Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

TRATANDO VARIABILIDADE EM LINHA DE PROCESSOS DE NEGÓCIO: UMA ABORDAGEM COMPOSICIONAL

Idarlan Martins Machado

Brasília
2011

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

TRATANDO VARIABILIDADE EM LINHA DE PROCESSOS DE NEGÓCIO: UMA ABORDAGEM COMPOSICIONAL

Idarlan Martins Machado

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientador
Prof. Dr. Vander Alves

Coorientador
Prof. Dr. Rodrigo Bonifácio

Brasília
2011

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenador: Prof. Dr. Mauricio Ayala Rincón

Banca examinadora composta por:

Prof. Dr. Vander Alves (Orientador) — CIC/UnB
Prof.ª Dr.ª Maristela Holanda — CIC/UnB
Prof. Dr. Uirá Kulesza — DIMAp/UFRN

CIP — Catalogação Internacional na Publicação

Martins Machado, Idarlan.

TRATANDO VARIABILIDADE EM LINHA DE PROCESSOS DE NEGÓCIO: UMA ABORDAGEM COMPOSICIONAL / Idarlan Martins Machado. Brasília : UnB, 2011.
174 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2011.

1. Linha de Produtos de Software, 2. Modelagem de Processos de Negócio, 3. Notação de Modelagem de Processos de Negócio,
4. Linguagem de Modelagem de Processos Orientada a Aspectos,
5. Modularidade.

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

TRATANDO VARIABILIDADE EM LINHA DE PROCESSOS DE NEGÓCIO: UMA ABORDAGEM COMPOSICIONAL

Idarlan Martins Machado

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof. Dr. Vander Alves (Orientador)
CIC/UnB

Prof.ª Dr.ª Maristela Holanda
CIC/UnB

Prof. Dr. Uirá Kulesza
DIMAp/UFRN

Prof. Dr. Mauricio Ayala Rincón
Coordenador do Mestrado em Informática

Brasília, 15 de julho de 2011

Dedicatória

Aos meus pais, por todo sacrifício despendido para me educar nos caminhos corretos da vida. Por meio desse sacrifício pude experimentar o amor descrito na Primeira Carta aos Coríntios, Capítulo 13:

"Ainda que eu falasse as línguas dos homens e dos anjos, e não tivesse amor, seria como o metal que soa ou como o sino que tine. E ainda que tivesse o dom de profecia, e conhecesse todos os mistérios e toda a ciência, e ainda que tivesse toda a fé, de maneira tal que transportasse os montes, e não tivesse amor, nada seria. E ainda que distribuísse toda a minha fortuna para sustento dos pobres, e ainda que entregasse o meu corpo para ser queimado, e não tivesse amor, nada disso me aproveitaria. O amor é sofredor, é benigno; o amor não é invejoso; o amor não trata com leviandade, não se ensoberbece. Não se porta com indecência, não busca os seus interesses, não se irrita, não suspeita mal; Não folga com a injustiça, mas folga com a verdade; Tudo sofre, tudo crê, tudo espera, tudo suporta. O amor nunca falha; mas havendo profecias, serão aniquiladas; havendo línguas, cessarão; havendo ciência, desaparecerá; Porque, em parte, conhecemos, e em parte profetizamos; Mas, quando vier o que é perfeito, então o que o é em parte será aniquilado. Quando eu era menino, falava como menino, sentia como menino, discorria como menino, mas, logo que cheguei a ser homem, acabei com as coisas de menino. Porque agora vemos por espelho em enigma, mas então veremos face a face; agora conheço em parte, mas então conhecerei como também sou conhecido. Agora, pois, permanecem a fé, a esperança e o amor, estes três, mas o maior destes é o amor."

A minha esposa, Rianne, companheira em todo tempo. Sem ela, certamente, eu não teria cumprido esta empreitada. Obrigado amor, por todo apoio, paciência e estímulos constantes.

Ao meu filho, Bernardo, que nasceu justamente no início da construção desta obra. Tive a companhia dele (chorando ou se mexendo) nas madrugadas que enveredei trabalhando. Seu sorriso foi constante incentivo para eu continuar trabalhando.

Agradecimentos

Agradeço aos meus orientadores, Orientador Prof. Dr. Vander Alves e Coorientador Prof. Dr. Rodrigo Bonifácio, por todo apoio, paciência e orientação. Em especial, ao Prof. Dr. Vander pela constante supervisão. Certamente, após a orientação dos Senhores, eu sou uma pessoa melhor, principalmente intelectualmente e profissionalmente. Agradeço também aos demais docentes do Departamento de Informática que colaboraram por meio do ensino das disciplinas-base. Agradeço aos colegas do grupo de pesquisa, especialmente a Lucinéia Turnes, Giselle Machado e Vinícius. Vocês três contribuiram diretamente para conclusão desta dissertação. Dos três, destaco a ajuda imprescindível de Lucinéia, colega que se tornou amiga para toda a vida. Muito obrigado Lucinéia por tudo. Aos meus chefes diretos do Serviço Federal de Processamento de Dados - SERPRO - que flexibilizaram meu horário de trabalho para viabilizar minha dedicação ao mestrado, Márcia Porto, Nilson Costa e Anderson Soares.

Abstract

It has become increasingly important to perform modeling of business processes in midsize and large companies. As a result, there are several studies, methods, standards, best practices and tools offered by the market and the academy. Meanwhile, the recent research area of Software Product Line (SPL) aims at strategic reuse of software. This paradigm can be used at any stage of software engineering. As the requirements phase of software should be aligned with business processes, it is, in principle, possible to apply the principles of SPL also to that stage. This is advantageous because larger organizations frequently suffer from redundancy in processes and, consequently, in systems, since the latter are merely tools to support the former. The lack of generic standards in organizations for modeling business processes as well as the limitation of modeling languages of business processes with respect to modularity are issues that motivate this research. Therefore, this dissertation presents the modeling studies, organization and configuration of business processes in order to avoid repetition of processes and provide an extension an earlier approach designed to manage variability in scenarios of use cases using an Aspect Oriented approach - OA. Our work proposes an extension of this approach to manage the variability of business processes, thereby focusing on the strategic reuse of business processes.

Keywords: Software Products Line, Business Process Modeling, Business Process Modeling Notation, Aspect-Oriented Business Process Language, modularity.

Resumo

Torna-se cada vez mais importante usar modelagem de processos de negócios nas empresas de médio e grande porte. Como resultado, há vários estudos, métodos, padrões, melhores práticas e ferramentas disponibilizados pelo mercado e pela academia. Enquanto isso, a recente área de pesquisa de Linha de Produtos de Software (LPS) visa reuso estratégico de software. Esse paradigma pode ser utilizado em qualquer fase da Engenharia de Software. Como a fase de requisitos de software deve estar alinhada com processos de negócio, é, em princípio, possível aplicar os princípios da LPS também a essa fase. Isso é vantajoso porque organizações maiores frequentemente sofrem de redundância em processos e, consequentemente, em sistemas, já que esses últimos são meramente ferramentas para apoiar aqueles. A ausência de padrão genérico nas organizações para modelagem de processos de negócio bem como limitação das linguagens de modelagem de processos de negócio com relação à modularidade são problemas que motivam esta pesquisa. Portanto, esta dissertação apresenta estudos de modelagem, sistematização e parametrização de processos de negócio com o objetivo de evitar repetição de processos e fornecer extensão a uma abordagem existente para gerenciar variabilidade em cenários de casos de uso usando a Abordagem Orientada a Aspectos - OA. Nosso trabalho propõe uma extensão dessa abordagem para gerenciar as variabilidades de processos de negócio, visando assim o reuso estratégico de processos de negócio.

Palavras-chave: Linha de Produtos de Software, Modelagem de Processos de Negócio, Notação de Modelagem de Processos de Negócio, Linguagem de Modelagem de Processos Orientada a Aspectos, Modularidade.

Sumário

Lista de Figuras

Capítulo 1

Introdução

1.1 Contexto

Atualmente, há uma forte necessidade, especialmente nas organizações de médio e grande porte, de se concentrar na gerência de seus processos visando os seguintes objetivos de negócio Hammer e Champy (1994b) : (i) Fornecer uma visão clara dos processos realizados pela organização bem como a responsabilidade envolvida em cada um; (ii) Estabelecer metas e indicadores por processo; (iii) Melhorar a qualidade dos produtos e serviços; (iv) Racionalizar a utilização de recursos; (v) Identificar gargalos nos processos; (vi) Minimizar custos e maximizar lucros; (vii) Aumentar a competitividade; (viii) Explorar novos nichos de mercado.

Com o intuito de suprir essa demanda, o Gerenciamento de Processos de Negócio (GPN) *Business Process Management* tem se popularizado Harmon (2003). No entanto, o uso de GPN sem método e ferramentas adequadas pode gerar outros problemas, tais como: (i) a ausência de um padrão genérico de GPN nas organizações e (ii) a restrição de recursos nas linguagens de Modelagem de Processos de Negócio *(Business Process Modeling)* (BPM) em relação à modularidade em processos de negócio. Esses problemas, em grandes corporações, geram diversos transtornos e prejuízos, tais como Harmon (2003): (1) repetição de processos; (2) utilização de vocabulários distintos para expressar os mesmos conceitos; (3) retrabalho para atualizar os processos idênticos, porém repetidos; (4) desenvolvimento de diferentes soluções de software para apoiar os mesmos processos; (5) uso ineficiente de recursos organizacional. Esses tipos de problemas na administração pública federal são latentes, por exemplo:

(i) Cada órgão da administração pública realiza o processo de cadastro de suas unidades internas a sua maneira. Ou seja, não há um processo único de cadastro de Órgãos e unidades na esfera federal. Como consequência, em cada entidade, há replicação de recursos organizacional (pessoas, máquinas, sistemas) para executar um processo que é único em toda a administração;

(ii) A administração pública tem órgões com diferentes ramificações, por isso, é comum que esses ramos tenham sistemas, por exemplo, de folha de pagamento próprio, apesar de pertencerem à mesma organização. Apesar de cada ramo ter seus próprios processos particulares (devido a alguma legislação local), espera-se que a maioria

dos processos sejam semelhantes aos processos dos outros ramos. Essa replicação é claramente contra os objetivos de negócio mencionados.

1.2 Problema

Nesse contexto de redundância em processos de negócio, o problema central abordado nesta dissertação é o gerenciamento da variabilidade e comunalidade latentes nos processos de negócio das organizações, ou seja, como organizar comunalidade e variabilidade de forma a gerar produtos otimizando recursos da organização? Abordagens existentes não tratam esse problema com uma abordagem que explore o reuso estratégico de artefatos. Uma das abordagens Rosa et al. (2011) propõe um conjunto de métodos e ferramentas para o desenvolvimento de processos baseados no modelo de processo configurável utilizando conectores de decisões *(gateways)* para anotar as variabilidades que ocorrem em atividades e recursos organizacionais. Dessa forma, não fornece meios para separar os espaços de solução e configuração, impedindo a reutilização da variabilidade. A outra proposta consiste em uma abordagem orientada a aspectos para modularizar interesses transversais em processos de negócios. Uma extensão da Notação de Modelagem de Processos de Negócio (BPMN) é proposta para expressar interesses transversais através da inserção de processos transversais (representado por atividades) e as relações transversais (representada por fluxos entre as atividades), sem, no entanto, explorar a abordagem de Linha de Produtos de Software (LPS) *(Software Product Line)*.

1.3 Objetivos

Diante do problema identificado, objetivo central desta dissertação é: **Propor técnica e ferramenta que tornem possível o gerenciamento de comunalidade e variabilidade em processos de negócio**.

Como objetivos secundários vinculados a esse objetivo central, temos:

- Estender uma infraestrutura existente, *Hephaestus* Bonifácio e Borba (2009);
- Realizar uma avaliação qualitativa dessa proposta no domínio de Recursos Humanos (RH) da administração pública federal.

1.4 Organização do Texto

Este trabalho está estruturado como se segue. Iniciamos com a fundamentação teórica (Capítulo 2). Em seguida, apresentaremos nossa proposta para tratar variabilidades em processos de negócio (Capítulo 3). Uma avaliação de nossa proposta é realizada em processos de negócio do domínio de Recursos Humanos (RH) da administração federal (Capítulo 4). E, por fim, fazemos as considerações finais, apresentando os trabalhos relacionados e trabalhos futuros (Capítulo 5).

Capítulo 2

Fundamentação Teórica

As áreas de conhecimento envolvidas na proposta deste trabalho são as seguintes: 1) Linha de Produtos de Software, subárea da Engenharia de Software focada em reuso estratégico (Seção 2.1) Clements e Northrop (2002); 2) Modelagem de Processos de Negócio, que provê técnicas e melhores práticas para modelar os processos de negócio das organizações (Seção 2.2) Havey (2005); 3) Desenvolvimento de Software Orientado a Aspectos, que propõem técnicas para melhorar a modularidade dos programas (Seção 2.3) Kiczales et al. (1997) e 4) Ferramenta *Hephaestus* (Seção 2.4).

2.1 Linha de Produtos de Software

Uma Linha de Produtos de Software (LPS) é um conjunto de aplicações no mesmo domínio e que são desenvolvidas sistematicamente a partir de um conjunto comum de artefatos genéricos Clements e Northrop (2002). Os produtos ou instâncias da LPS são gerados pela composição e adaptação dos artefatos genéricos com artefatos específicos a cada produto. Os artefatos genéricos são os elementos comuns da linha de produtos e os artefatos específicos são as variações que ocorrem de um produto para o outro dessa linha de produtos.

Os benefícios potenciais de LPS incluem redução de tempo e custos de desenvolvimento e aumento da qualidade dos produtos. Em particular, isto implica que a qualidade dos artefatos genéricos e a qualidade dos artefatos específicos sejam combinadas de forma eficiente para garantir a qualidade de produtos na LPS. Os desafios centrais de LPS são o gerenciamento das variações dos artefatos e as estratégias de adoção Krueger (2002). O gerenciamento das variações diz respeito à representação das variações e sua composição com artefatos reutilizáveis para a geração dos produtos específicos. O desafio específico é como representar a variabilidade e a composição de forma modular, minimizando o esforço na composição. Tradicionalmente, as variações existem em diferentes níveis de abstração, correspondentes aos ciclos de vida do desenvolvimento do software, indo desde testes, implementação, arquitetura até requisitos. As estratégias de adoção dizem respeito a como adotar LPS em uma organização. No entanto, pode-se considerar que um nível de abstração mais elevado do que requisitos é constituído por processos de negócio da organização Hammer e Champy (1994a).

Para viabilizar o tratamento de variabilidade de uma Linha de Produtos (LP), independentemente do tipo de artefato que compõe essa LP, a LPS preconiza alguns artefatos

que fazem parte da engenharia do domínio e da aplicação Czarnecki e Eisenecker (2000). Nesse contexto, os artefatos dessa primeira engenharia são: Modelo de Features *(Feature Model)* (FM) e a Configuração do Conhecimento *(Configuration Knowledge)* (CK). Já os da segunda engenharia são: o Modelo *(Model)*, que contém os artefatos da LP, e a Configuração do Produto *(Configuration Knowledge)* (PC). No FM estão contidas as *features* da LP, que podem ser obrigatórias (devem estar presentes em qualquer produto da LP gerado) e as opcionais (escolhidas para estarem ou não em um produto). O CK permite o mapeamento dessas *features* para funções (transformações) responsáveis por resolver as variabilidades dos artefatos vinculados às *features* de modo a gerar produto final. Já o Modelo representa os ativos da organização *(core assets)*, artefatos comuns e variáveis da LP, que se deseja instanciar gerando assim o produto. Para viabilizar isso, é necessário selecionar as *features* da LP que se deseja no produto. Esse conjunto de *features* escolhidas representam o PC.

A Engenharia de Requisitos inclui o processo de descobrir, analisar, documentar, e verificar as funções e restrições do sistema. De fato, a elicitação e a análise de requisitos devem ser compatíveis com os objetivos de negócio da organização, o que implica necessariamente a aderência aos processos delas. Em particular, variações em nível de requisitos estão ligadas a variações em processos de negócio. Desta forma, variabilidade em processo de negócio rege variabilidade em software. Por exemplo, na Administração Pública, *licitação* pode ser visto como um processo de negócio genérico, que precisa ser estendido para diferentes licitações particulares, que por sua vez podem requerer software de apoio também específico, levando, pois, a uma LPS de software regida por um processo de negócio.

2.2 Modelagem de Processos de Negócio

A responsabilidade de gerenciar uma organização se resume, basicamente, em controlar a alocação dos recursos às atividades que apoiam o negócio central da entidade. Nesse contexto, a modelagem de processo de negócio é "uma abordagem sistemática e estruturada para analisar, melhorar, controlar e gerenciar processos com o objetivo de melhorar a qualidade dos produtos e serviços" Gurp et al. (2001). Na tentativa de contribuir com uma notação padrão para BPM, a Iniciativa de Gerenciamento de Processos de Negócio criou a notação padrão BPMN e o *Object Management Group* (OMG) a mantém. Segundo esse grupo Obj (2009), a BPMN "é uma notação gráfica que retrata os passos de em um processo de negócio. BPMN representa o caminho completo do fluxo de um processo de negócio. A notação foi especialmente elaborada para coordenar a sequência de processos e as mensagens que fluem entre diferentes processos participantes em um conjunto de atividades relatado".

Os processos de negócios são uma forma de uma entidade organizar o trabalho e os recursos (pessoas, equipamentos, informações, e assim por diante) para realizar os seus objectivos Dumas et al. (2005). Estes processos especificam as principais atividades, papéis e artefatos produzidos de forma específica por uma organização. Algumas atividades são realizadas manualmente e outras podem ser automatizadas através do desenvolvimento de um ou mais sistemas. Tais processos são essencias para atingir os objetivos de negócio e estar em conformidade com eles é uma importante medida de maturidade organizacional Jeston e Nelis (2006). No entanto, foi relatado que é frequente o caso que a replicação

das atividades ou até mesmo de todo um processo ocorrem e falha na identificação de tais replicações resulta em desnecessários custos organizacionais, independentemente da qualidade do software que apoiam tais processos Rosa et al. (2011). Por exemplo, em uma organização com ramificações diferentes pode acontecer que esses ramos tenham seus sistemas de folha de pagamento próprio, apesar de pertencer à mesma organização. Apesar de cada ramo ter seus próprios processos particulares (devido a alguma legislação local), espera-se que a maioria dos processos seja semelhante aos processos dos outros ramos. Essa replicação é claramente contra os objetivos de negócio (por exemplo, eficiência, maximização da alocação de recursos, entre outros). Para minimizar esse risco e maximizar a alocação de recursos organizacionais, é importante estar ciente dessa replicação e, em seguida, lidar com tais comunalidades e variabilidades.

2.3 Abordagem Orientada a Aspectos

O Desenvolvimento de Software Orientado a Aspectos (DSOA) Kiczales et al. (2001b,a) surgiu da necessidade de melhorar a modularidade das aplicações, de modo a melhorar a *separação de interesses*. O termo interesses vem do inglês *concerns* e diz respeito a requisitos, funcionais ou não, que são relevantes e precisam estar presentes nos sistemas.

O conceito de separação de interesses envolve quebrar o esforço de desenvolvimento de um programa em partes, o que muitas vezes se reflete na quebra do programa em módulos que se sobrepõem em funcionalidade o mínimo possível. Os interesses que não se encaixam em formas de encapsulamento de paradigmas tradicionais de desenvolvimento, por exemplo, paradigmas funcional ou orientado a objetos, são denominados interesses transversais (*crosscutting concerns*). Exemplos desses casos seriam requisitos não funcionais como *logging*, persistência, *debugging* e gerenciamento de exceções, além de alguns requisitos funcionais.

Quando uma porção razoável de linhas de código de um método ou classe está relacionada a interesses transversais, o código se torna disperso e entrelaçado: disperso porque interesses estão espalhados por meio de diversas partes do programa; entrelaçado porque uma parte do programa pode envolver diversos interesses e o entendimento e modificação do código torna-se oneroso, já que se deve entender e considerar todos esses interesses. O DSOA provê técnicas, método e ferramentas para auxiliar o desenvolvedor na tarefa de identificação, separação, representação e composição de interesses transversais (aspectos) modularizando-os de forma a que eles se sobreponham em funcionalidade o mínimo possível em diferentes níveis de abstração (código, arquitetura, requisitos, processos de negócio). Por exemplo, AspectJ é uma linguagem de programação em que os aspectos *(aspects)* podem ter um ou mais *advice(s)* e esses pode(m) estar(em) associado(s) a um ou mais ponto-de-conte *(pointcut)*. Portanto, uma forma possível de se realizar a modularização visando a realização da composição é: (i) modelar os interesses transversais em *aspects*; (ii) definir os *(advices)*; (iii) definir os pontos-de-corte. Dessa forma, a composição dos aspectos com os artefatos de diferentes níveis de abstração da aplicação será viabilizada por meio dos pontos-de-corte inteceptando os pontos variáveis - pontos-de-junção *(joinpoints)* - dos artefatos. Os pontos-de-junção podem ter sido definidos previamente (anotando manualmente os pontos-de-junção), quanto se está trabalhando com pontos-de-corte semânticos (inspirados em *annotation-based approach*), ou de forma automática

(viabilizando *obliviousness* Filman e Friedman (2000)), quando se está trabalhando com pontos-de-corte sintáticos.

2.4 *Hephaestus*

Hephaestus é uma ferramenta construída na linguagem de programação funcional Haskell. Essa ferramenta atualmente provê uma infraestrutura de funcionalidades para tratar variabilidades em cenários de caso de uso. A abordagem utilizada na implementação dessa ferramenta baseia-se na abordagem paramétrica e composicional baseada no paradigma Orientação a Aspectos (OA) Bonifácio e Borba (2009). Além da utilização dos conceitos da AO, esta ferramenta também implementa conceitos da LPS.

