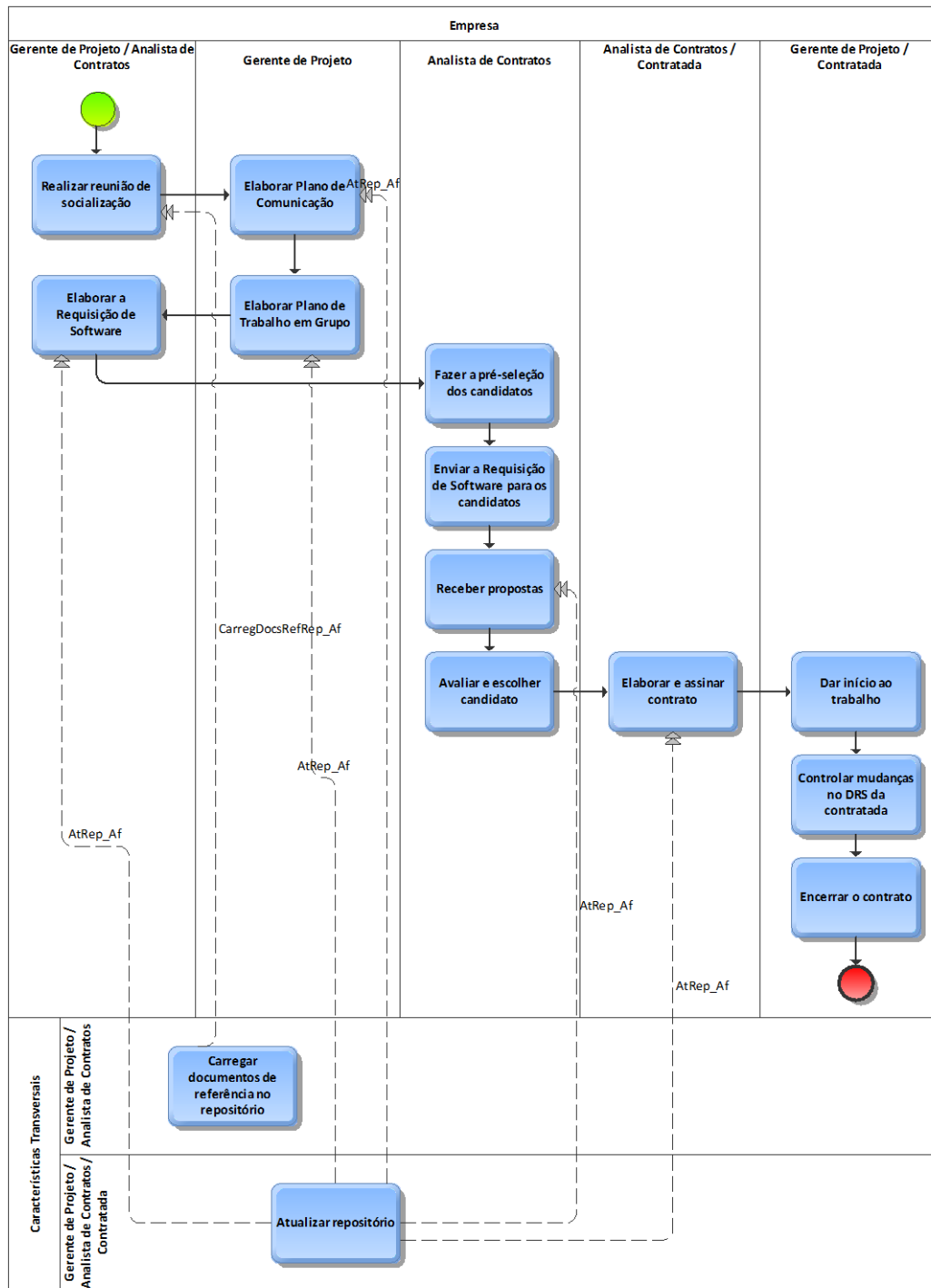


Figura 51 “Processo de Contratação de Software” com elementos de colaboração e atividades de acesso a repositório aspectualizadas

5. CONCLUSÃO

Este trabalho apresentou na introdução a motivação da proposta, como é possível facilitar o processo de modelagem, promover o reuso e a manutenção do modelo de processos, modularizando as características transversais. Os trabalhos existentes que tratam especificamente deste tema são poucos mas trazem propostas que abriram as portas para a exploração de aspectos em BPM. A partir da proposta de Cappelli [2009], foi apresentada uma aplicação a representação de aspectos na ferramenta ARIS Express 2.2.

O estudo de caso mostrou outro exemplo de como o conceito de aspectos pode ser utilizado na modelagem de processos de negócio, permitindo diferentes visões para diferentes necessidades na análise dos modelos. A proposta é uma alternativa simples para representação dos relacionamentos transversais na ferramenta ARIS Express 2.2 que pode ser reutilizada em outros diagramas.

Algumas limitações foram encontradas durante este estudo. Na análise de processos reais para utilizar como exemplo, identificou-se a dificuldade em representar aspectos quando estes se encontram entre conectores lógicos. Como representar a inclusão de aspectos quando esta depende de qual fluxo foi instanciado em uma divisão ou junção de fluxos? Além disso, há um conflito quando há inclusão de mais de um aspecto em um mesmo objeto. Se as inclusões são realizadas *Before*, como decidir qual aspecto é realizado primeiro? A mesma coisa acontece com a inclusão *After*. No caso de *Around* não há esse problema pois dois ou mais aspectos como *Around* em um mesmo elemento podem acontecer ao mesmo tempo. Isto pode ser resolvido em alguns casos pela semântica. No caso da Figura 49 e da Figura 50, a decisão de incluir a sequência de quatro atividades como *Before* na atividade “Fazer a pré-seleção dos candidatos” ou *After* na atividade “Elaborar a Requisição de Software” deveu-se ao aspecto “Atualizar repositório”. Este é incluído como *After* na atividade “Elaborar a Requisição de Software” pois semanticamente é um acesso ao repositório para armazenar a “Requisição de Software” elaborada anteriormente. Se a sequência estivesse também como *After* na atividade “Elaborar a Requisição de Software”, teria um conflito. Desta forma se optou por representá-la como *Before* na atividade “Fazer a pré-seleção dos candidatos”.

Além disso, existem casos de aspectos dentro de aspectos. Na Figura 49 e na Figura 50, a sequência de quatro atividades possui duplicada a atividade “Atualizar repositório”, que também é um aspecto. Neste caso, como tratar aspectos de menor granularidade que aparecem no meio de outros aspectos de maior granularidade.

A ferramenta também limita no que diz respeito a não permitir raias no tipo de diagrama *Business process* (EPC). Isso impede que se coloque raias transversais. No entanto, poderia fazer uma representação diferente nesse caso para os aspectos. Para acelerar a modelagem orientada a aspectos, o ARIS Express 2.2 poderia permitir salvar os relacionamentos criados e até incluir na mini barra de ferramentas dos elementos.

O trabalho mostrou como os conceitos de OA podem contribuir com a modularização de forma prática sem trazer complexidade. A ferramenta utilizada é de uso gratuito e está disponível para todos os usuários.

Como trabalhos futuros, a mesma representação gráfica feita no ARIS Express 2.2 poderia ser estudada no ARIS Platform, para ampliar o uso da abordagem, já que essa apresenta funcionalidades mais abrangentes e é mais utilizada nas empresas profissionalmente. As perguntas levantadas e limitações encontradas também podem ser estudadas para evolução do tema de representação de aspectos em modelos de processo de negócio. E principalmente, como trabalho futuro, não apenas a representação visual de aspectos, mas a integração de fato na ferramenta, para que os aspectos, relacionamentos transversais e todos os conceitos apresentados sejam “entendidos” pela ferramenta, facilitando a modelagem OA.

BIBLIOGRAFIA

- AGUILAR-SAVÉN R.S. (2004). Business process modelling: Review and framework, *International Journal of Production Economics* 90 (2004) 129–149.1.
- ARIS Community. <http://www.ariscommunity.com/>, acessado em novembro de 2010.
- ARIS Express 2.2. Software AG, IDS Scheer. <http://www.ariscommunity.com/aris-express>, acessado em novembro de 2010.
- ARIS Platform. Software AG, IDS Scheer. http://www.ids-scheer.com/en/ARIS_ARIS_Platform/3730.html, acessado em novembro de 2010.
- ASPECTC. <http://www.aspectc.org/>, acessado:em novembro de 2010.
- BARBOSA G., RABAÇA A. (2002). *Dicionário de Comunicação*. – 2ª. Ed. – Rio de Janeiro. ISBN 85-352-0854-2. Editora Campus.
- BATISTA T., CHAVEZ C., GARCIA A., SANT’ANNA C., KULESZA U. RASHID A., FILHO, F. F. (2006). ”Reflections on Architectural Connection: Seven Issues on Aspects and ADLs”. *Early Aspects at ICSE: 7th Workshop in Aspect-Oriented Requirements Engineering and Architecture Design*, Shangai - China. *ACM Proceedings of Early Aspects*. New York : ACM Press. p. 3-10.
- BPMN – Business Process Modeling Notation - <http://www.bpmn.org/>, acessado em novembro de 2010.
- CAPPELLI C., LEITE J., BATISTA T, SILVA L. (2009a). “An Aspect-Oriented Approach to Business Process Modeling”. *8th Early Aspects Workshop at AOSD*, Charllotesville.
- CAPPELLI C., SANTORO F. M., LEITE J. C., BATISTA, T. (2009b). *Applying the Aspect-Oriented Paradigm to Modularize Crosscutting Concerns in BPM*. WBPM, Fortaleza, Brasil.