A Figura 2.4 ilustra a abordagem implementada por essa ferramenta. O primeiro parâmetro de entrada, SPL *Use Case Model (SPLUCM)*, representa a Linha de Produtos de Casos de Uso de um domínio de negócio específico. O segundo parâmetro de entrada, *Feature Model* (FM), representa o conjunto de características (*features*) do domínio de negócio dos casos de uso da SPLUCM. Portanto, os casos de uso contidos na SPLUCM referenciam as *features* desse segundo parâmetro. Isso permite então que uma *feature* possa representar um ou mais casos de uso. O terceiro parâmetro, *Product Configuration* (PC), é um subconjunto do FM. O PC representa quais *features* do FM que devem estar contidas no produto final gerado pela ferramenta *Hephaestus*. O quarto e último parâmetro de entrada, *Configuration Knowledge* (CK), provê o mapeamento das *features* escolhidas no PC para as transformações que devem ser executadas pela ferramenta. Essas transformações são funções implementadas em Haskell e nelas estão implementadas a lógica referente ao tratamento das variabilidades em cenários de caso de uso. Portanto, o passo denominado *weaving process* (WP) representa a execução dessas funções de transformação.

Logo, como saída, a ferramenta gera, após a execução do WP, o produto final com as variabilidades resolvidas (ou parcialmente resolvidas). Esse resultado é ilustrado pela imagem *Product Specific Use Case Model* (PSUCM) da Figura 2.4. Detalhamento mais o núcleo do *Hephaestus*, a Figura 2.4 ilustra o trecho de código que essa ferramenta irá executar para realizar o WP.

Na Figura 2.4, na seção *"Abstract syntax"*, vemos a definição da estrutura de dados do CK. Essa estrutura em Haskell é denominada tipo algébrico de dados. Ela é composta por uma lista de *ConfigurationItem*. E, por sua vez, por meio da definição *"**data** ConfigurationItem..."*, temos que um *ConfigurationItem* é composto por uma *FeatureExpression* e por uma lista de transformações, *"[Transformation]"*. E, por sua vez, temos que uma transformação, *"Transformation"*, é definida como *"**type** Transformation = SPL -> Product -> Product"*. Isso representa a definição genérica de uma função cujos parâmetros de entrada são "SPL" e *"Product"* e o parâmetro de saída é *"Product"*. Essa é uma definição genérica para as funções de transformação utilizadas pelo *Hephaestus*. Isso indica que, dado um *Software Product Line* - SPL, e dado um produto dessa SPL, segundo parâmetro, *"Product"*, temos como saída esse produto transformado (com variabilidades parcialmente ou totalmente resolvidas), representado pelo terceiro parâmetro, *"Product"*. Portanto, por meio dessa estrutura, *Configuration Knowledge*, é possível saber quais funções de transformação utilizar dando como entrada para essa tabela às características desejadas (que estão contidas na configuração do produto). Logo, a segunda seção da Figura 2.4, *"Interpreter-based semantics"*, sem detalhar cada linha, apresentando apenas a funcionalidade intuitiva,

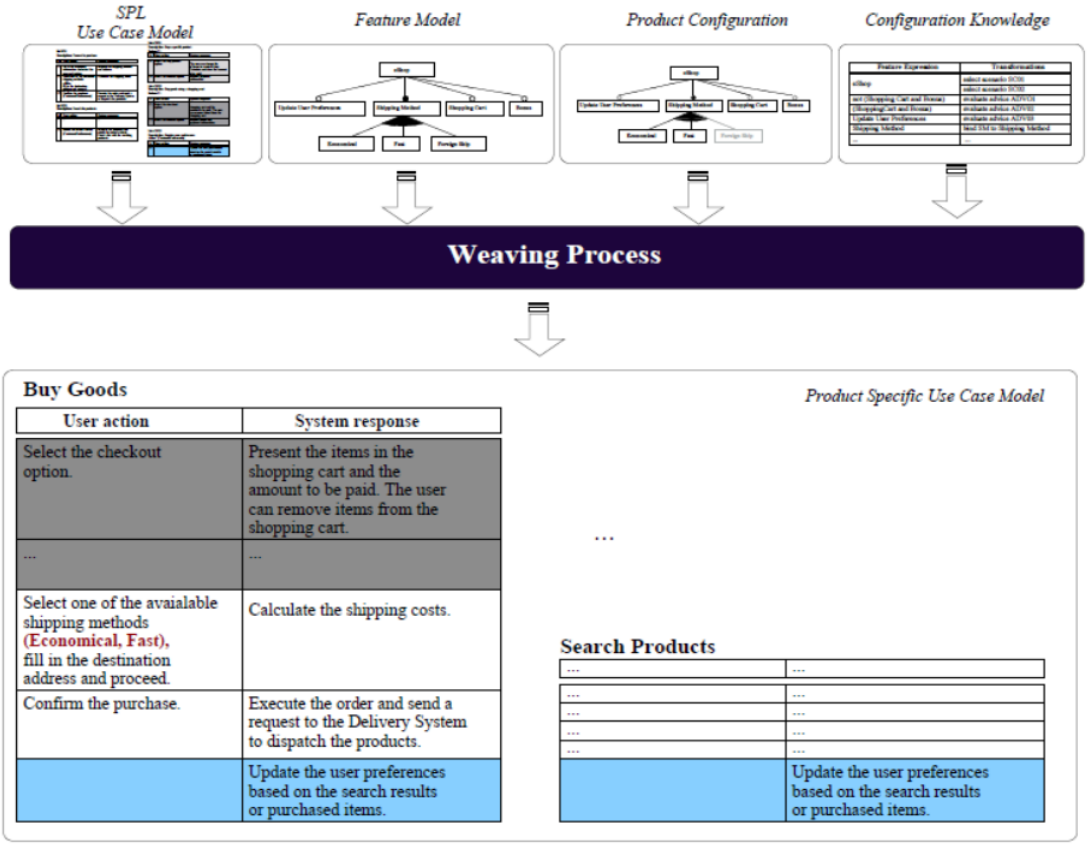


Figura 2.1: Weaving Process Bonifácio e Borba (2009)

Abstract syntax

```
type ConfigurationKnowledge = [ConfigurationItem]
data ConfigurationItem = ConfigurationItem {
  expression :: FeatureExpression,
  transformations :: [Transformation]
}
type Transformation = SPL → Product → Product
```

Interpreter-based semantics

```
weavingProcess fm pc ck spl = refine ts spl newProduct
  where
    ts = concat [transformations c | c ← ck, eval pc (expression c)]
    newProduct = ...
    refine [] pla product = product
    refine (t : ts) spl product = refine ts spl (t spl product)
```

Figura 2.2: Contexto da Extensão Proposta Bonifácio e Borba (2009)

significa que serão executadas, em um dado produto, as funções de transformação do CK que forem correspondentes às expressões de características (*feature expressions*) do CK que casarem (forem iguais) às expressões de características da configuração do produto. Quando essa lista de transformação estiver vazia, indica que se executou todas as funções de transformação, resta então, retornar o produto, último parâmetro da função *refine*, com suas variabilidades totalmente ou parcialmente resolvidas.

Capítulo 3

Tratando Variabilidade em Processos de Negócio com uma Abordagem Composicional

Diante do exposto no Capíutlo 2, Seção 2.2, sobre redundância em processos de negócio, faz-se necessário ter técnicas e ferramentas que tornem possível resolver variabilidades em processos de negócio de modo a auxiliar as organizações na maximização de resultados tendo em vista o aumento da qualidade no fornecimento de produtos e serviços em menor custo e prazo. Portanto, neste capítulo, propomos técnica e extensão da ferramenta *Hephaestus* que permite o tratamento dessas variabilidades. Essa proposta é baseada em uma abordagem paramétrica e composicional com OA. Ela aproveita uma infraestrutura existente e a estende fornecendo novas transformações, modelagem de artefatos relevantes (processos de negócios) e sua variabilidade, e um mapeamento de novos conhecimentos de configuração apresentam expressões para essas novas transformações. Iniciamos este capítulo apresentando uma visão geral do método proposto (Seção 3.1) para, em seguida, apresentarmos detalhamento do mesmo e extensão da ferramenta *Hephaestus* (Seção 3.2).

3.1 Método Utilizado

Nesta seção apresentamos o método utilizado para gerar produtos de processos de negócio. Na Figura 3.1 apresentamos a visão estática da abordagem proposta (adaptação da Figura 2.4). Já na Figura 3.2 é apresentada uma visão dinâmica dos passos seguidos para chegarmos a produtos finais de processos de negócio utilizando essa abordagem. Iniciamos esta seção explicando como fizemos para elaborar o (FM) (Subseção 3.1.1); em seguida, descrevemos a fase de modelagem dos processos de negócio comuns e variantes na notação BPMN (Subseção 3.1.2); explicamos como esses processos nessa notação foram representados nos tipos algébricos propostos (Subseção 3.1.3); informamos a etapa de elaboração do CK (Subseção 3.1.4); a fase de elaboração das configurações do produto (Subseção 3.1.5); por fim, fornecendo todos esses artefatos como entrada, mostramos como geramos os produtos de processos de negócio (Subseção 3.1.6).

Antes de detalharmos cada subseção, é importante explicar o significado do *gateway* da Figura 3.2, representando paralelismo, logo ao iniciarmos este método. Esse paralelismo ocorre por causa do processo extrativo adotado. Conforme é preconizado em LPS, a

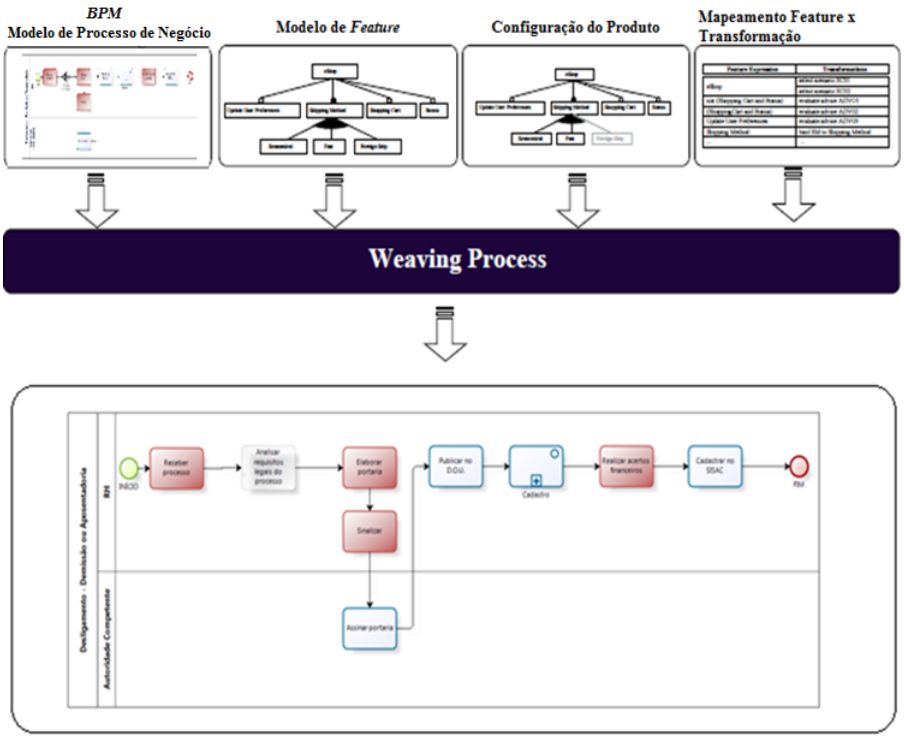


Figura 3.1: Weaving Process para BPM - Visão Estática

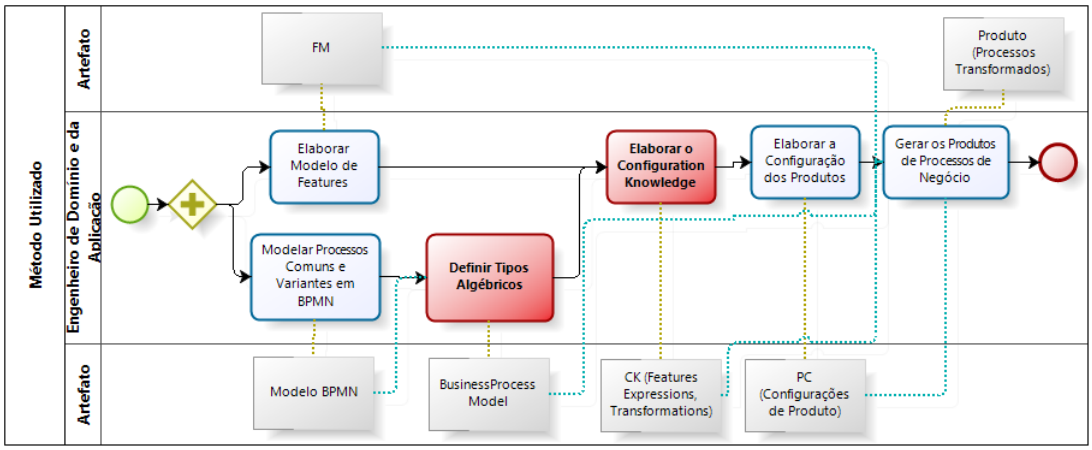


Figura 3.2: Método Utilizado - Visão Dinâmica

adoção extrativa é escolhida quando a organização já possuí, por exemplo, um conjunto de aplicações e deseja torná-las uma LPS Krueger (2001). Além disso, as atividades "Definir Tipos Algébricos" e "Elaborar o Configuration Knowledge", destacadas na ilustração do método, representam as principais contribuições do método proposto e são descritas na Seção 3.2.

3.1.1 Elaborar Modelo de *Features*

A partir da análise dos processos de negócio fornecidos como entrada, construiremos, desempenhando o papel de Engenheiro de Domínio, o FM do domínio de RH. Uma instância de tal modelo é um conjunto de *features* selecionadas que, por sua vez, tal conjunto corresponde a um conjunto de processos de negócios destinados para uma determinada organização. Por exemplo, no Brasil, duas configurações possíveis poderiam ser Servidor Público (vinculado à Lei n º 8.112) e os regimes da Consolidação das Leis Trabalhistas - CLT (mais comum nas organizações privadas). Nessa última configuração, por exemplo, a licença para casamento é de três dias, enquanto que para a anterior, essa mesma licença é de oito dias. Outras configurações são possíveis para uma grande quantidade de organizações públicas e privadas específicas.

3.1.2 Modelar Processos Comuns e Variantes em BPMN

A partir de processos de negócio fornecidos, realizaremos, desempenhando o papel de Engenheiro da Aplicação, uma análise manual para identificar as comunalidades e variabilidades entre os processos de negócio. Essa análise compreenderá em identificar atividades e outros objetos *(gateways, messsage flows, entre outros)* comuns. Após identificadas as comunalidades e variabilidades, modelaremos as variabilidades em processos a parte para que possamos utilizar a abordagem proposta, paramétrica e composicional com OA. Nessa abordagem, as variabilidades ficam separadas das comunalidades para que possamos melhorar a modularização à medida que diminuimos o espalhamento e o entrelaçamento de tais variabilidades com as comunalidades.

3.1.3 Definir Tipos Algébricos

Com a extensão da BPMN proposta na Seção 3.2.1, modelaremos, desempenhando o papel engenheiro da aplicação, os processos comuns e variantes para os tipos algébricos propostos.

3.1.4 Elaborar o *Configuration Knowledge*

Com base nas *features* identificadas no FM e nas suas relações (obrigatórias, opcionais, *or-inclusive*, *or-exclusive*), elaboraremos/preencheremos, desempenhando o papel de Engenheiro de Domínio, o artefato reutilizável *Cofiguration Knowledge - CK* da infraestrutura *Hephaestus* para viabilizar a execução das transformações propostas (extensão de *Hephaestus* proposta) já que esse artefato (CK) mapea as expressões de *features* (selecionadas na configuração do produto) para as suas transformações. As transformações e o CK são refinados na Seção 3.2.2.

3.1.5 Elaborar a Configuração dos Produtos

Elaboraremos, desempenhando o papel de Engenheiro da Aplicação, as configurações de produto (PCs) para gerarmos o produto de processo(s) de negócio desejado. Isto é feito criando uma instância do FM.

3.1.6 Gerar os Produtos de Processos de Negócio

Este passo refere-se à obtenção dos modelos de processos de negócio transformados após a aplicação das funções de transformação explicadas na Seção 3.2.2. Para fazer isso, conforme ilustrado na Figura 3.1, é necessário fornecer como entrada para a infraestrutura *Hephaestus* os artefatos BPM, FM, PC e CK.

3.2 Tratando Variabilidade em Processos de Negócio Utilizando uma Abordagem Composicional

Nesta seção, apresentaremos o cerne de nossa abordagem para tratar variabilidade em processos de negócio Machado et al. (2011). Ela é construída sobre o modelo de variabilidade de *Modeling Scenario Variability as Crosscutting Mechanisms* MSVCM Bonifácio e Borba (2009), que também fornece um conjunto de bibliotecas Haskell Jones et al. (2002) e ferramentas para desenvolvimento de linha de produtos (chamado Hehaestus), e, portanto, caracteriza-se pelo seguinte: a) considera a contribuição dos conhecimentos de configuração (*configuration knowledge*), um modelo voltado para relacionar recursos às transformações que resolvem as variabilidades de uma Linha de Processos de Negócio (LPN); b) usa parametrização e construções orientadas a aspectos, levando a uma especificação modular dos ativos (bens/artefatos da organização) essenciais (núcleo) e variantes de um domínio de negócio. Seguindo a orientação do método proposto na Seção 3.1, esta seção detalha o passo descrito na Seção 3.1.3 e na Seção 3.1.4.

Iniciamos esta seção descrevendo uma extensão da notação BPMN para representar variabilidade em LPN. Essa extensão apresenta uma noção de aspectos de processo de negócio, que visa modularizar variabilidade em processos de negócio. Os tipos algébricos propostos na Subseção 3.2.1 são uma implementação dessa extensão. Depois disso, na Subseção 3.2.2, detalhamos a customização realizada no CK bem como apresentamos as transformações propostas para tratar variabilidade em LPN. Por fim, na Seção 3.2.3, descrevemos as decisões de projeto quanto aos tipos algébricos e boa-formação *(well typed)* dos processos.

3.2.1 Definir Tipos Algébricos

Para viabilizar o tratamento de variabilidade em uma LPN, nós estendemos os elementos principais da BPMN, de tal forma que possamos representar a parte variante de um processo principalmente usando a noção de parametrização e composição de *advice*-ponto de corte. Para implementar essa extensão, usamos a linguagem de programação funcional Haskell, a mesma linguagem usada para implementar Hehaestus. Essa decisão leva a uma definição concisa da fase de derivação do produto, como explicamos na Subseção 3.2.2. Nós estendemos os elementos da BPMN por meio de uma linguagem que incorpora um pequeno

domínio específico Hudak (1996), que compreende um conjunto de tipos algébricos e operadores que um analista pode usar para instanciar processos de negócios usando Haskell como uma linguagem hospedeira.

Na Listagem 3.1, apresentamos a definição dos tipos algébricos que estendem a BPMN. Nesta listagem temos um conjunto de processos de negócio (linhas 1–2), que forma o modelo de processos de negócio, *BusinessProcessModel*. Nesse modelo há processos (linhas 3–9) de dois tipos. Essa noção de tipo de processo (ou um processo base ou *advice*) é a nossa principal extensão para BPMN (linha 6 e linhas 10–12). Normalmente, em nossa abordagem, nós modelamos comunalidades utilizando processos de negócios base, enquanto que representamos variabilidade de fluxo utilizando processos do tipo de *advice* (que está mais especializado como *before, after* ou *around* - linha 21 da Listagem 3.1). Cada processo do tipo *advice* tem um atributo ponto-de-corte *(pointcut)* que identifica os locais dentro do modelo de processo de negócio onde a composição de aspecto (*aspect*) deve ocorrer (linhas 12 e 22 da Listagem 3.1). Também estendemos a BPMN para que pudéssemos atribuir anotações (linha 17) para o tipo de dados *FlowObject* (linhas 13–19), o que pode ser usado para representar atividades *(Activity)* BPMN ou *gateways* (linha 20), a fim de expor pontos-de-junção *(joinpoints)*, pontos de variabilidade da LPN, ao conjunto de BPMN do tipo *advice*. Outra extensão a BPMN, implementada também no *FlowObject*, é a lista de parâmetros (linha 18 da Listagem 3.1). Por fim, o fluxo dos processos é representado por meio de uma lista de transições (linhas 8 e 23). Cada transição (linha 23) é formada por três elementos, objeto de origem *(source)*, de destino *(target)* e uma condição.

```
1 data BusinessProcessModel =
2  BPM { processes :: [BusinessProcess] }
3 data BusinessProcess =
4  BusinessProcess {
5     pid :: Id,
6     ptype :: ProcessType,
7     objects :: [FlowObject],
8     transitions :: [Transition]
9 }
10 data ProcessType =
11  BasicProcess |
12  Advice {advType :: AdviceType, pc :: Pointcut}
13 data FlowObject =
14  FlowObject {
15     fId :: Id,
16     fType :: FlowObjectType,
17     annotations :: [Annotation],
18     parameters :: [Parameter]
19 } | Start | End | Proceed
20 data FlowObjectType = Activity | Gateway
21 data AdviceType = Before | After | Around
22 data Pointcut = Pointcut String | PCut String [String]
23 type Transition =  (FlowObject, FlowObject, Condition)
```

Listing 3.1: Sintaxe abstrata extendida da BPMN

3.2.2 *Configuration Knowledge* e Transformações

O conhecimento de configuração (CK) é um ativo específico da abordagem que relaciona expressões de *features* às transformações do modelo. Em mais detalhes, o CK corresponde a uma lista de itens de configuração, que representa um mapeamento das expressões de *features* às transformações (Listagem 3.2, linhas 2-5). Se uma expressão de *feature* é satisfeita por um membro da LPS, as transformações relacionadas são aplicadas. Desta forma, o conjunto de transformações apropriadas é responsável pela geração automática de um produto específico. Para adaptar o CK para o domínio de processos de negócios, tivemos que mudar a assinatura do tipo algébrico *Transformation*.