- CAPPELLI C. (2009). Uma Abordagem para Transparência em Processos Organizacionais Utilizando Aspectos. Tese (Doutorado em Ciências - Informática), Cap. 4 — PUC-Rio, Rio de Janeiro, RJ, Brasil. Orientador: Julio Cesar Sampaio do Prado Leite.
- CAPPELLI C., SANTORO F. M., LEITE J. C., BATISTA T., MEDEIROS A. L., ROMEIRO C. S. C., (2010). "Reflections on the modularity of business process models: The case for introducing the aspect-oriented paradigm", Business Process Management Journal, Vol. 16 Iss: 4, pp.662 – 687.
- CHARFI A. e MEZINI, M. (2007). Aspect-Oriented Web Service Composition with AO4BPEL. World Wide Web, Vol. 10, Issue 3, pp. 309 – 344.
- CrossOryx. <http://www.ppgsc.ufrn.br/~analuisa/crossoryxeditor>, acessado em novembro de 2010.
- FERNANDES F. A., BATISTA T. V. (2004). AspectLua - uma extensão de Lua para Programação Orientada a Aspectos. Anais do II Workshop Técnico-Científico do DIMAp, pp.67-82, Natal – RN.
- FILMAN R. E., ELRAD T., CLARKE S., AKSIT M. (2005) Aspect-Oriented Software Development. Addison-Wesley, San Francisco.
- HILL J. B., CANTARA M., KERREMANS M., PLUMMER D. C. (2009). Magic Quadrant for Business Process Management Suites, Gartner.
- JAVA. Oracle. <http://www.java.com>, acessado em novembro de 2010.
- KELLER G., NÜTTGENS M., SCHEER A. W. (1992). Semantische Prozessmodellierung auf der Grundlage \Ereignisgesteuerter Prozessketten (EPK)", Technical Report 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany.
- KICZALES G., LAMPING J., MENDHEKAR A., MAEDA C., LOPES C. V., LOINGTIER J-M., IRWIN J. (1997). "Aspect-Oriented Programming". European Conference on Object-Oriented Programming (ECOOP), LNCS 1241, Springer, Finland.

- KICZALES G, HILSDALE E., HUGUNIN J., KERSTEN M., PALM J., GRISWOLD W. G. (2001) An Overview of AspectJ. In Proceedings of the 15th European Conference on Object-Oriented Programming. Springer-Verlag.
- LADDAD R. (2002). I want my AOP! Part 1. JavaWorld.com. Disponível em: http://www.javaworld.com/javaworld/jw-01-2002/jw-0118-aspect_p.html, acessado em novembro de 2010.
- MAGDALENO A. M. (2006). Explicitando a Colaboração em Organizações através da Modelagem de Processos de Negócio. Dissertação de Mestrado, NCE/UFRJ, Rio de Janeiro.
- PARK C., CHOI H-J., LEE D., KANG S., CHO H-K., SOHN J-C. (2007). Knowledge-Based AOP Framework for Business Rule Aspects in Business Process. ETRI Journal, vol. 29, n.4, pp. 477-488.
- RASHID A., MOREIRA A., ARAUJO J. (2003). “Modularisation and Composition of Aspectual Requirements”. AOSD '03 Proceedings of the 2nd international conference on Aspect-oriented software development. New York, NY, USA, 11-20.
- SHARP A. e MCDERMOTT P. (2001). “Workflow Modeling: Tools for Process Improvement and Application Development”, Artech House, Inc., Norwood, MA.
- SILVA L. (2006). “Uma Estratégia Orientada a Aspectos para Modelagem de Requisitos”. Rio de Janeiro. 220p. Tese de Doutorado em Engenharia de Software - PUC-Rio.
- TIRELO F., BIGONHA R. S., BIGONHA M. A. S., VALENTE M. T. O. (2004). Desenvolvimento de Software Orientado por Aspectos. Jornada de Atualização em Informática – JAI.
- Oryx. Business Process Technology, Hasso-Plattner-Institute, Universität Potsdam. <http://bpt.hpi.uni-potsdam.de/Oryx>, acessado em novembro de 2010.

- OSSHHER H., TARR P. (2000). Hyper/J: Multi-Dimensional Separation of Concerns for Java. In *Proceedings of the 22nd international conference on Software engineering*. ACM Press.
- SOARES D.C. (2008). An Aspect-Oriented Approach Model Business Process Crosscutting Concepts (in Portuguese). Undergraduate project, Department of Applied Informatics, UNIRIO, Brazil.
- THOMPSON S. e ODGERS B. (1999) Aspect-Oriented Process Engineering. In Proc. of the Workshop on Object-Oriented Technology in conjunction with ECOOP.
- VANDERFEESTEN I., REIJERS H. A., MENDLING J., van der Aalst, W.M.P., CARDOSO, J. (2006). On a Quest for Good Process Models: The Cross-Connectivity Metric, *Proceedings of International Conference on Cooperative Information Systems (CoopIS), LNCS*, Vol. 4275, pp. 183-200.
- WADA H., SUZUKI J., OBA K. (2008). Early Aspects fo Non-functional Properties in Service Oriented Business Processes. *IEEE Congress on Services*.