Como mostrado na Listagem 3.2, linha 6, uma transformação é uma função que espera dois argumentos finais (um argumento *SPL*, que representa os ativos da LPS, no nosso caso, uma LPN (do tipo *BusinessProcessModel*), e um argumento *Product* que representa um produto em uma fase específica do processo de derivação (também do tipo *BusinessProcessModel*) e retorna uma versão refinada do produto com alguma variabilidade resolvida. Os tipos algébricos *SPL* (linhas 7–10) e *Product* (linhas 11–14) de *Hephaestus* também foram adaptados de modo que passassem a referenciar (linha 9 e 13, respectivamente) o novo tipo algébrico *BusinessProcessModel*. O tipo algébrico *SPL*, como se trata de um artefato da engenharia do domínio Czarnecki e Eisenecker (2000), faz referência (linha 8) ao artefato, também desse mesmo domínio, *FeatureModel*. Já o tipo algébrico *Product*, por fazer parte da engenharia da aplicação Czarnecki e Eisenecker (2000), faz referência (linha 12) ao artefato, também desse mesmo domínio, *ProductConfiguration*. Ressaltamos que essas adaptações nos tipos algébricos no *Hephaestus* bem como integração das transformações propostas foram realizadas por uma colega. Por isso, as funções apresentadas nesta seção estão na versão anterior da integração a essa infraestrutura.

```
1 type ConfigurationKnowledge = [ConfigurationKnowledgeItem]
2 data ConfigurationKnowledgeItem = ConfigurationItem {
3   expression :: FeatureExpression,
4   Transformations :: [Transformation]
5 }
6 type Transformation = SPL → Product → Product
7 data SPL = SPL {
8   fm :: FeatureModel,
9   splBpm :: BusenessProcessModel
10 }
11 data Product = Product {
12   pc :: ProductConfiguration,
13   productBpm :: BusinnessProcessModel
14 }
```

Listing 3.2: Extensão do Configuration Knowledge.

Portanto, nossa representação CK mapeia expressões de *features* para as transformações que podem selecionar ou configurar os processos de negócios para uma configuração de produto específico. Diferentes transformações lidam com diferentes tipos de variabilidades. No entanto, uma vez que estas funções são adaptadas, a sua aplicação parcial leva a funções que obedecem à assinatura de *Transformation*.

As transformações mencionadas são agrupadas nas seguintes categorias:

(a) **Seleção de Processo de Negócio:** Como mostramos no trecho de código abaixo, a

transformação *selectBusinessProcess* primeiro cria uma lista de processos *(bps)* a partir dos processos da LPN (*spl*) cujos identificadores são iguais ao processo informado (*bpId* - primeiro argumento). Depois disso, essa transformação concatena os processos selecionados (*bps*) com os processos já selecionados no produto. Caso o processo informado não exista na LPN, a lista *bps* será vazia e será retornado o mesmo produto (product) informado como entrada.

```
1 selectBusinessProcess :: Id → BusinessProcessModel → BusinessProcessModel →
      BusinessProcessModel
2 selectBusinessProcess bpId spl product =
3  let bps = [bp
4             | bp ∈ (processes spl)
5             , pid bp == bpId]
6  in  product {productBpm = [bps] ∪ (processes product)}
```

(b) **Vincular parâmetro:** O trecho seguinte de código detalha a transformação *bindParameter*. Ela aplica a função auxiliar *bindParameter'* em todos os processos do produto. Essa função por sua vez chama a função auxiliar *applyParam* para cada objeto do processo *(bp)*. Na linha 8 dessa função, o(s) parâmetro(s) do(s) objeto(s) avaliados (*(parameters fo)*) cujo nome *(i)* tenha casado (*np == i*) com o parâmetro informado *(np)* é(são) remontado(s), realizada a vinculação *(np,vp)*. Na linha 9, é mantido o estado dos parâmetro(s) do(s) objeto(s) onde não tenha ocorrido o casamento (*np /= i*).

```
1 bindParameter :: Name → Value → BusinessProcessModel → BusinessProcessModel
2 bindParameter np vp product =
3  BPM ( map (bindParameter' np vp) (processes product) )
4  where
5   bindParameter' np vp bp =
6     bp {objects = (map (applyParm np vp) (objects bp))}
7   applyParm np vp fo =
8     fo {parameters' = ([(np,vp) | (i,j) ∈ (parameters fo), np == i ] ∪
9                        [(i,j)    | (i,j) ∈ (parameters fo), np /= i ])}
```

(c) **Avaliar *Advice*:** A transformação *evaluateAdvice* primeiro obtém uma lista de *advices* (*advs*), cujos identificadores são iguais ao *advId*. No primeiro caso, o modelo de processo de negócio resultante do produto é processado pela aplicação da função *eval()* para todos os processos do produto. No segundo caso (lista vazia), nós apenas retornamos o produto sem aplicar qualquer transformação. A função *eval* apenas chama a avaliação adequada de *advice before* (Listagem 3.3), *after* (Listagem 3.6) ou *around* (Listagem 3.7).

```
1 evaluateAdvice :: Id → BusinessProcessModel → BusinessProcessModel →
      BusinessProcessModel
2 evaluateAdvice advId spl product =
3  let advs = [a | a ∈ (processes spl)
4             , (pid a) == advId]
5  in case advs of
6   [adv] → product {productBpm = wovenProcesses }
7   [] → product
8  where wovenProcesses =
9   map (eval adv) (processes product)
10
11 eval adv bp =
12  in case ptype adv of
13    BasicProcess       → bp
14    (Advice After  _) → evaluateAfterAdvice adv bp
15    (Advice Before _) → evaluateBeforeAdvice adv bp
16    (Advice Around _) → evaluateAroundAdvice adv bp
```

A seguir, detalharemos e explicaremos as funções chamadas pela transformação *evaluateAdvice* (item c supracitado). Essas implementações são apresentadas nas Listagens 3.3, 3.6 e 3.7.

```
1 evaluateBeforeAdvice :: BusinessProcess → BusinessProcess → BusinessProcess
2 evaluateBeforeAdvice adv bp =
3  bp {
4   objects = nub ((objects bp) ∪ (objects adv)),
5   transitions = ([(i,j,k) | (i,j,k) ∈ (transitions bp), not (matches j k (pointcut adv
        ))] ∪
6                 [(i,y,k) | (i,j,k) ∈ (transitions bp), (x,y,z) ∈ startTransitions
                      adv, matches j k (pointcut adv)] ∪
7                 [(x,j,z) | (i,j,k) ∈ (transitions bp), (x,y,z) ∈ endTransitions adv,
                       matches j k (pointcut adv)] ∪
8                 [(x,y,z) | (x,y,z) ∈ (transitions adv), (x, y, z) ∉ ((
                      startTransitions adv) ∪ (endTransitions adv))])
9  }
```

Listing 3.3: Semântica operacional para avaliação de composição do tipo before.

A função *evaluateBeforeAdvice* (Listagem 3.3) resolve a variabilidade em processos de negócio base da linha de processos inserindo o *advice* (processo de negócio variante) antes do objeto de fluxo desses processos de negócio base, conforme ilustrado na Figura 3.3. Para fazer isso, é necessário que ocorra o casamento do ponto de corte *(pointcut)* do *advice* com o(s) objeto(s) de fluxo anotado(s) *(joinpoint(s))* (Figura 3.3-a, objeto B) do processo base (Figura 3.3-a) e a condição do ponto de corte (caso ele tenha sido criado pelo construtor *PCut*) case com a condição (terceiro item) da transição do processo base.

Nas linhas 1-2 da Listagem 3.3, temos que essa função recebe dois parâmetros. O primeiro é o processo de negócio do *advice* (Figura 3.3-b) e o segundo é o processo de negócio base (Figura 3.3-a).

O retorno dessa função é o processo-base *(bp)* que estará sendo modificado a partir da linha 3 (Listagem 3.3). Ela retorna o processo de negócio base com alguma variabilidade resolvida (Figura 3.3-c). Na linha 3 está-se mantendo os mesmos valores do *bp* de entrada (recurso de *Haskell* para evitar que se tenha que construir o tipo algébrico novamente).

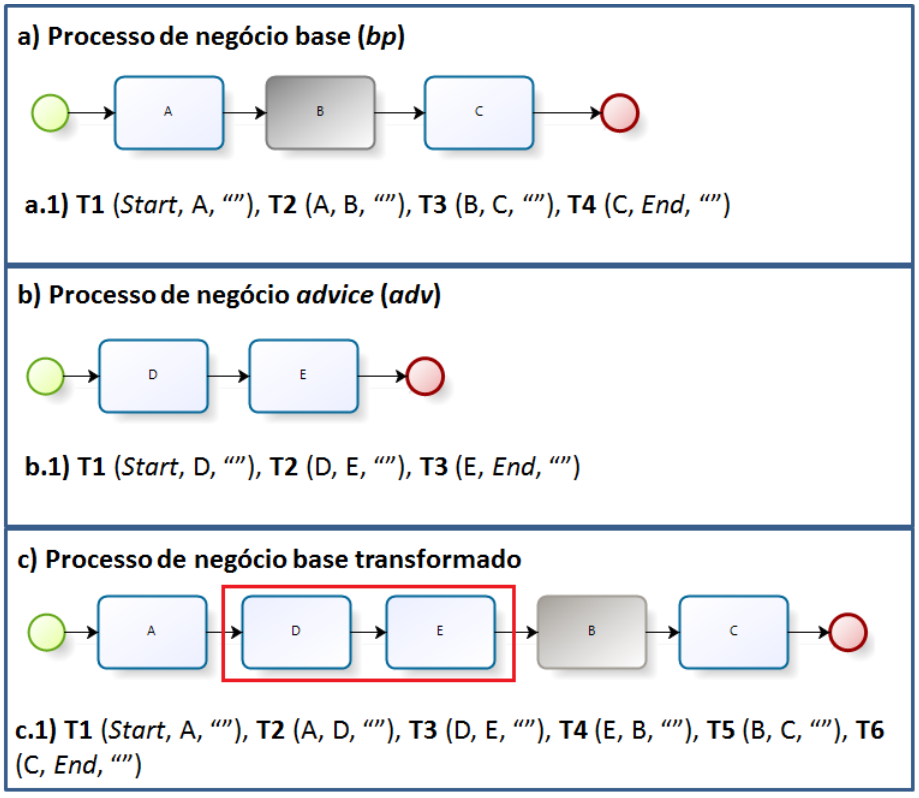


Figura 3.3: Ilustrar Execução Incremental da Função evaluateBeforeAdvice

Na linha 4, é iniciado o processo de composição do processo base com o *advice*. Como o *advice* do tipo *before* realiza inclusão do *advice* no processo base, no contexto da lista de objetos do processo base, basta realizar a união dos objetos do processo base com os objetos do *advice*. Para eliminar os objetos repetidos após a união entre esses dois conjuntos, utilizamos a função *nub* da infraestrutura de bibliotecas da linguagem Haskell. Há dois objetos dos processos base e dos *advices* que são comuns, são eles: o *Start* e o *End*. Portanto, após a composição realizada nessa linha 4, a lista de objetos do processo base retornado deve conter apenas um elemento *Start* e *End*. A função *nub* (linha 4) então garante essa unicidade.

O restante do código, da linha 5 à linha 9 (explicadas de forma incremental a seguir), representa a parte central da composição do processo do tipo *advice* (Figura 3.3-b) com o processo base (Figura 3.3-a). A lista de transições de um processo de negócio representa o fluxo entre os objetos, como ilustrado por Figura 3.3 - a.1, b.1 e c.1 (os identificadores **T*x1*, ..., T*xn*** dessas listas de transições são utilizados aqui para fins de clareza). O que estamos fazendo nesse trecho de código é inserindo as transições do *advice* na lista de transições do processo base, realizando assim a composição do *advice* com o processo base.

A linha 5 da Listagem 3.3 seleciona as transições do processo base (Figura 3.3-a.1) onde não haja casamento do ponto de corte (*pointcut*) do *advice* com qualquer objeto anotado *(joinpoint)* do processo base e a condição desse ponto de corte (caso o ponto de corte tenha sido criado pelo construtor *PCut*) não case com a condição (contida em k) da transição do *bp* . Portanto, as transições resultantes, ilustradas em Figura 3.3-c.1, serão:

T1(*Start*, A, “”), **T5**(B, C, “”) e **T6**(C, *End*, “”). A transição **T2** da lista Figura 3.3-a.1 não foi selecionada porque o seu segundo objeto (item j da tripla (i,j,k)) casa com o ponto de corte (*pointcut*) do *advice*. Essa transição **T2** será o foco da próxima linha da função, linha 6.

A linha 6 da Listagem 3.3 cria a transição **T2**(A, D, “”) ilustrada em Figura 3.3-c.1. Para fazer isso, os seguintes passos são necessários: (i) selecionar, da lista de transições do processo base (Figura 3.3-a.1), a(s) transição(ões) cujo ponto de corte do *advice* case(m) com o objeto j anotado *joinpoint* e a condição desse ponto de corte (caso o ponto de corte tenha sido criado pelo construtor *PCut*) casem com a condição (contida em k) da transição, no caso, será **T2**(A, B, “”); (ii) selecionar, da lista de transições do processo *advice* (Figura 3.3-b.1), a transição cujo objeto x seja o objeto de início (*Start*), no caso, será **T1**(*Start*, D, “”); (iii) o resultado final dessa linha 6 é compor essas duas transições em uma única, no caso, o objeto i de **T2**(A, D, “”) (objeto A), com o objeto y da transição **T1**(*Start*, D, “”) (objeto D), portanto, resultando em **T2**(A, D, “”) da Figura 3.3-c.1.

A linha 7 da Listagem 3.3 cria a transição **T4**(E, B, “”) ilustrada em Figura 3.3-c.1. Para fazer isso, os seguintes passos são necessários: (i) selecionar, da lista de transições do processo base (Figura 3.3-a.1), a transição cujo ponto de corte do *advice* casem com o objeto j anotado *joinpoint* e a condição desse ponto de corte (caso o ponto de corte tenha sido criado pelo construtor *PCut*) casem com a condição (contida em k) da transição, no caso, será **T2**(A, B, “”); (ii) selecionar, da lista de transições do processo *advice* (Figura 3.3-b.1), a transição cujo objeto y seja o objeto de fim (*End*), no caso, será **T3**(E, *End*, “”); (iii) o resultado final dessa linha 7 é compor essas duas transições em uma única, no caso, o objeto x de **T3**(E, *End*, “”) (objeto E) com o objeto j da transição **T2**(A, B, “”) (objeto B), portanto, resultando em **T4**(E, B) da Figura 3.3-c.1.

A linha 8 da Listagem 3.3 cria a última transição, **T3**(D, E, “”) ilustrada em Figura 3.3-c.1. Para fazer isso, os seguintes passos são necessários: (i) selecionar, da lista de transições do processo *advice* (Figura 3.3-b.1), a transição(ões) cujo objeto x não seja o objeto de início (*Start*) e o objeto y não seja o objeto fim (*End*), no caso, a única transição que se enquadra nesses critérios é **T2**(D, E, “”); (ii) o resultado final dessa linha 8 é adicionar essa transição na lista de transições transformada, portanto, resultando em **T3**(D, E, “”) da Figura 3.3-c.1.

```
1 startTransitions :: BusinessProcess → [Transition]
2 startTransitions bp = [(i,j,k) | (i,j,k) ∈ (transitions bp), i == Start]
3 endTransitions :: BusinessProcess → [Transition]
4 endTransitions bp = [(i,j,k) | (i,j,k) ∈ (transitions bp), j == End]
```

Listing 3.4: Funções auxiliaries.

A função de casamento (Listagem 3.5) verifica se um dado objeto *(FlowObject)* corresponde à cláusula ponto-de-corte *(pointcut)* de um *advice (ADV)* (quando o construtor usado para instanciar o *pointcut* foi “Pointcut” - linha 2) e verifica, na transição que esse objeto faz parte, se a condição dessa transição casa com alguma condição da lista de condições do *pointcut* (construtor “PCut” - linha 3). Neste caso, os objetos e transições *adv* são introduzidas no processo de negócio (uma semântica operacional desta composição nas Listagens 3.3, 3.6 e 3.7).

```
1 matches :: FlowObject → Condition → Pointcut → Bool
2 matches f _ (Pointcut pc) = pc ∈ (annotations f)
3 matches f c (PCut pc cs) = (pc ∈ (annotations f)) && (c ∈ cs)
```

Listing 3.5: Uma função que faz casamento de pontos de corte e anotação baseado em pontos de junção e condição de *gateways*

A função *evaluateAfterAdvice* (Listagem 3.6) resolve a variabilidade em processos de negócio base da linha de processos inserindo o *advice* (processo de negócio variante) após o objeto de fluxo desses processos de negócio base, conforme ilustrado na Figura 3.4. Para fazer isso, é necessário que ocorra o casamento do ponto de corte *(pointcut)* do *advice* (Figura 3.4-b) com o(s) objeto(s) de fluxo anotado(s) *(joinpoint(s))* (Figura 3.4-a, objeto B) do processo base (Figura 3.4-a) e a condição desse ponto de corte (caso o ponto de corte tenha sido criado pelo construtor *PCut*) case com a condição (terceiro item) da transição.

```
1 evaluateAfterAdvice :: BusinessProcess → BusinessProcess → BusinessProcess
2 evaluateAfterAdvice adv bp =
3  bp {
4   objects = nub ((objects bp) ∪ (objects adv)),
5   transitions = ([(i,j,k) | (i,j,k) ∈ (transitions bp), not (matches i k (pointcut adv
        ))] ∪
6                 [(i,y,k) | (i,j,k) ∈ (transitions bp), (x,y,z) ∈ startTransitions
                      adv, matches i k (pointcut adv)] ∪
7                 [(x,j,z) | (i,j,k) ∈ (transitions bp), (x,y,z) ∈ endTransitions adv,
                       matches i k (pointcut adv)] ∪
8                 [(x,y,z) | (x,y,z) ∈ (transitions adv), (x, y, z) ∉ ((
                      startTransitions adv) ∪ (endTransitions adv))])
9  }
```

Listing 3.6: Semântica operacional para avaliação de composição do tipo after.

Nas linhas 1-2 da Listagem 3.6, temos que essa função recebe dois parâmetros. O primeiro é o processo de negócio do *advice* (Figura 3.4-b) e o segundo é o processo de negócio base (Figura 3.4-a).

O retorno dessa função (Listagem 3.6) é montado pelas linhas 3-9. Ela retorna o processo de negócio base com alguma variabilidade resolvida (Figura 3.4-c). A linha 4 dessa função é idêntica à linha 4 da função explicada anteriormente (*evaluateBeforeAdvice*), portanto, dispensa explicação.

O restante do código, da linha 5 à linha 9, da Listagem 3.6 representa a parte central da composição do processo do tipo *advice* (Figura 3.4-b) com o processo base (Figura 3.4-a). A lista de transições de um processo de negócio representa o fluxo entre os objetos, como ilustrado por Figura 3.4 - a.1, b.1 e c.1. Portanto, o que estamos fazendo nesse trecho de código é inserindo as transições do *advice* na lista de transições do processo base, realizando assim a composição do *advice* com o processo base.

A linha 5 da Listagem 3.6 seleciona as transições do processo base (Figura 3.4-a.1) onde não haja casamento do ponto de corte *(pointcut)* do advice com qualquer objeto anotado do processo *(joinpoint)* base e a condição desse ponto de corte (caso o ponto de corte tenha sido criado pelo construtor *PCut*) não case com a condição (contida em k) da transição do *bp* . Portanto, as transições resultantes, ilustradas em Figura 3.4-c.1, serão: **T1**(*Start*, A, “”), T2(A, B, “”) e T6(C, *End*, “”). A transição **T3** da lista Figura 3.4-a.1

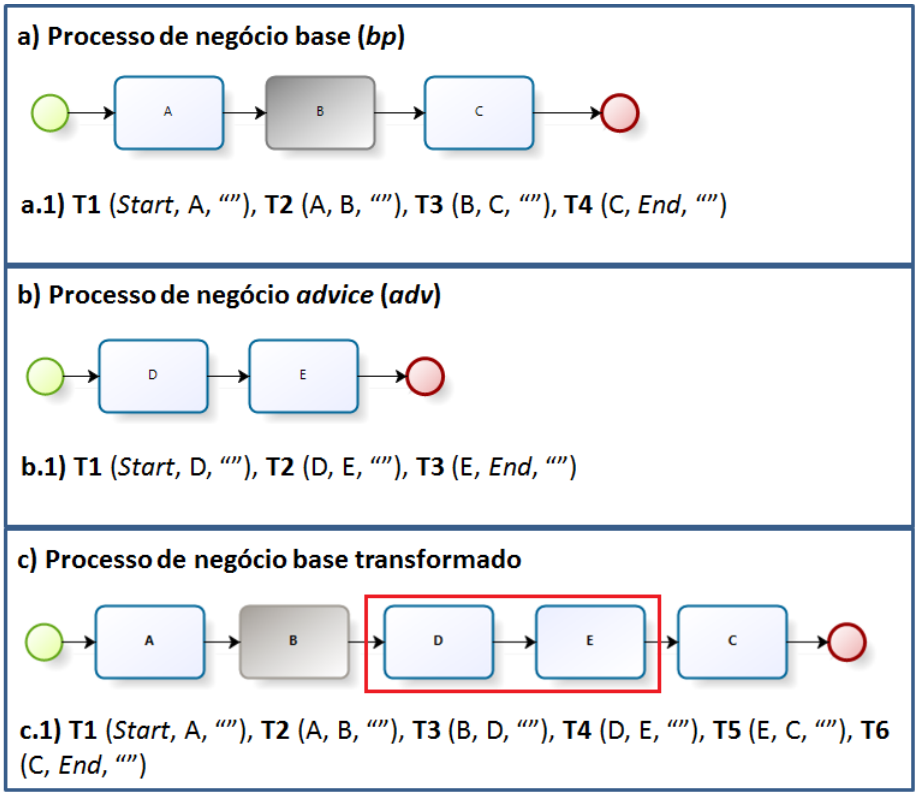


Figura 3.4: Ilustrar Execução Incremental da Função evaluateAfterAdvice

não foi selecionada porque o seu segundo objeto (item i da tripla (i,j,k)) casa com o ponto de corte (*pointcut*) do *advice*. Essa transição **T3** será o foco da próxima linha, linha 6 da Listagem 3.6.

A linha 6 do código a seguir criam a transição **T3**(B, D) ilustrada na Figura 3.4-c.1. Para fazer isso, os seguintes passos são necessários: (i) selecionar, da lista de transições do processo base (Figura 3.4-a.1), a transição cujo ponto de corte do *advice* case com o objeto j anotado *joinpoint* e a condição desse ponto de corte (caso o ponto de corte tenha sido criado pelo construtor *PCut*) case com a condição (contida em k) da transição, no caso, será **T3**(B, C, “”); (ii) selecionar, da lista de transições do processo *advice* (Figura 3.4-b.1), a transição cujo objeto x seja o objeto de início (*Start*), no caso, será **T1**(*Start*, D, “”); (iii) o resultado final dessa linha 6 é compor essas duas transições em uma única, no caso, o objeto i de **T3**(B, D, “”) (objeto B), com o objeto y da transição **T1**(*Start*, D, “”) (objeto D), portanto, resultando em **T3**(B, D, “”) da Figura 3.4-c.1.

A linha 7 da Listagem 3.6 cria a transição **T5**(E, C, “”) ilustrada em Figura 3.4-c.1. Para fazer isso, os seguintes passos são necessários: (i) selecionar, da lista de transições do processo base (Figura 3.4-a.1), a transição cujo ponto de corte do *advice* case com o objeto j anotado *joinpoint* e a condição desse ponto de corte (caso o ponto de corte tenha sido criado pelo construtor *PCut*) case com a condição (contida em k) da transição, no caso, será **T3**(B, C, “”); (ii) selecionar da lista de transições do processo *advice* (Figura 3.4-b.1) a transição cujo objeto y seja o objeto de fim (*End*), no caso, será **T3**(E, *End*, “”); (iii) o resultado final dessa linha 7 é compor essas duas transições em uma única, no caso, o objeto x de **T3**(E, *End*, “”) (objeto E), com o objeto j da transição **T3**(B, C, “”) (objeto

C), portanto, resultando em **T5**(E, C, “”) da Figura 3.4-c.1.

A linha 8 da Listagem 3.6 cria a última transição, **T4**(D, E, “”) ilustrada em Figura 3.4-c.1. Para fazer isso, os seguintes passos são necessários: (i) selecionar, da lista de transições do processo advice (Figura 3.4-b.1), a(s) transição(ões) cujo objeto x não seja o objeto de início (*Start*) e o objeto y não seja o objeto fim (*End*), no caso, a única transição que se enquadra nesses critérios é **T2**(D, E, “”); (ii) o resultado final dessa linha 8 é adicionar essa transição na lista de transições transformada , portanto, resultando em **T4**(D, E, “”) da Figura 3.4-c.1.

A função *evaluateAroundAdvice* (Listagem 3.7) resolve a variabilidade em processos de negócio base da linha de processos inserindo o *advice* (processo de negócio variante) de duas formas alternativas:

(i) Quando há o objeto de fluxo *Proceed* no processo *advice*, o *advice* é inserido antes do objeto anotado (ponto de junção) do processo base e o objeto *Proceed* do *advice* é substituído pelo objeto anotado do processo base, ou seja, conforme ilustrado na Figura 3.5, essa função se comporta similarmente à função *evaluateBeforeAdvice* quando o *advice* do tipo *around* contiver em seu processo o objeto *Proceed*. Isto está implementado nas linhas 10–16.

(ii) Quando não há o objeto de fluxo *Proceed* no processo *advice*, o *advice* é inserido antes do objeto anotado (ponto de junção) do processo base e a ausência do objeto *Proceed* do *advice* indica que objeto anotado do processo base deve ser suprimido. A diferença deste item para o item (i) é que ocorre uma supressão do objeto anotado no processo base transformado (Figura 3.6-(c)). Isto está implementado nas linhas 17–21.

```
1 evaluateAroundAdvice :: BusinessProcess → BusinessProcess → BusinessProcess
2 evaluateAroundAdvice adv bp =
3  let objAnnot = [obj | obj ∈ (objects bp), matches obj "" (pointcut adv)];
4      transitionsTemp = ([(i,j,k) | (i,j,k) ∈ (transitions bp), not (matches i k (
           pointcut adv))] ∪
5                         [(i,y,k) | (i,j,k) ∈ (transitions bp), (x,y,z) ∈
                              startTransitions adv, matches j k (pointcut adv)] ∪
6                         [(x,j,z) | (i,j,k) ∈ (transitions bp), (x,y,z) ∈
                              endTransitions adv, matches i k (pointcut adv)] ∪
7                         [(x,y,z) | (x,y,z) ∈ (transitions adv), (x, y, z) ∉ ((
                              startTransitions adv) ∪ (endTransitions adv))]);
8      objProceed       = [ obj | obj ∈ (objects adv), obj == Proceed ]
9  in case objProceed of
10     — com Proceed
11     [objProceed] → bp {
12                        objects = (objects bp) ∪ [obj|obj ∈ (objects adv), not (obj
                              ∈ [Start, End, Proceed]) ],
13                        transitions = ([(i,obj,k) | (i,j,k) ∈ transitionsTemp, obj ∈
                               objAnnot, j == Proceed ] ∪
14                                       [(obj,j,k) | (i,j,k) ∈ transitionsTemp, obj ∈
                                            objAnnot, i == Proceed ] ∪
15                                       [(i,j,k) | (i,j,k) ∈ transitionsTemp, i /=
                                            Proceed, j /= Proceed ])
16                       }
17     — sem Proceed
18     otherwise → bp {
19                       objects = [obj|obj ∈ (objects bp), obj ∉ objAnnot] ∪ [obj|obj
                             ∈ (objects adv), not (obj ∈ [Start, End]) ],
20                       transitions = [(i,j,k) | (i,j,k) ∈ transitionsTemp]
21                      }
```

Listing 3.7: Um interpretador baseado em semântica operacional para avaliação de composição do tipo around.

3.2.3 Decisões de Projeto e Boa-Formação dos Processos

Nesta seção, descreveremos as decisões de projeto quanto à elaboração dos tipos algébricos propostos (Seção 3.2.1) e às transformações propostas (Seção 3.2.2) para gerarem processos bem formados *(well typed)*. Para subsidiar a elaboração dos tipos algébricos, inicialmente fizemos uma análise da sintaxe abstrata da notação BPMN Obj (2009) e selecionamos os elementos principais dessa notação para representar processos de negócio. Para iniciarmos a construção das transformações, realizamos uma análise prévia dos processos de negócio do domínio de RH (escolhido para a avaliação, Capítulo 4) para termos uma visão global dos tipos de transformações que deveriam ser contempladas e quais funcionalidades elas deveriam ter para gerar processos bem formados.

Ao analisarmos a sintaxe abstrata da BPMN e os processos de negócio do domínio de RH, percebemos que poderíamos representar uma boa parte dos elementos essenciais dos processos de negócio modelados nessa notação apenas com os elementos *Start, Activity, Gateway e End*, classificados nessa notação como *FlowObject*, e com as transições (*FlowObject* de origem*(source)*, *FlowObject* alvo*(target)*, condição) para representar o fluxo dos objetos, denominadas na BPMN de *ConnectionObject* (por simplificação didática, no-

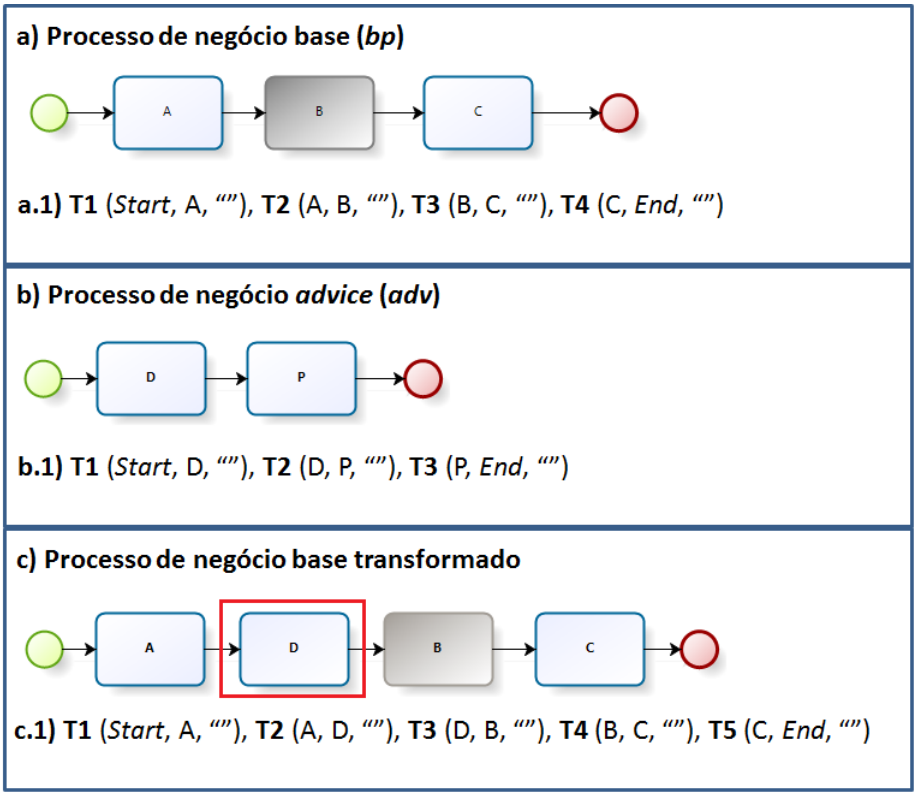


Figura 3.5: Ilustrar Execução Incremental da Função evaluateAroundAdvice - Utilizando Proceed

meamos como *transition*) em nossa proposta. Diante disso, inferimos que a estrutura principal do tipo algébrico *BusinessProcess* seria composta por uma lista de transições (*transitions*). Mas, decidimos também, que uma lista de objetos deveria compor esse tipo algébrico para podermos tratar a variabilidade de valores (campo *parameters* do *FlowObject*) que podem ocorrer somente nos objetos *FlowObject*. Implementamos controle nas transformaçãoes para manter a consistência entre essas duas listas (*objects* e *transitions*) ao realizar a(s) composição(ões), conforme mostrado na Seção 3.2.2. Esse controle foi implementado quando usamos a função *nub* na linha 3 das funções *evaluateBeforeAdvice* (Listagem 3.3) e *evaluateAfterAdvice* (Listagem 3.6) e quando usamos filtros (*“not (obj ‘elem‘ [Start, End, Proceed])”*, na linha 12, e *”not (obj ‘elem‘ [Start, End])”*, na linha 19) na função *evaluateAroundAdvice* (Listagem 3.7). No entanto, faz parte de trabalhos futuros a implementação de verificador de tipos para garantir essa consistência.

Por uma questão de uniformidade, simplificação e boa-formação, decidimos que os processos variantes (*advices*) teriam a mesma estrutura definida no *BusinessProcess*. Por isso, adicionamos o campo (*field*) *ptype* (do tipo *ProcessType*) nesse tipo algébrico para diferenciar os processos de negócio base (item terminativo *BasicProcess* do tipo algébrico *ProcessType*) dos variantes (tipo algébrico *advice* criado pelo construtor *Advice* do tipo *ProcessType*). Dessa forma, teremos um trecho de processo de negócio bem formado nos *advices*.

Para manter a boa-formação do processo de negócio após a transformação (composição do *advice* com o processo base), os elementos terminais (*Start*, *End* e *Proceed*) do *FlowOject* contidos no processo do *advice* devem ser retirados (viabilizado pelo controle

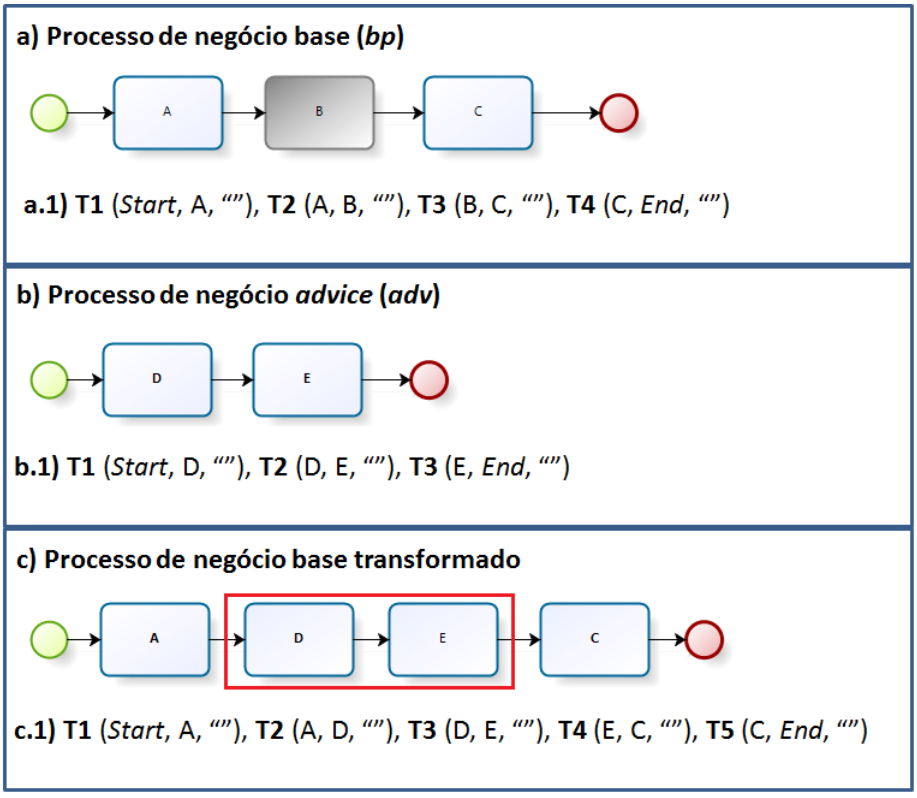


Figura 3.6: Ilustrar Execução Incremental da Função evaluateAroundAdvice - Sem Proceed

informado para manter a consistência entre as listas *objects* e *transitions* do *BusinessProcess*).

Para viabilizar a simplificação do *advice* e a boa-formação após sua composição com o(s) processo(s) de negócio(s) base, foi imposta a restrição na elaboração dos *advices* de haver somente um ponto de início *(Start)* e de fim *(End)* no processo do *advice*. Essa restrição é para garantir a generalidade do *advice* para qualquer processo de negócio base (que pode ter diversos tipos de transições chegando no(s) objeto(s) anotado(s) - *joinpoint(s)*). Faz parte de trabalhos futuros a implementação de um verificador de tipos para impor essa restrição. Portanto, o uso desses dois elementos (*Start* e *End*), nos processos variantes *(advices)*, além do papel de (a) manter a uniformidade com os processos de negócio base, (b) manter a simplicidade e (c) manter a boa-formação após as composições, é também de (d) servirem como pontos de referência para vincular, ao realizar uma composição do *advice* com a base, o processo variante ao processo base, conforme explicado na Seção 3.2.2 (no momento em que as funções auxiliares (*startTransitions* e *endTransitions*) da Listagem 3.4 são chamadas pelas transformações).

Quanto ao último item terminativo do tipo algébrico *FlowObject*, *Proceed*, ele pode estar presente somente nos processos do tipo (*ProcessType*) *Advice* e deve estar ligado às transições, ou seja, deve ter transição chegando nele e saindo dele. A verificação de tipo para este caso também faz parte de trabalhos futuros. Ele (o *Proceed*) serve para expressar a ligação de um objeto do *advice* às transições do objeto anotado do processo base, onde o fluxo foi interrompido, quando realizamos a composição, conforme explicado na Seção 3.2.2 (explicação do código das linhas 20 a 35 da função *evaluateAroundAdvice*).

Com essa separação entre processos de negócio base e variantes (ambos contidos em *BusinessProcessModel* quando instanciado), a granularidade do tratamento das variações é fina Kästner et al. (2008), pois tratamos variabilidade em nível de objetos *(FlowObjects)* bem como em nível de parâmetros contidos nos objetos.

O Desenvolvimento de Software Orientado a Aspectos (DSOA) preconiza a modularização de interesses como aspectos (*aspects*). Os aspectos podem ter um ou mais *advice(s)* e esses pode(m) estar(em) associado(s) a um ou mais *pointcut(s)*. Em nossa abordagem, como decidimos que um *advice* afeta um ou mais processo(s) de negócio base a cada chamada da transformação *evaluateAdvice*, decidimos nomear os interesses transversais (aspectos) apenas como *Advice*. Por isso, no tipo algébrico *ProcessType*, há o construtor denominado *Advice* que serve para criar a estrutura do tipo algébrico *Advice advType :: AdviceType, pc :: Pointcut*. Esse tipo algébrico é composto pelos campos: (i) *advType :: AdviceType*, que indica o tipo do *advice* (*AdviceType = Before | After | Around*); (ii) *pc :: Pointcut*, que (ii.a) representa o ponto de corte (*Pointcut String*) que casa com as transições *sem* condição definida (quando o texto do terceiro parâmetro da transição é igual à "") cujo um dos objetos (*FlowObject* de origem ou *FlowObject* alvo) esteja anotado; ou (ii.b) representa o ponto de corte *PCut String [String]* que casa com a(s) transição(ões) *com* condição definida cujo um dos objetos esteja anotado.

Para termos *pointcuts* mais robustos (não frágeis Störzer e Koppen (2004)), escolhemos *pointcuts* semânticos (inspirados em *Annotation-based Pointcuts*) Kulesza et al. (2006b). Esse tipo de *pointcut* tem a vantagem de ser difícil de "quebrar"[1], mas a desvantagem de ser invasivo, visto que temos que anotar, manualmente, o(s) ponto(s) variante(s) do(s) processo(s) de negócio base. Por esse processo ser manual, essa desvantagem se estende em não permitir automação da anotação. Devido essa escolha, criamos a lista de anotações *(annotations :: [Annotation])* no *FlowObject*. Criamos uma lista de anotações para permitir que diferentes *advices* possam ser aplicados ao mesmo objeto anotado *(joinpoint)*. Com o compartilhamento da estrutura do processo de negócio base com a estrutura do processo variante *(advice)*, permitimos que o *advice* também possa ter objetos anotados. Com isso, viabilizamos a composição de um *advice* com outro *advice* ou com ele mesmo. A outra alternativa seria implementar *pointcuts* sintáticos (implementados em Cappelli et al. (2009)), mas não o escolhemos devido à sua fragilidade. Esse tipo de ponto de corte não é invasivo, já que não exige anotação manual nos processos de negócio base. Portanto, como vantagem, eles viabilizam a automação da anotação. No entanto, sua desvantagem é de ser frágil.

Por fim, para permitirmos o tratamento de variabilidade de valor(es), criamos o campo *parameters :: [Parameter]* no *FlowObject*. Trata-se de uma lista do tipo *Parameter* (*type Parameter = (Name, Value)*), tupla formada por nome(String) e valor (tipo algébrico *data Value = Unbound | Value String*).

3.2.4 Variabilidade em Outros Objetos

Nesta seção ilustramos a aplicação das transformações propostas (Seção 3.2.2) quando outro tipo de *FlowObject* for anotado no(s) processo(s)-base, o *gateway*. Antes de tudo, temos de separar o comportamento comum e comportamento variante dos processos de

[1] "Quebramento" de *pointcut(s)* - quando deixa(m) de casar (ser(em) igual(is)) com seu(s) *joinpoint(s)* em caso de um refatoramento no(s) processo(s)-base, por exemplo.

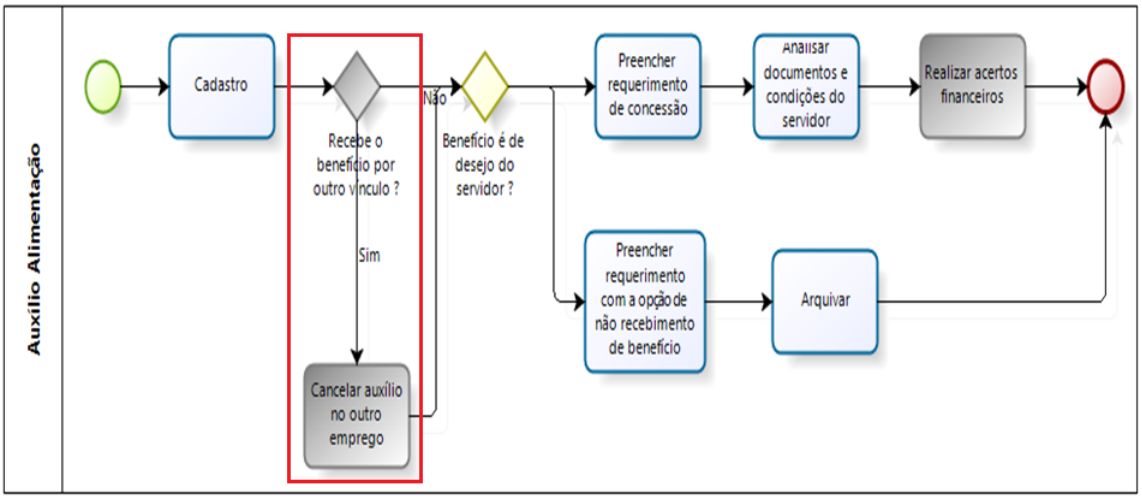


Figura 3.7: Processo Original - Auxílio Alimentação - Com Parte Variante Destacada

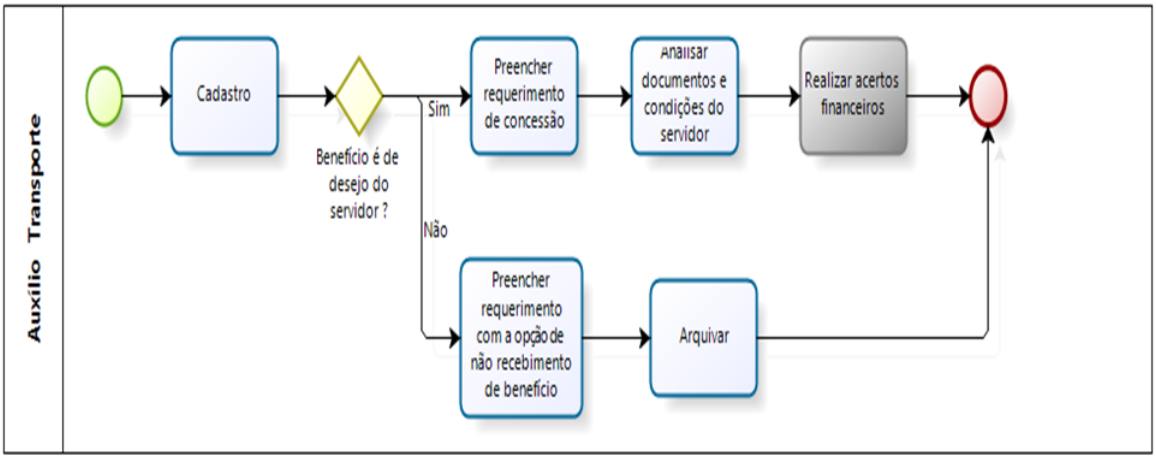


Figura 3.8: Processo Original - Auxílio Transporte

negócio apresentados nas Figuras 3.7 e 3.8. Nós criamos: (a) um processo de negócio base com os objetos de fluxo e transições compartilhados (Figura 3.9-(a)); (b) um *advice* formado pelos ativos (artefatos) variantes, neste caso, um *gateway* adicional e uma nova atividade (Figura 3.9-(b)).

Em seguida, para permitir a configuração dos processos de negócios comuns, temos que relacionar a expressão *Auxilio* com a transformação *selectBusinessProcess "bpAuxílioComum"*, onde *"bpAuxílioComum"* é o identificador dos processos de negócios que lida com os aspectos comuns entre os processos de *auxílio*. De um modo semelhante, a fim de avaliar o *advice* Auxílio Alimentação (Figura 3.9-(b)), temos que relacionar expressão de (*feature*) *Auxílio Alimentação* para a transformação *evaluateAdvice "advAuxílioAlimentação"*, onde *"advAuxílioAlimentação"* é o identificador do *advice* Auxílio Alimentação contido nos ativos da LPS. Figura 3.9-(c) mostra um fragmento do CK com essas transformações.

Portanto, o processo de derivação do produto aplica as transformações que estão relacionadas com as expressões válidas para uma configuração de produto. Por exemplo, se um produto é configurado com as *features* Auxílio e Auxílio Alimentação, o produto será composto por um processo de negócios formado pela composição dos processos de negócios

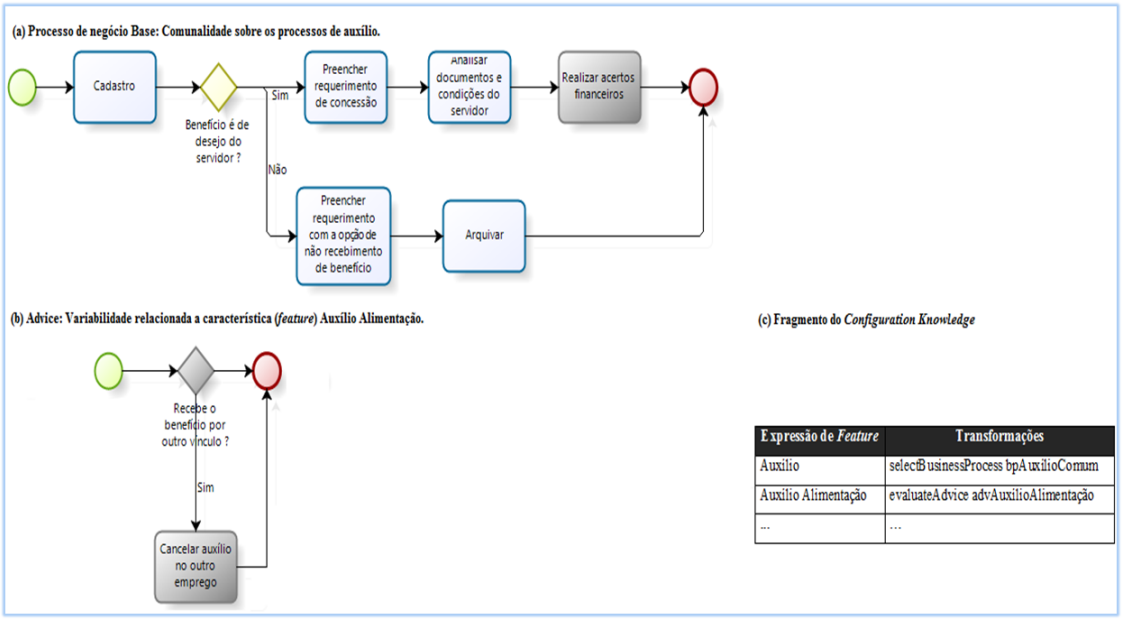


Figura 3.9: Composição de Processo de Negócio Comum com Variante

de Auxílio e pelo *advice* Auxílio Alimentação, como a Figura 3.7 mostra. Diferentemente, se o produto não está configurado com a *feature* de Auxílio Alimentação, a composição mencionada não vai acontecer, já que a transformação *evaluateAdvice* "advAuxílioAlimentação" não será avaliada. Em suma, o processo de derivação do produto avalia se uma expressão de *feature* é válida ou não para um determinado produto, gerando uma lista com as transformações relacionadas que devem ser aplicadas. Depois disso, ele chama cada transformação iterativamente, resolvendo as variabilidades do produto de entrada.

No Capítulo 4, Seção 4.2.5, pode ser visto um exemplo de tratamento de variabilidade em condições ("Sim" e "Não") de *gateway* quando anotado no processo-base.

Capítulo 4

Avaliação

Neste capítulo fizemos uma avaliação empírica no domínio de RH, avaliando a extensão de MSVCM Bonifácio e Borba (2009), proposta no Capítulo 3. Este capítulo está estruturado da seguinte forma: iniciamos com a descrição do contexto e objetivos da avaliação (Seção 4.1), descrevemos a Pesquisa-Ação realizada (Seção 4.2) e, por fim, realizamos as considerações finais (Seção 4.3) onde registramos as principais reflexões sobre a Pesquisa-Ação.

4.1 Contexto e Objetivos

Em organizações de médio e grande porte, tanto públicas quanto privadas, existem processos de negócio repetidos sem que haja um padrão genérico definido. Por exemplo, as atividades relacionadas ao processo de cadastro de órgãos e unidade da administração federal estão replicadas em cada órgão já que esse processo não é genérico. No âmbito da Tecnologia da Informação e Comunicação (TIC), isso gera problemas de redundância de funcionalidades e de dados em diferentes sistemas, sub-aproveitamento de recursos humanos e difícil integração entre sistemas por utilizarem código diferente para representar os mesmos órgãos ou unidades da administração. Este cenário traz uma necessidade clara de estudos de modelagem, sistematização e parametrização de processos de negócio com a finalidade de evitar a replicação de processos, otimizar alocação de recursos, agilizar o atendimento aos clientes, aumentar a qualidade desse atendimento, melhorar o planejamento, controle e implementação de melhorias nos sistemas Hammer e Champy (1994a). No entanto, não apenas existe uma carência na modelagem destes processos, mas também se observa uma inadequação das técnicas utilizadas Sawy (2001); Harmon et al. (2000); Davenport (1993). Por isso, se fazem necessárias uma técnica e uma ferramenta capazes de permitir gerenciar as variabilidades ilustradas em processos de negócio.

A fim de avaliar o método/extensão descrita no Capítulo 3, conduzimos um estudo empírico (descrito na Seção 4.2), com o objetivo ilustrado na Figura 4.1, descrito segundo *Goal, Question, Metric* GQM Basili et al. (1994).

Objetivo	*gerenciar*
questão (issue)	*a variabilidade*
objeto	*de BPM*
perspectiva	*Engenharia do Domínio e de Aplicação*
contexto	*RH*

Figura 4.1: Objetivo do Estudo Empírico - Método GQM

4.2 Pesquisa-Ação

Utilizamos Pesquisa-Ação (*Action Research*) e método empírico de investigação e validação, descrita em Easterbrook (2007). O motivo de utilizá-la foi porque estamos interessados em resolver um problema real (como tratar variabilidade em processos de negócio no domínio de RH, enquanto estudamos a experiência em resolvê-lo). Como preconizado, existem algumas premissas para se adotar Pesquisa-Ação, tais como: (i) ter um dono do problema (*problem owner*) para colaborar, identificar o problema e estar engajado no esforço para resolvê-lo; e (ii) o problema deve ser autêntico (um problema real deve existir) e, como consequência, não haver conhecimento anunciado de resultados autênticos para tal problema.

Esta Pesquisa-Ação foi guiada pelo método apresentado no Capítulo 3, Seção 3.1. A primeira etapa desse método é elaborar o Modelo de *Features* (FM). Para isso, utilizando a estratégia de adoção extrativa Krueger (2001), foram catalogados os processos de negócio do domínio de RH. A partir desses processos, para que se chegasse à identificação das variabilidades (*features* opcionais do FM), foi necessário fazer uma prévia análise das comunalidades (*features* obrigatórias do FM), que envolveu identificar manualmente atividades e outros objetos *(gateways, messsage flows, entre outros)* comuns. A partir disso, foi realizada a análise das variabilidades. Essas atividades foram realizadas baseadas em processos de negócio, no domínio de RH, modelados por uma equipe de projeto de Pesquisa & Desenvovimento (P&D).

O objetivo desse projeto foi de modelar e melhorar tais processos de negócio. Nessa equipe, um grupo de analistas foi liderado na realização das seguintes tarefas: (1) modelar tais processos em BPMN, resultando em 52 processos (a escolha dessa notação foi uma restrição imposta pelo patrocinador desse projeto); (2) foi analisado manualmente o resultado dos processos de negócio por comunalidade e variabilidade; (3) foi construído o modelo de domínio (modelo de características - *feature model*).

Para viabilizar a etapa (1), o processo de extração foi realizado tendo como entrada modelos de processo de negócios existentes de três diferentes organizações no domínio de RH. Eles foram especificados em diferentes notações (textual e diagramas). Como integrante desse projeto (P&D), uma colega de mestrado, inicialmente, desempenhou o papel de *dona do problema* (item (i) supracitado, primeira premissa para realizar Pesquisa-Ação) por ter obtido conhecimento do problema ao participar ativamente da fase (1) do projeto. Esse papel foi assumido integralmente por este pesquisador durante as demais fases deste trabalho dado que os processos de negócio modelados no projeto foram cedidos para este trabalho. Além disso, este pesquisador participou das atividades (2) e (3) descritas. Portanto, dada a realização dessas atividades por este pesquisador, foi constatado que o item (ii) supracitado (segunda premissa para realizar Pesquisa-Ação) se aplica já que o problema de redundância de processos de negócio continua latente, conforme detalhado no Capítulo 5, na Seção 5.2. Nesse contexto, o FM está ilustrado na Figura 4.2. Como se

pode perceber nessa figura, 12 *features* foram modeladas, cada uma representando duas opções (relação *or-inclusive* do FM). O motivo dessa estrutura, foi porque, dos 52 processos analisados manualmente, constatamos comunalidades e variabilidades em 12 pares de processos, ou seja, 24 processos (24 de 52 = 46%). Os processos restantes representavam baixa comunalidade e/ou a granularidade das variabilidades eram tão finas Kästner et al. (2008) que não justificavam usar nossa abordagem. Identificamos que, a partir de um conjunto de três processos, as comunalidades eram baixas e/ou as variabilidades eram finas demais. Portanto, as comunalidades e variabilidades que motivaram a aplicação de nossa abordagem foram percebidas entre pares de processos. Como explicado no Capítulo 3, Seção 3.1.1, uma instância de um FM é um conjunto de *features* selecionadas que, por sua vez, tal conjunto corresponde a um conjunto de processos de negócios destinados para uma determinada organização. Logo, neste capítulo, nos concentraremos apenas no FM dos 12 casos avaliados, que estão resumidos na Figura 4.2.

Esta seção está estrtuturada da seguinte forma. Apresentamos os resultados da avaliação dos casos (pares de processos de negócio) ortogonais nas Subseções 4.2.1- 4.2.5. Cada caso desses compartilham a seguinte estrutura: Processos Originais, *BusinessProcessModel* - Processos Comuns e Variantes, *Configuration Knowledge*, Configuração do Produto e Análise do Resultado. Os casos restantes (outros sete casos) se encaixam nesses casos ortogonais e, por isso, são apresentados no Apêndice B com a mesma estrutura, exceto por não ter a seção "Análise do Resultado", já que se encaixa em algum caso ortogonal já explicado.

Antes de detalharmos cada caso, algumas observações comuns a todos os casos se fazem necessárias, são elas:

- Na Seção "Processos Originais" de cada caso, são apresentados os processos originais que serviram como insumo para a análise de comunalidades e variabilidades.
- Na Seção "*BusinessProcessModel* - Processos Comuns e Variantes" de cada caso apresentamos o processo comum e os variantes. Esses processos em formato de tipo algébrico estão no Apêndice C.
- Na Seção *"Configuration Knowledge"* de cada caso, é apresentado o CK específico. Esse artefato faz o mapeamento das expressões de *features* de cada caso para a(s) transformação(ões). O conjunto de todos esses CKs formam o CK para o FM apresentado na Figura 4.2. Em alguns casos, devido à Pesquisa-Ação, o CK evoluiu como parte de aperfeiçoamento da solução. Por isso, nesses casos, será apresentado o CK original e o corrigido.
- Na Seção "Configuração do Produto" de cada caso, são apresentadas as configurações possíveis de cada caso. Em cada seção dessa, mostramos uma figura de uma tabela, onde, na primeira coluna, ilustramos as possíveis configurações (PCs) e, na segunda coluna, ilustramos o resultado da execução da(s) transformações(ão) do CK mapeada(s) pela(s) *feature(s)* selecionada(s) em cada PC. Esse resultado ilustra o estado do produto final, que são os tipos algébricos instanciados mostrados no Apêndice D. Nessa figura, são utilizados símbolos cujos significados estão descritos na Figura 4.3. A medida que cada símbolo for apresentado em cada caso, uma breve explicação será feita para facilitar a leitura.

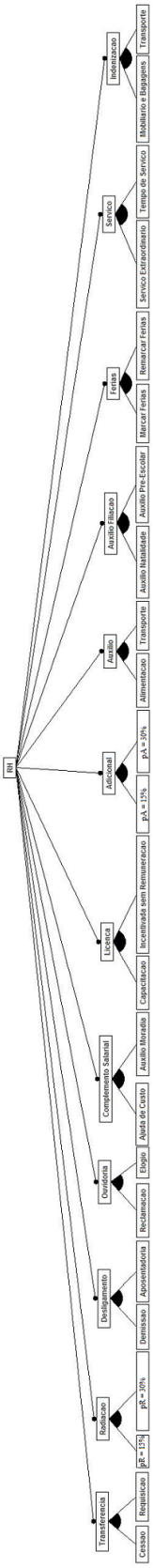


Figura 4.2: Modelo de Feature dos 12 casos avaliados

- Os tipos algébricos que representam os processos de negócios (comuns e variantes), referenciados em cada caso (estão no Apêndice C), não representam as raias desses processos visto que está fora do escopo deste trabalho tratar variabilidades em papéis de negócio (representados pelas raias). As notas/textos explicativos contidos nos diagramas estão sendo desconsiderados pelo mesmo motivo. Devido à decisão de projeto (descrito na Seção 3.2.3) de contemplar apenas os elementos *Start, Activity, Gateway, End, Proceed* no tipo algébrico *FlowObject* e, para nos restringirmos a esse conjunto reduzido de elementos, convencionamos, sem prejudicar o uso de nossa abordagem, representar as variações do objeto mensagem da BPMN (*MessageFlow* Obj (2009)) utilizadas nos processos de negócio do domínio RH como *Activity*. O código-fonte de cada tipo algébrico que representa os processos-base e processos-variantes (*BusinessProcessModel*) foi criado manualmente pelo pesquisador. Eles foram compilados, executados e testados e estão no Apêndice C. Foram criadas funcionalidades de *parser* para gerar esses tipos algébricos tendo como entrada os XMLs dos processos-base e processos-variantes. Esses *parsers* foram criados por um aluno de mestrado, mas, devido a restrições de cronograma, não foi possível esperar sua conclusão para iniciarmos esta avaliação. Dessa forma, instanciamos os tipos algébricos manualmente para modelar o *BusinessProcessModel*. Representar os elementos da BPMN faltantes, tratar variabilidades de raias bem como utilizar os *parsers* fazem parte de trabalhos futuros.
- Em todas as imagens de processos de negócio com linhas vermelhas envolvendo algum elemento do processo ou um trecho do processo, estaremos evidenciando a variabilidade desse processo em relação ao seu par.
- Todos acentos e caracteres especiais foram retirados dos artefatos - *BusinessProcessModel* (tipos algébricos), FM, CK, PC - que são as entradas para a infraestrutura estendida *Hephaestus* por causa de restrição da linguagem de programação utilizada por essa ferramenta, linguagem Haskell.

4.2.1 Caso I - Transferência - Cessão e Requisição

Processos Originais

Nesta seção temos os processos (Figuras 4.4, 4.5) que descrevem os passos para ceder um servidor de um órgão para outro bem como fazer a sua requisição.

BusinessProcessModel - Processos Comuns e Variantes

Como podemos perceber, os processos originais (Figuras 4.4, 4.5) deste caso são idênticos, exceto com relação ao nome dos mesmos. Portanto, para este caso, não temos processo-variante e, qualquer um desses processos pode ser eleito para desempenhar o papel do processo-base. O tipo algébrico que representa esse processo-base está em Apêndice, na Seção C.1.

Configuration Knowledge

Na Figura 4.6, apresentamos o CK deste caso.

Símbolo	Significado
	Representa o processo de negócio original sem a aplicação de qualquer transformação (processo-base)
	Representa o processo comum dos processos de negócio origionais sem a aplicação de qualquer transformação (processo-base)
	Representa as variações (processo-advice) do processo-base
X% , Y%	Representa as variabilidades de valor do processo-base
,	Representa o término da execução de alguma transformação
	Representa a execução da transformação *"select ..."*
	Representa a execução da transformação *"bindParameter ..." ou "selectBind ..."*
	Representa a execução da transformação *"evaluateAdvice ..."* ou *"selectEval ..."*
	Representa o *Weaving Process*

Figura 4.3: Significado dos Símbolos Utilizados em Cada Caso

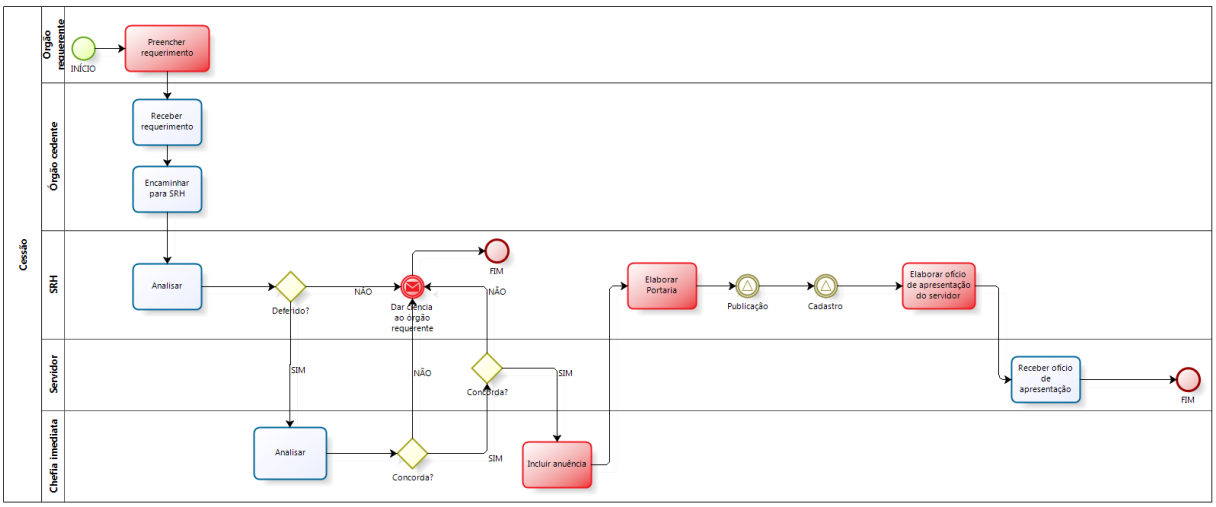


Figura 4.4: Processo Original - Cessão

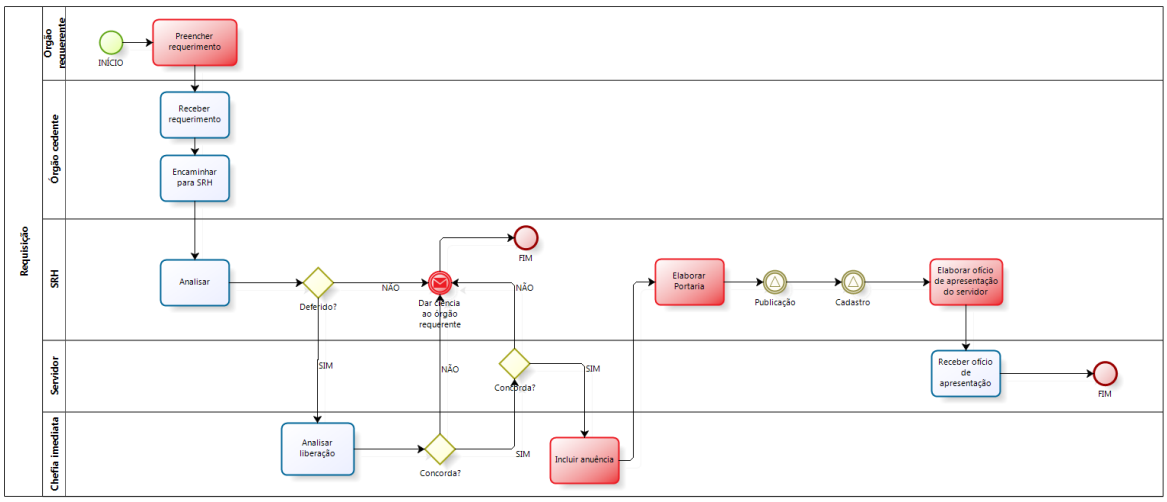


Figura 4.5: Processo Original - Requisição

CK	
Feature Expressions	**Transformations**
Cessao	selectBusinessProcess "bpParteComumCessaoRequisicao" spl product
Requsicao	selectBusinessProcess "bpParteComumCessaoRequisicao" spl product

Figura 4.6: CK - Transferência - Cessão e Requisição

Configuração do Produto

Na Figura 4.7, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.1. Na segunda coluna dessa figura, temos um símbolo (uma seta retangular apontando para à direita) que representa a execução da transformação *selectBusinessProcess* (explicada no Capítulo 3, Seção 3.2.2 e presente integralmente, juntamente com suas funções auxiliares, no Apêndice A). Esse símbolo significa que o processo-base *"bpParteComumCessaoRequisicao" (símbolo quadrado vermelho)* está sendo selecionado da Linha de Processos de Negócio (LPN) (variável *spl* da transformação) e sendo adicionado no produto (variável *product*) de cada configuração (primeira coluna da figura). O produto final de cada configuração é apresentado entre chaves "{...}".

Análise do Resultado

Este caso retrata apenas comunalidade entre os processos originais, visto que, integralmente (exceto pelo nome dos processos) se demonstraram iguais. Este é o caso mais simples na utilização de nossa abordagem, já que utiliza apenas a transformação se seleção de processos da *BusinessProcessModel* e os adiciona no produto final como instâncias do tipos algébricos de processo de negócio (ativos do *BusinessProcessModel*).

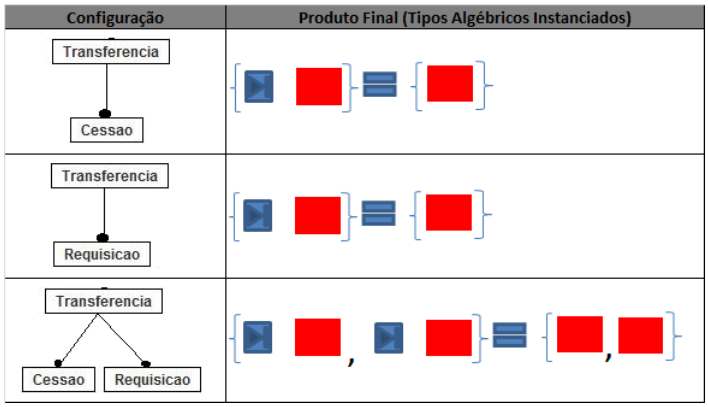


Figura 4.7: Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

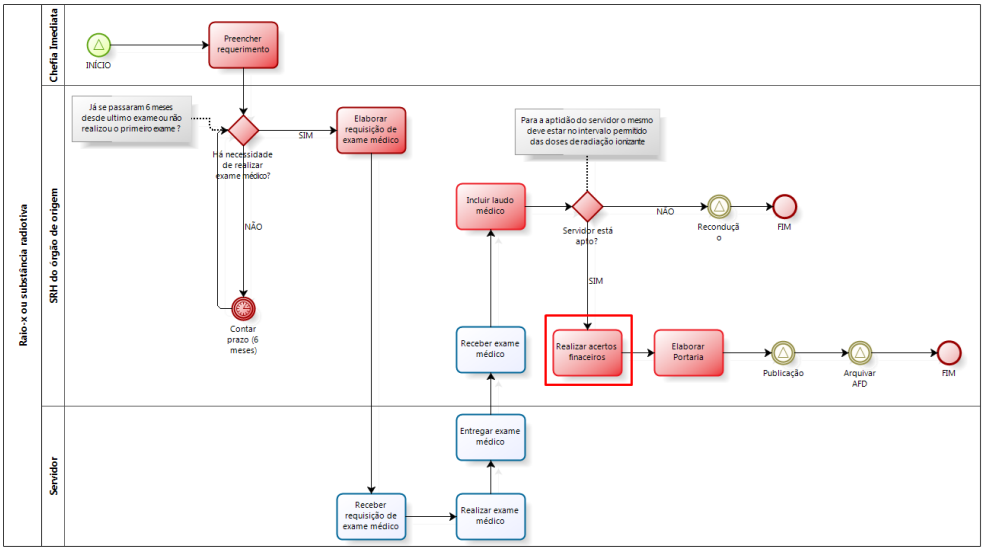


Figura 4.8: Processo Original - Raio-X ou Substância Radioativa - Com Ponto de Variabilidade Destacado

4.2.2 Caso II - Radiação - Raio-X ou Substância Radioativa e Irradiação Ionizante

Processos Originais

Nesta seção temos os dois processos originais do domínio de RH (Figuras 4.8, 4.9) que descrevem os passos para submeter um servidor exposto à radiação (RaioX ou Irradiação Ionizante) a exames médicos.

BusinessProcessModel - Processos Comuns e Variantes

Neste temos variabilidade apenas de valor, em uma atividade comum aos processos originais (Figuras 4.8 e 4.9). Com base nessas ilustrações, percebemos, na atividade destacada, "Realizar ajustes financeiros", o ponto de variabilidade entre esses dois processos. Os demais elementos, exceto o nome dos processos, são idênticos. Portanto, o processo comum desses processos representa, em termos de estrutura, qualquer um dos processos originais,

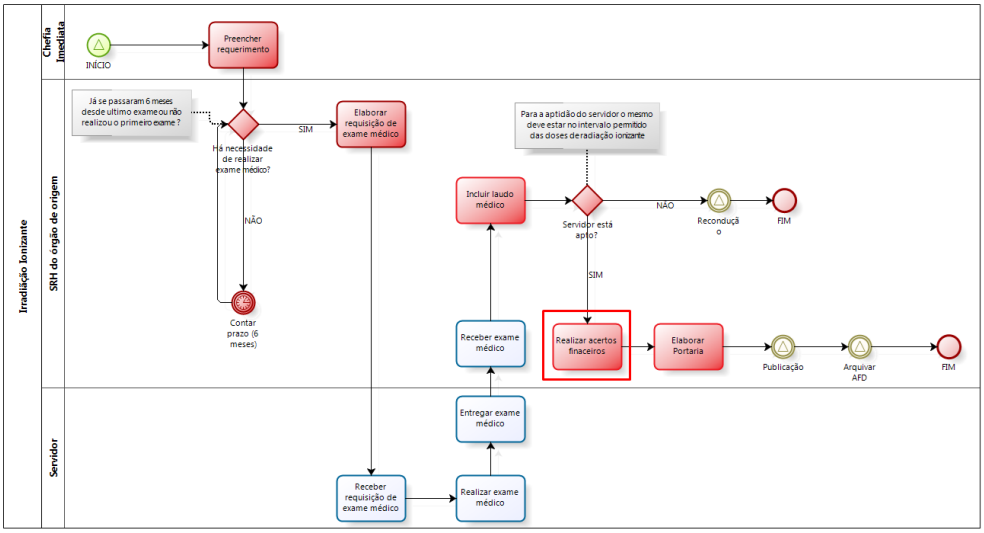


Figura 4.9: Processo Original - Irradiação Ionizante - Com Ponto de Variabilidade Destacado

CK	
Feature Expressions	**Transformations**
pRX = 15%	selectBusinessProcess "bpParteComumRaioXRadiacaoIonizante" spl product bindParameter "pR" "15%" spl product
pRX = 30%	selectBusinessProcess "bpParteComumRaioXRadiacaoIonizante" spl product bindParameter "pR" "30%" spl product

Figura 4.10: Caso II - Radiação - CK Origional - Problema ao Executar bindParameter pela segunda vez

exceto por ter dois valores, um para cada processo. A variabilidade é expressa no processo comum através de valores distintos atribuídos ao parâmetro. O tipo algébrico do processo-base (processo-comum eleito) deste caso está em Apêndice, na Seção C.2.

Configuration Knowledge

Neste caso apresentamos dois CKs, o CK original 4.10, elaborado antes da avaliação, e o CK corrigido, Figura 4.11.

Configuração do Produto

Identificamos um problema na execução da transformação *bindParameter*, mostrada no CK original, (Figura 4.10) quando as duas *features*, "pR = 15%" e "pR = 30%", eram selecionadas no PC. Na Figura 4.12, ilustramos o problema.

CK	
Feature Expressions	**Transformations**
pRX = 15%	selectBind "bpParteComumRaioXRadiacaoIonizante" spl "pRX" (Value "15%")
pRX = 30%	selectBind "bpParteComumRaioXRadiacaoIonizante" spl "pIonizante" (Value "30%")

Figura 4.11: Caso II - Radiação - CK Corrigido - Problema Corrigido com a Criação da transformação selectBind

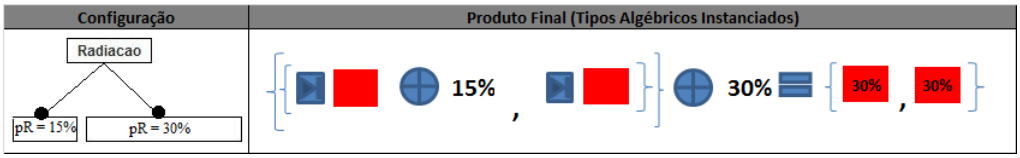


Figura 4.12: Caso II - Radiação - Problema quando as Duas *Features* eram Selecionadas no PC

Antes de detalhar o problema, faz-se necessário explicar dois novos símbolos, o símbolo "+", em forma circular, e os valores ("15%" e "30%"). Os elementos entre as chaves indicam que (leitura da esquerda para direita): (i) o processo-base "bpParteComumRaioXRadiacaoIonizante" foi selecionado da *spl* e adicionado no produto (*product*); (ii) por meio do símbolo "+" em forma circular, é atribuído o valor "15%" ao parâmetro *pR*; (iii) selecionado e adicionado no produto um novo processo-base, ""bpParteComumRaioXRadiacaoIonizante"; e, em seguida, agora surge o problema, (iv) por meio do símbolo "+" em forma circular, ao atribuir o valor "30%" ao parâmetro *pR* do último processo-base inserido no produto, é feito o esperado para o segundo, mas sobrescrito o valor "15%" por esse último valor. Isso ocorria porque, quando se ia executar a transformação *bindParameter*(explicada no Capítulo 3, Seção 3.2.2 e contida integralmente no Apêndice A) da segunda feature do CK original(Figura 4.10), "pR = 30%", essa transformação realizava alteração do valor do parâmetro *pR* do processo transformado referente à primeira feature, "pR = 15%", conforme é ilustrado na Figura 4.12 e pode ser constatado no produto final no Apêndice, Subseção D.2.3. O motivo desse erro é porque a função *bindParameter* realiza transformação de escopo global, ou seja, em todos os processos contidos no produto. Portanto, quando as duas *features* estão selecionadas no PC, a aplicação dessa transformação, a partir da segunda feature, sobrepõe o valor atribuído (no caso, "15%") ao parâmetro *pR*, por isso são mostrados na Figura 4.12 dois produtos finais com o valor "30%".

Para resolver esse problema, foi criada a transformação *selectBind*, conforme mostrada no segundo CK (Figura 4.11). Essa transformação resolve o problema porque realiza a resolução da variabilidade de valor de forma local para, após isso, inserir o processo transformado dentro do produto. Na Figura 4.13, mostramos as configurações possíveis e os resultados deste caso corrigidos. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.2.

Apresentamos abaixo a transformação *"selectBind"*.

```
1 selectBind :: Id → BusinessProcessModel → Name → Value → BusinessProcessModel
2 selectBind bpId spl np vp =
3   let process1 = selectBusinessProcess bpId spl (BPM {processes = []});
4       process2 = bindParameter np vp process1
5   in case (processes process2) of
6     [process] → process2
7     otherwise → BPM {processes = []}
```

Essa transformação, basicamente, se resume a fazer uma chamada a transformação *selectBusinessProcess* e à *bindParameter* (linhas 2-3). Ela recebe quatro parâmetros de entrada - linha 1, *bpId*(identificação do processo-base), *spl*(LPN), *np*(nome do parâmetro contido no processo-base) e *vp*(valor a ser atribuído ao parâmetro) - e retorna um

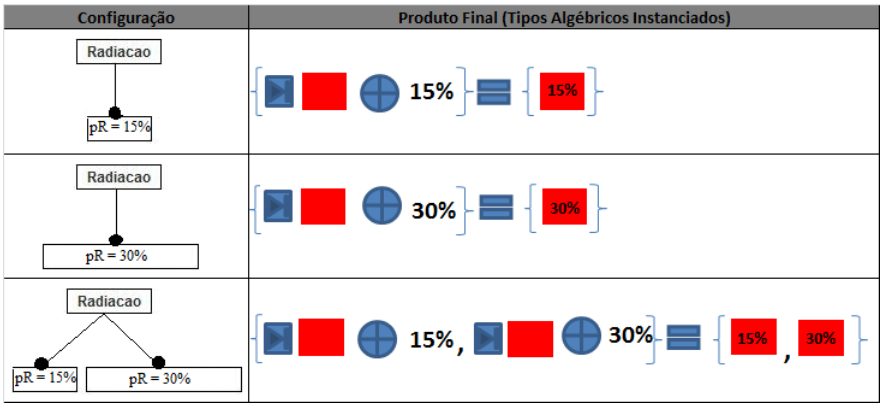


Figura 4.13: Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

produto (do tipo *BusinessProcessModel*) contendo o processo transformado (linha 5) ou um produto vazio (linha 6) - caso o *process2* (linha 4) seja vazio. Essa variável poderá estar vazia somente por um motivo: quando *process1* estiver vazia, seja porque o *bpId* informado não existe em *spl* ou porque a *spl* está vazia. Observe que na linha 2 é feita a seleção do processo-base (*bpId*), a partir da *spl*, e um produto vazio (terceiro parâmetro é informado). Ou seja, nesse ponto que foi viabilizada a restrição de escopo, motivação principal da criação desta transformação, visto que *bindParameter* é aplicada (linha 3) a um produto (*process1*) que contém apenas o processo-base de interesse (o retornado por *selectBusinessProcess* na linha 2).

Análise do Resultado

Neste caso, foi identificado um problema na aplicação da transformação *"bindParameter"*, que ocorre quando as *features* selecionadas estão vinculadas a transformações que afetam processos-base de mesmo nome contidos no produto, que está sendo refinado. O motivo disso ocorrer é porque essa transformação resolve variabilidade de valor em todos os processos do produto desde que ocorra o casamento do nome do parâmetro informado como entrada para a função com os nomes dos parâmetros de qualquer processo contido no produto. Para solucioná-lo, tivemos que restringir o escopo de aplicação dessa função apenas para o processo-base desejado e, após feito isso, o processo transformado é inserido no produto. Essa solução foi viabilizada com a criação da transformação *"selectBind"*. Neste caso, não esperávamos combinar objetos de fluxo e as transições com processo de negócios existentes, em vez disso, nós apenas tivemos que vincular os valores de um parâmetro de objeto de fluxo com a(s) opção(ões) selecionada(s) da *feature Radiacao*. Isso foi viabilizado por causa da extensão a BPMN para permitir tratamento de variabilidade de granularidade fina, conforme descrito no Capítulo 3, Seção 3.2.1.

4.2.3 Caso III - Desligamento - Demissão e Aposentadoria

Processos Originais

Nesta seção temos os dois processos originais do domínio de RH (4.14 e Figuras 4.15) que descreve os passos para um servidor ou empregado se aposentar ou ser demitido.

Figura 4.14: Processo Original - Demissão

Figura 4.15: Processo Original - Aposentadoria - Com Ponto de Variabilidade Destacado

BusinessProcessModel - Processos Comuns e Variantes

Para esse par de processos, o processo Demissão, ilustrado na Figura 4.14, desempenha o papel do processo comum, visto que o processo Aposentadoria (Figura 4.15) possui os mesmos elementos (e o mesmo fluxo) do processo Demissão mais a variablidade destacada no processo Aposentadoria, atividade "Analisar requisitos legais do processo". Portanto, para representar essa variabilidade, a modelamos como processo do tipo *advice*, conforme apresentado na Figura 4.16. Os tipos algébricos do processo-base (Demissão) e do processo-advice (Adv-Aposentadoria) estão em Apêndice, na Seção C.3.

Configuration Knowledge

Na Figura 4.17, mostramos o CK original e, na Figura 4.18, o CK corrigido.

Configuração do Produto

Na Figura 4.19, resumimos as configurações possíveis e os resultados deste caso baseado no CK original. Nessa figura inserimos um novo símbolo, o "+", que representa a execução da transformação *evaluateAdvice*(explicada no Capítulo 3, Seção 3.2.2). Essa função resolve a variabilidade destacada no processo Aposentadoria (Figura 4.15) e representada no processo-advice Aposentadoria(Figura 4.16). Essa transformação faz a composição do processo-base (processo Demissão) com o processo-advice gerando ao final um produto que tenha o processo Aposentadoria com sua variabilidade resolvida.

Neste caso também foi encontrado um problema quando as duas *features* eram escolhidas no PC, conforme o "X" ilustrado na Figura 4.19 para a terceira configuração. Mas, para descrever esse problema detalhadamente, apresentaremos nesta seção o produto final, de forma ilustrativa, quando as duas *features* Demissao e Aposentadoria forem

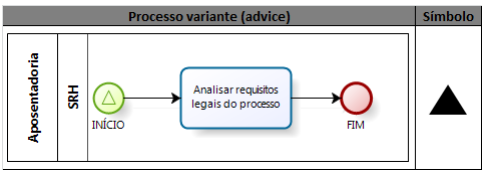


Figura 4.16: Processo Variante (Advice) - Aposentadoria

CK	
Feature Expressions	**Transformations**
Demissao	selectBusinessProcess "bpDemissao" spl product
Aposentadoria	selectBusinessProcess "bpDemissao" spl product evaluateAdvice "advAposentadoria" spl product

Figura 4.17: Caso III - Desligamento - CK Original - Problema da Ordem das Linhas

CK	
Feature Expressions	**Transformations**
Aposentadoria	selectBusinessProcess "bpDemissao" spl product evaluateAdvice "advAposentadoria" spl product
Demissao	selectBusinessProcess "bpDemissao" spl product

Figura 4.18: Caso III - Desligamento - CK Corrigido - Problema Corrigido Alterando a Ordem das Linhas

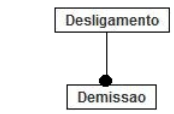
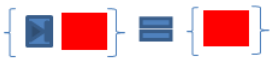
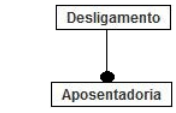
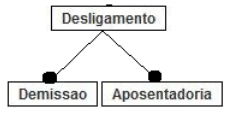
Configuração	Produto Final (Tipos Algébricos Instanciados)
	
	
	

Figura 4.19: Caso III - Desligamento - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado - Baseado no CK Original

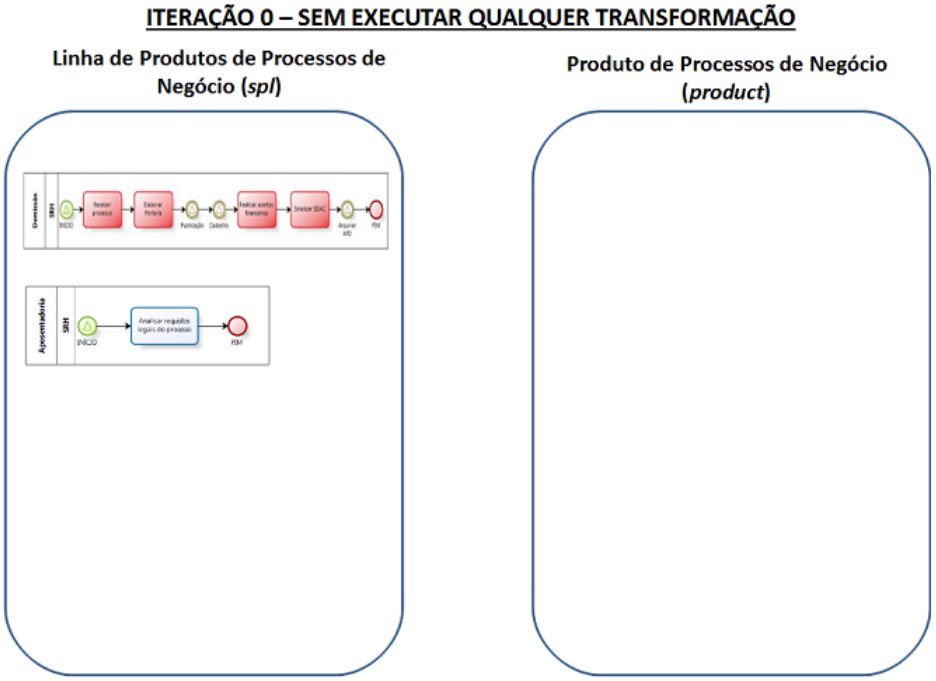


Figura 4.20: Caso III - Desligamento - Problema - Iteração 0

selecionadas no PC e utilizando o CK original 4.17 como base. Utilizaremos as ilustrações das Figuras 4.20, 4.21, 4.22 e 4.23, apresentadas de forma iterativa. Todas essas imagens tem em comum dois lados, o esquerdo e o direito. O lado esquerdo representa a LPN (variável *spl* utilizada nas transformações). O lado direito representa o produto(variável *product* presente nas transformações), em fase de refinamento ou em fase final. Conforme conceitos sobre engenharia de domínio (*domain engineering*) e engenharia de aplicação (*application engineering*) em Czarnecki e Eisenecker (2000), a LPN faz parte dos artefatos de nossa engenharia de domínio o produto da engenharia da aplicação.

A Figura 4.20 representa o estado do produto antes da aplicação de qualquer transformação, neste caso, temos um produto sem qualquer processo de negócio em sua lista de processos. Mas é importante esclarecer que poderiam existir processos dentro desse produto caso outras *features*/transformações tivessem sido escolhidas se considerarmos que estaríamos trabalhando com um PC mais amplo (com mais *features* selecionadas) que o ilustrado neste caso.

Seguindo então a execução das transformações do CK original, Figura 4.17, a partir da primeira linha - feature Demissao, na Figura 4.21, vemos a aplicação da única transformação da feature Demissao. Essa transformação, *selectBusinessProcess "bpDemissao" spl product*, seleciona o processo de negócio "bpDemissao" da LPN (lado esquerdo da figura) e o carrega na lista de processos do produto (lado direito). Como não há qualquer outra transformação vinculada à feature Demissao, isso indica que esse produto nessa fase contém o processo que reflete o processo Demissão em sua fase final.

Agora, a partir da segunda linha do CK original, Figura 4.17, na Figura 4.22, vemos a transformação *selectBusinessProcess "bpDemissao" spl Product* sendo aplicada novamente já que é a primeira transformação vinculada à feature Aposentadoria. Essa operação produz o mesmo resultado descrito na iteração anterior.

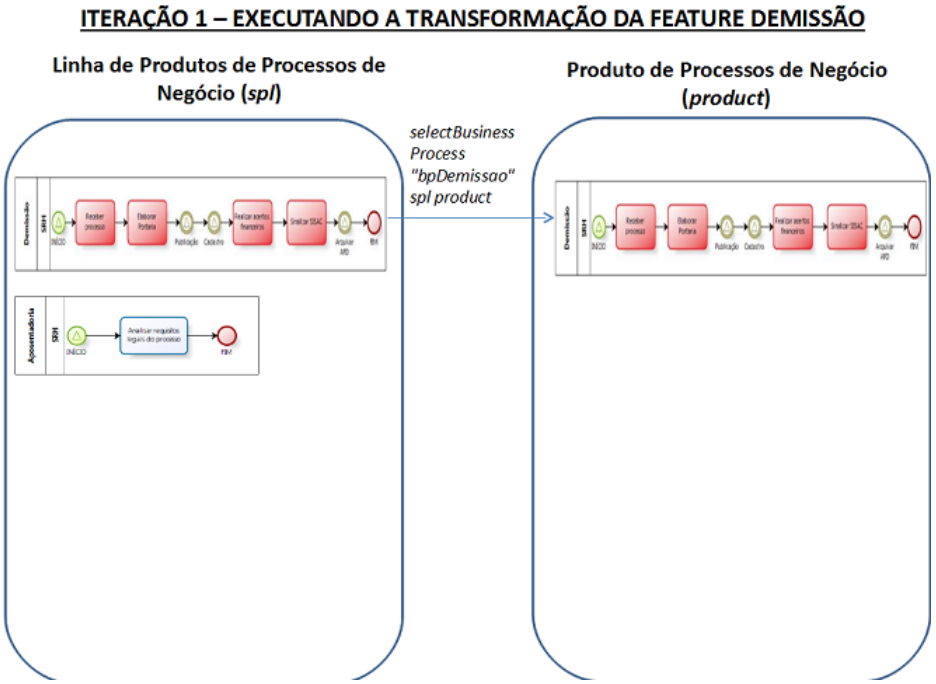


Figura 4.21: Caso III - Desligamento - Problema - Iteração 1

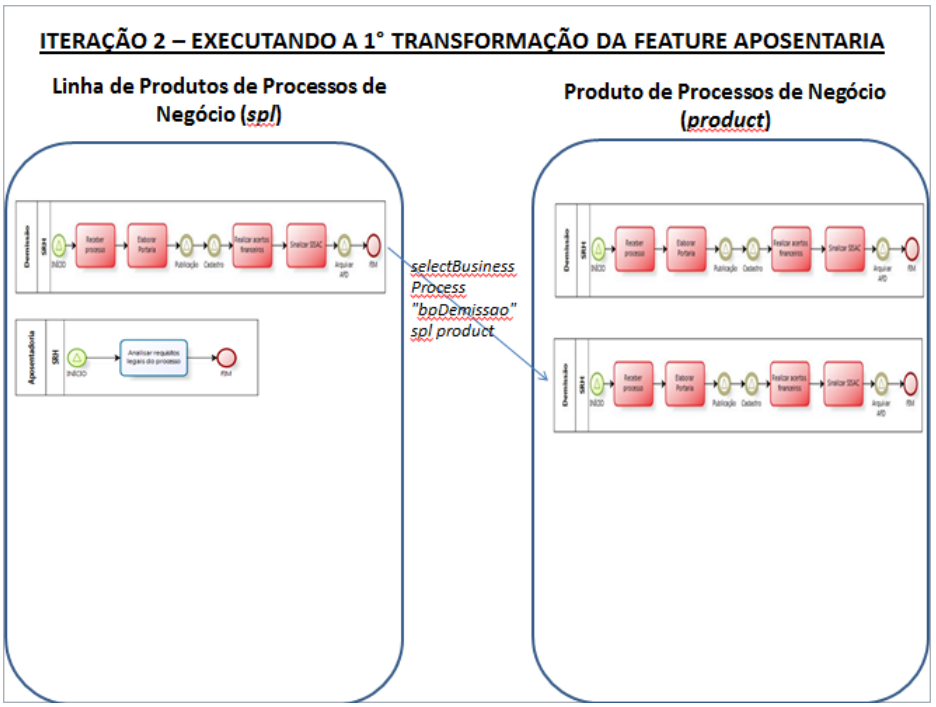


Figura 4.22: Caso III - Desligamento - Problema - Iteração 2

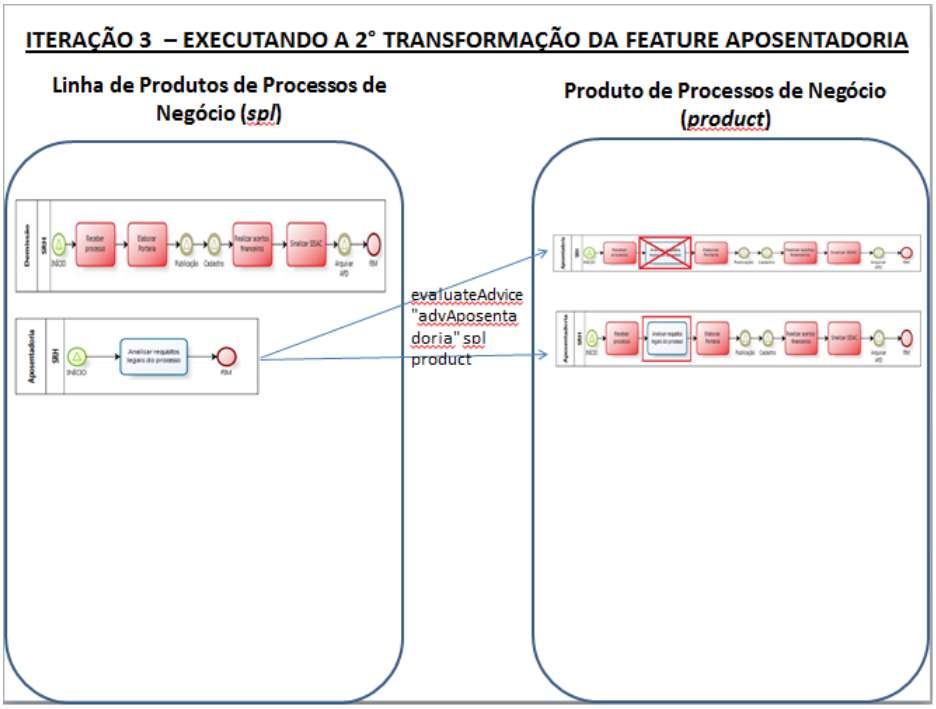


Figura 4.23: Caso III - Desligamento - Problema - Iteração 3

Em seguida, executando a segunda transformação vinculada à feature Aposentadoria, ainda na segunda linha do CK da Figura 4.17, podemos constatar o problema ilustrado na Figura 4.23. Vemos que, ao se aplicar a função *evaluateAdvice "advAposentadoria" spl product*, como seu comportamento é realizar a composição do processo-advice informado, "advAposentadoria" - Figura 4.15, com todos os processos do produto, isso insere um erro no estado do produto final, visto que o primeiro processo do produto, processo Demissão, não deveria ser transformado (atividade inserida inadequadamente marcada com "X" nessa última iteração).

Para solucionar esse problema, foi alterada a ordem das linhas do CK original, Figura 4.17, surgindo então o CK ajustado, Figura 4.18. Para ilustrarmos a correção do problema, apresentamos, nas ilustrações das Figuras 4.24, 4.25 e 4.26, a execução iterativa do CK ajustado, mostrando, por fim, na última iteração, iteração 3, Figura 4.26, que os processos do produto final refletem as *features* selecionadas no PC e são iguais aos processos originais, apresentados nas Figuras 4.14 e 4.15. Ressaltamos que a Figura 4.20 é comum a essas duas explicações iterativas - a que narra o problema e esta referente à solução.

Em Apêndice, na Seção D.3.3, listamos os tipos algébricos que refletem o produto final da Figura 4.23. Na Seção D.3.4, listamos os tipos algébricos que refletem o produto final da Figura 4.26. Por fim, na Figura 4.27, resumimos as configurações possíveis e os resultados deste caso baseado no CK corrigido. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.3.

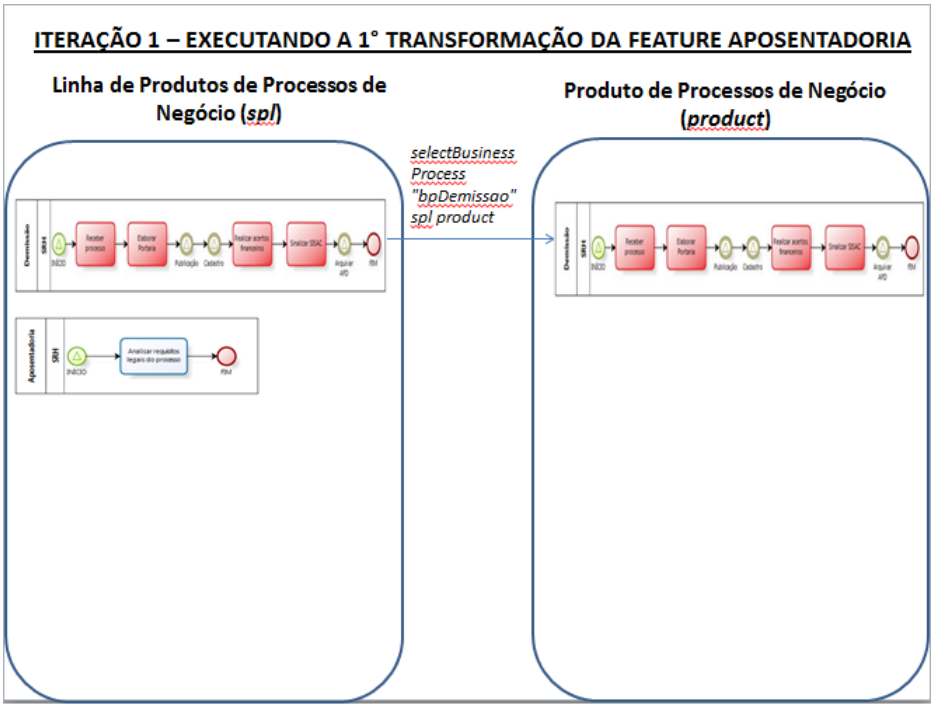


Figura 4.24: Caso III - Desligamento - Solução - Iteração 1

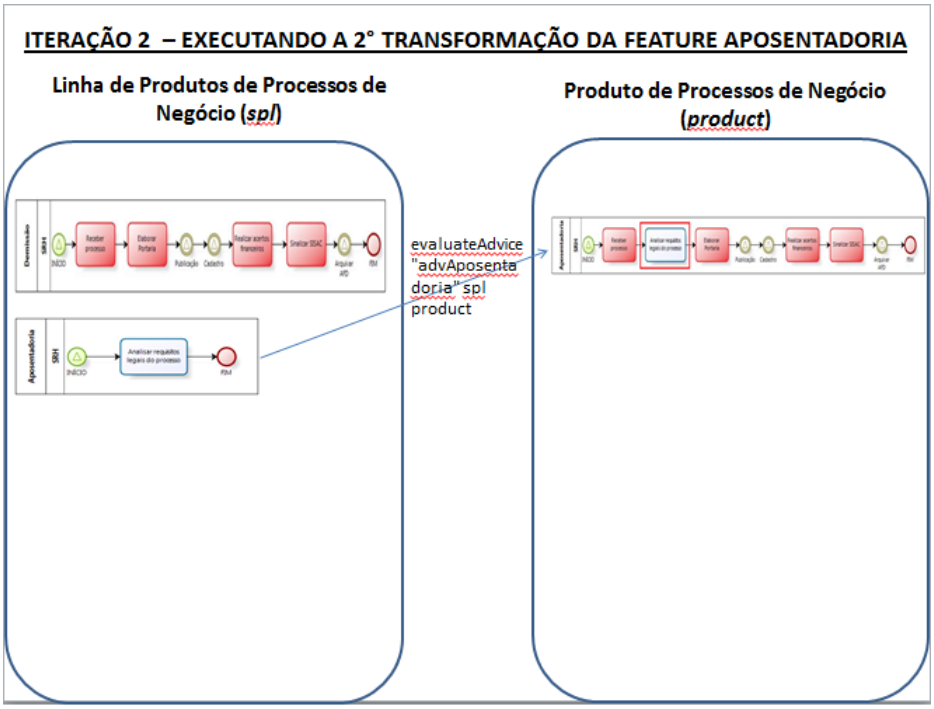


Figura 4.25: Caso III - Desligamento - Solucao - Iteração 2

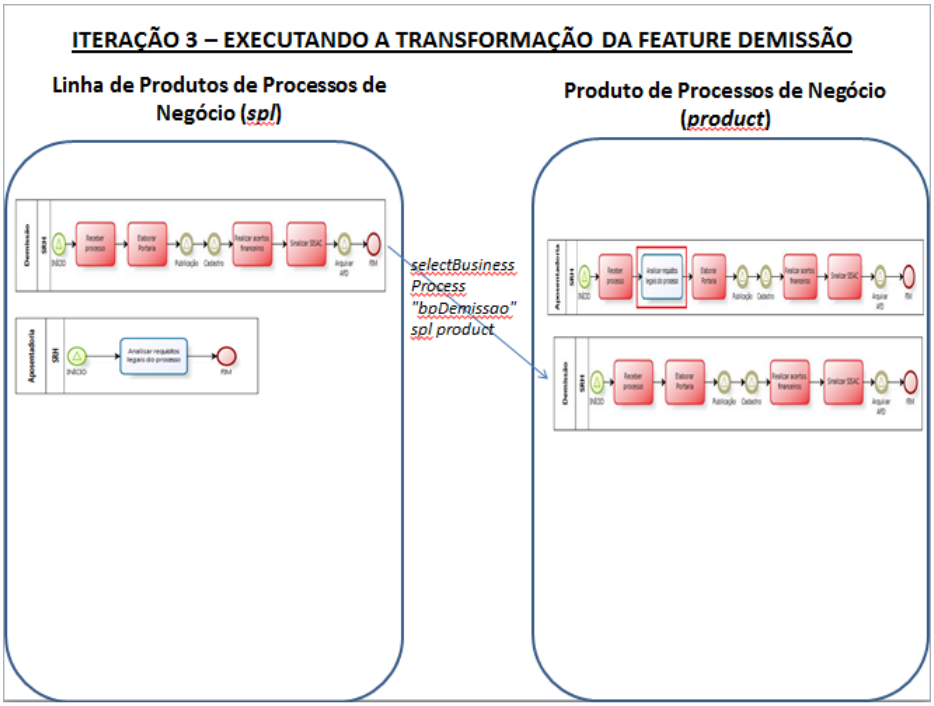


Figura 4.26: Caso III - Desligamento - Problema - Iteração 3

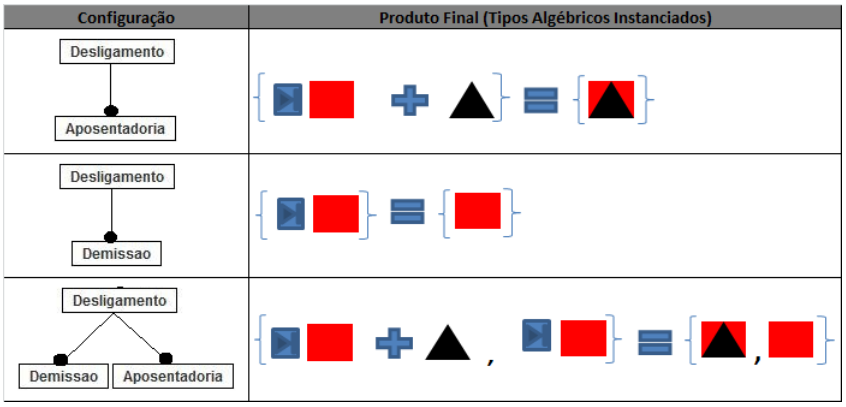


Figura 4.27: Caso III - Desligamento - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado - Baseado no CK Corrigido

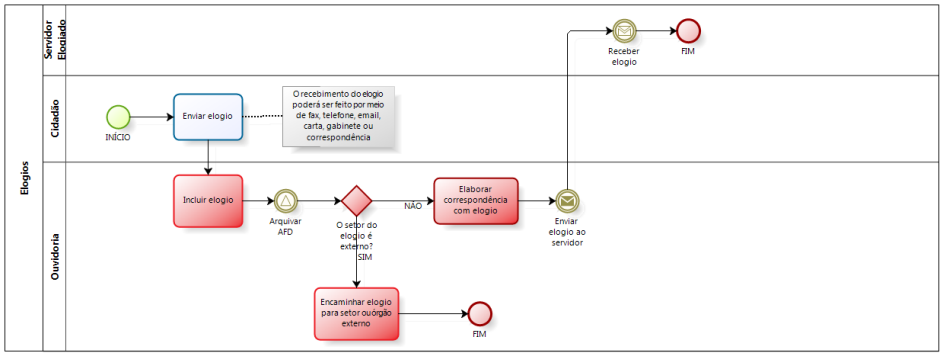


Figura 4.28: Processo Original - Elogio

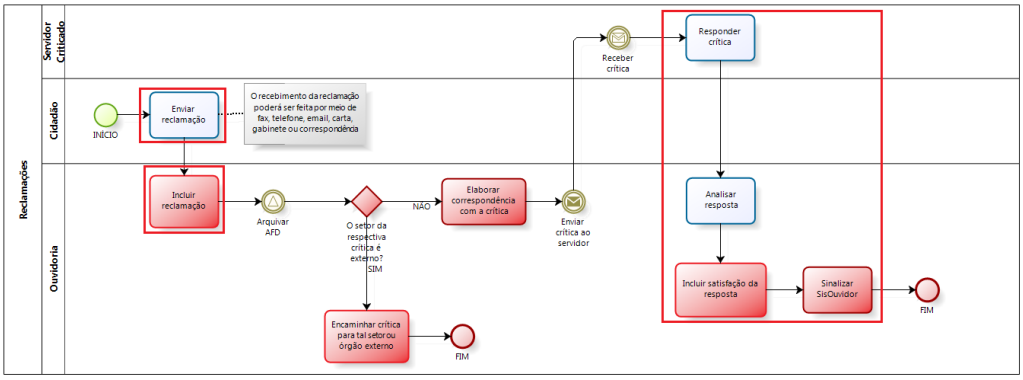


Figura 4.29: Processo Original - Reclamação

Análise do Resultado

O problema foi resolvido com o reordenamento das linhas do CK. Este tipo de problema ocorre quando um dos processos desempenha o papel de processo-base, neste caso, o processo Demissão, que deve ficar na última linha do CK.

4.2.4 Caso IV - Ouvidoria - Reclamação e Elogio

Processos Originais

Nesta seção temos os dois processos originais do domínio de RH (Figuras 4.28, 4.29) que são descritos como formalizar um elogio e uma reclamação bem como os desdobramentos disso.

BusinessProcessModel - Processos Comuns e Variantes

Como podemos perceber, os processos originais (Figuras 4.28, 4.29) tem uma comunalidade (processo-comum, processo Elogio) e as variabilidades destacadas no processo Reclamação. A Figura 4.30 ilustra como essas variações foram tratadas em três processos-variantes (advices). Os tipos algébricos deste caso estão em Apêndice, na Seção C.4.

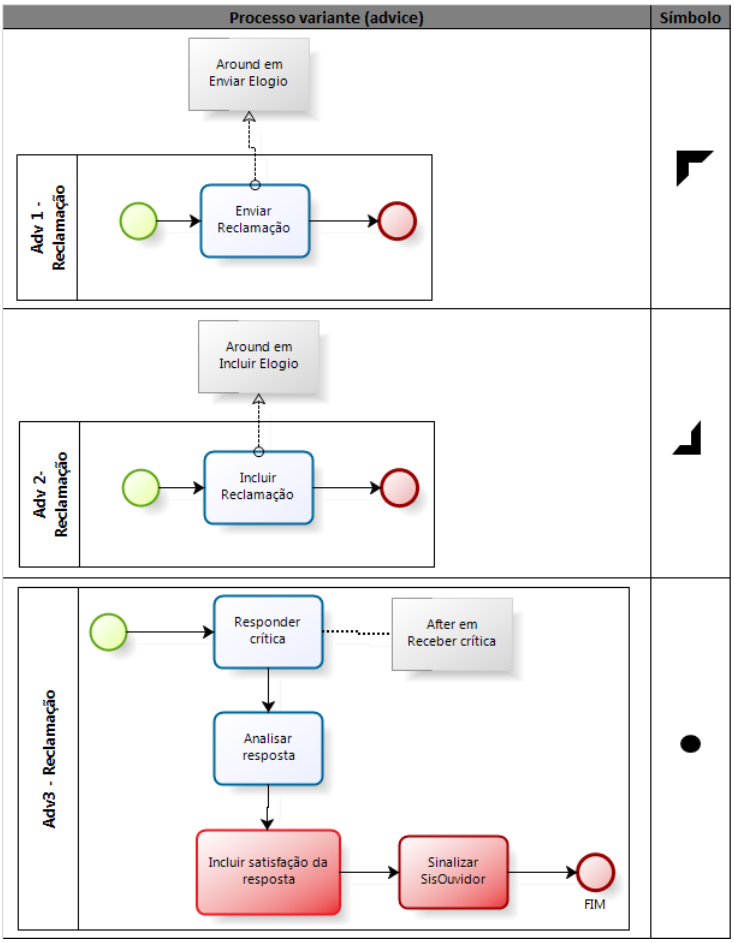


Figura 4.30: Processos Variantes (advices) entre Elogio e Reclamação

CK	
Feature Expressions	**Transformations**
Reclamacao	selectBusinessProcess "bpElogioComum" spl product, evaluateAdvice "advReclamacao1" spl product, evaluateAdvice "advReclamacao2" spl product, evaluateAdvice "advReclamacao3" spl product
Elogio	selectBusinessProcess "bpElogioComum" spl product

Figura 4.31: CK - Ouvidoria - Reclamação e Elogio

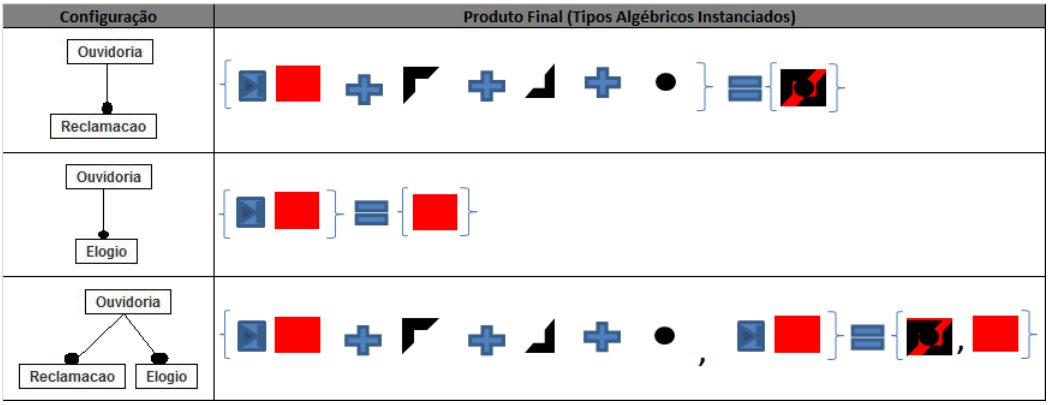


Figura 4.32: Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

Configuration Knowledge

Na Figura 4.31, apresentamos o CK deste caso.

Configuração do Produto

Na Figura 4.32, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.4.

Análise do Resultado

Neste caso, notamos que a ordem do CK é relevante, similar à situação reportada para a *feature* Desligamento (Demissao e Aposentadoria), Seção 4.2.3. Portanto, aplicamos os *advices* primeiro.

4.2.5 Caso V - Complemento Salarial - Ajuda de Custo e Auxílio Moradia

Processos Originais

Nesta seção temos os dois processos originais do domínio de RH (Figuras 4.33, 4.34) que descrevem os passos para conceder ajuda de custo ou auxílio moradia a um servidor/empregado.

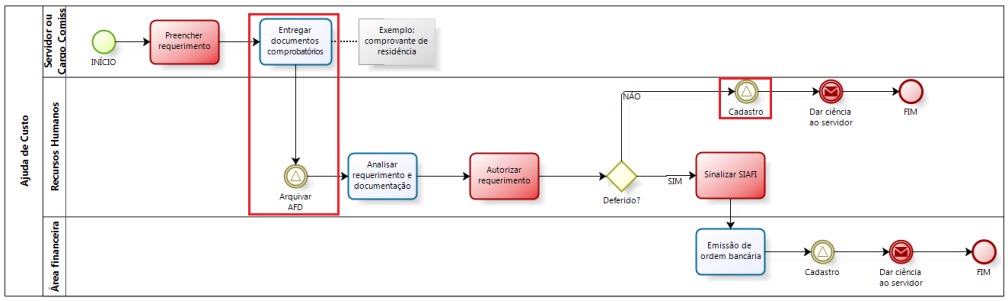


Figura 4.33: Processo Original - Ajuda de Custo

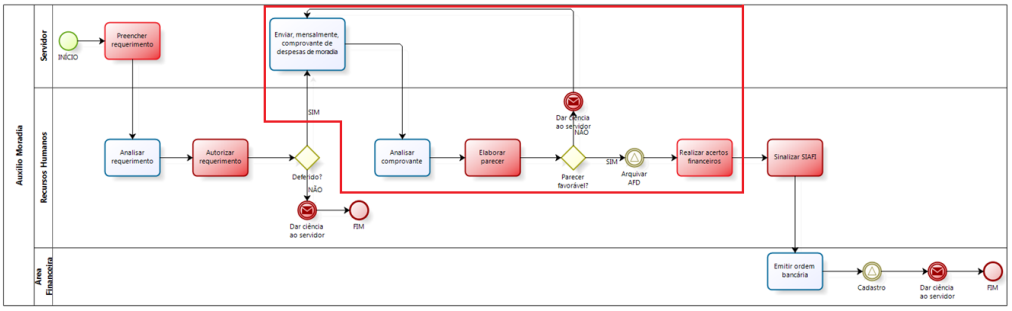


Figura 4.34: Processo Original - Auxílio Moradia

BusinessProcessModel - Processos Comuns e Variantes

Como podemos perceber destacado nos processos originais (Figuras 4.33, 4.34), há variações em ambos. Neste caso, temos que modelar em um processo a parte, o processo comum (Figura 4.35). As variações destacadas nos processos originais foram modeladas em processos-advice que estão ilustrados na Figura 4.36. Os tipos algébricos deste caso estão em Apêndice, na Seção C.5.

Configuration Knowledge

Na Figura 4.37, apresentamos o CK original e, na Figura 4.38, o CK corrigido.

Configuração do Produto

Na Figura 4.39, mostramos as configurações possíveis quando somente uma das *features* é selecionada. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, para feature Ajuda de Custo, na Seção D.5.1 e, para feature Auxilio Moradia, na Seção D.5.2.

Identificamos para este caso outro problema quando as duas *features* são selecionadas em uma configuração. Para explicar o problema, utilizaremos ilustrações iterativas (Figuras 4.40, 4.41, 4.42, 4.43, 4.44 e 4.45). Conforme se pode notar nas imagens, os processos do lado esquerdo, da Linha de Processos de Negócio *(spl)*, são os mesmos processos apresentados na Seção 4.2.5, mas mostrados de forma miniaturizada.

A Figura 4.41 mostra o que será feito ao executar a transformação *selectBusinessProcess "bpAjudaCustoComum" spl product* da feature "Ajuda de Custo" do CK original (Figura 4.37). É feita uma seleção do processo "bpAjudaCustoComum" da LPN (*spl*), lado

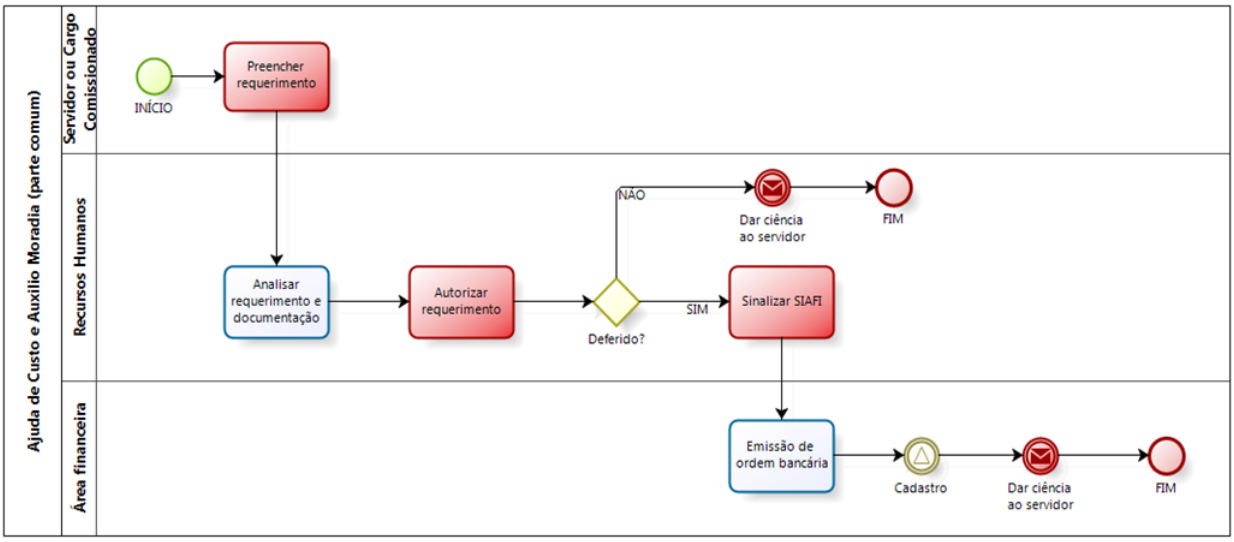


Figura 4.35: Processo Comum entre Ajuda de Custo e Auxílio Moradia

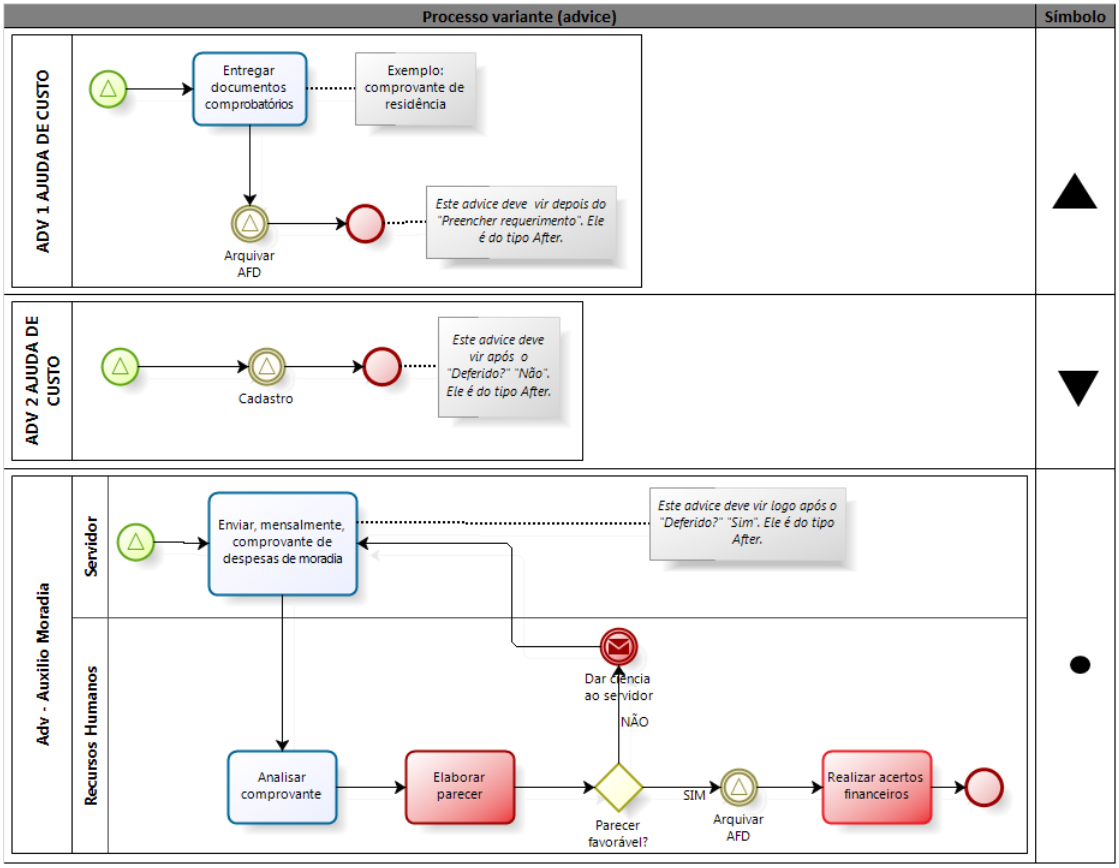


Figura 4.36: Processos Variantes (advices) do Processo Comum de Ajuda de Custo e Auxílio Moradia

CK	
Feature Expressions	**Transformations**
Ajuda de Custo	selectBusinessProcess "bpAjudaCustoComum" spl product evaluateAdvice "advAjudaCusto1" spl product evaluateAdvice "advAjudaCusto2" spl product
Auxilio Moradia	selectBusinessProcess "bpAjudaCustoComum" spl product evaluateAdvice "advAuxilioMoradia" spl product

Figura 4.37: Caso V - Complemento Salarial - Ajuda de Custo e Auxílio Moradia - CK Origional - Problema ao Executar evaluateAdvice vinculada à segunda feature

CK	
Feature Expressions	**Transformations**
Ajuda de Custo	selectEval "bpAjudaCustoComum" "advAjudaCusto1" spl product evaluateAdvice "advAjudaCusto2" spl product
Auxilio Moradia	selectEval "bpAjudaCustoComum" "advAuxilioMoradia" spl product

Figura 4.38: Caso V - Complemento Salarial - Ajuda de Custo e Auxílio Moradia - CK Corrigido - Criada Transformação selectEval

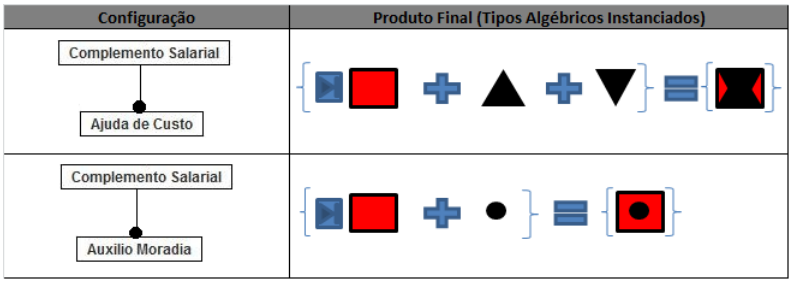


Figura 4.39: Configuração do Produto - Ilustração de PC para *features* isoladas

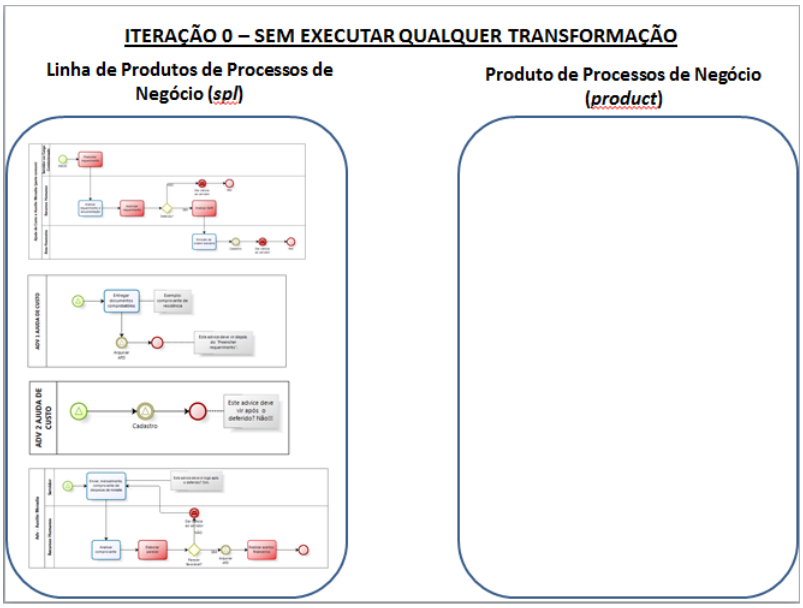


Figura 4.40: Caso V - Complemento Salarial - Problema - Iteração 0

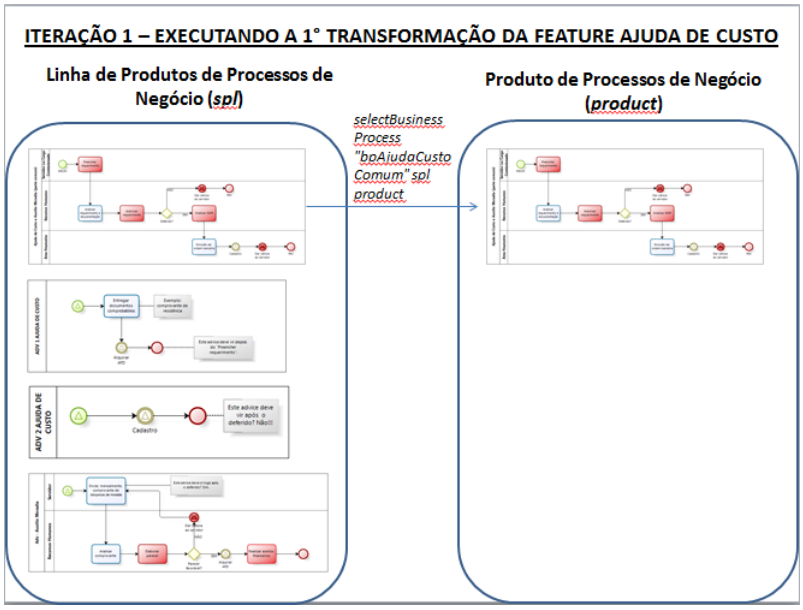


Figura 4.41: Caso V - Complemento Salarial - Problema - Iteração 1

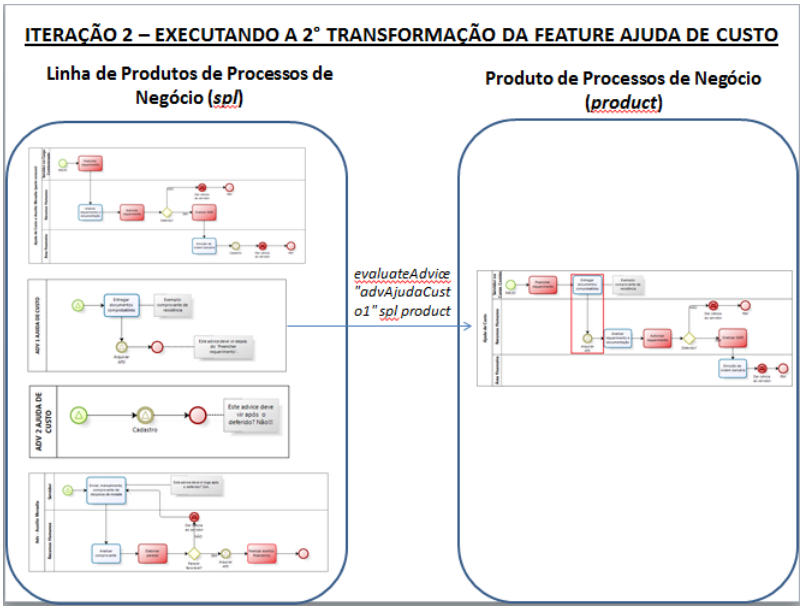


Figura 4.42: Caso V - Complemento Salarial - Problema - Iteração 2

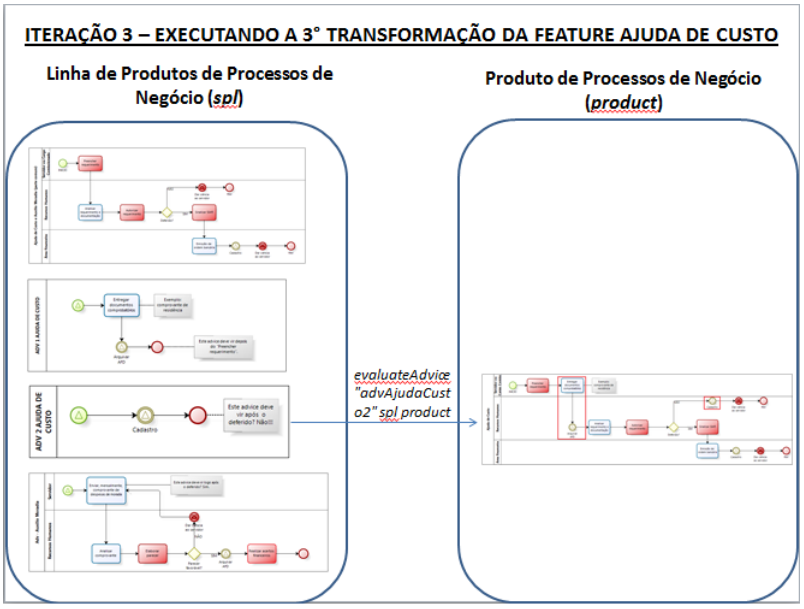


Figura 4.43: Caso V - Complemento Salarial - Problema - Iteração 3

esquerdo da figura, e adicionado no produto (*product*), lado direito da figura. Em seguida, ao executar *evaluateAdvice "advAjudaCusto1" spl product*, ilustrado na Figura 4.42, é feita a composição do processo-advice *advAjudaCusto1* com todos os processos do produto, no caso só há um, e, em seguida, ao executar *evaluateAdvice "advAjudaCusto2" spl product*, ilustrado na Figura 4.43, é feita a composição do processo-advice *advAjudaCusto2* com os processos do produto.

Em seguida, ilustrado na Figura 4.43, dentro do produto, temos o processo Ajuda de Custo com suas variabilidades resolvidas, figura que corresponde ao processo Ajuda de Custo original (Figura 4.33). Até aqui, este é o resultado parcial esperado, mas, ao executarmos a segunda linha do CK original (as transformações referentes à *feature* Auxilio Moradia), temos o seguinte problema: a transformação *selectBusinessProcess "bpAjudaCustoComum" spl product* da feature Auxilio Moradia irá copiar o processo *bpAjudaCustoComum* para o produto, ilustrado na Figura 4.44, e, em seguida, ao executar *evaluateAdvice "advAuxilioMoradia" spl product*, ilustrado na Figura 4.45, fará a composição do processo-advice *advAuxilioMoradia* com os processos do produto. Neste ponto, dentro do produto, há dois processos, o inserido há pouco e transformado pelas transformações da feature Ajuda de Custo e o inserido e transformado pelas transformações da feature Auxilio Moradia. Mas o problema é que, ao aplicar a transformação *evaluateAdvice "advAuxilioMoradia" spl product* da feature Auxilio Moradia, iteração ilustrada na Figura 4.45, serão realizadas composições em todos os processos do produto, logo, no processo *bpAjudaCustoComum* transformado que se refere ao processo Ajuda de Custo, será inserido o processo-advice *advAuxilioMoradia*, trecho do processo do produto ilustrado nessa figura com "X" em vermelho. Assim, Figura 4.45, teremos dentro do produto dois processos transformados, o processo *bpAjudaCustoComum* transformado que contem os processos-advices de ambas as *features*, primeiro processo do produto, de cima para baixo, e o processo *bpAjudaCustoComum* transformado que se refere ao processo Auxilio Moradia, segundo processo, que corresponde ao processo Auxilio Moradia apresentado

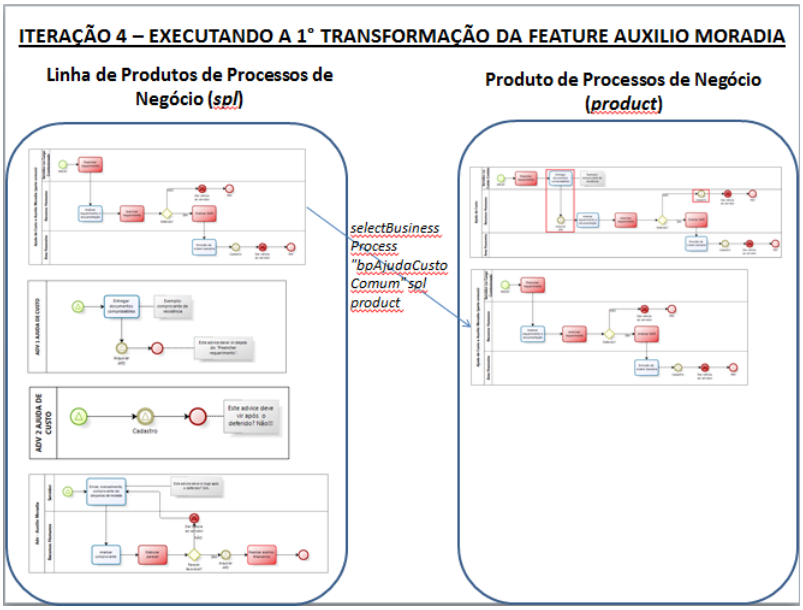


Figura 4.44: Caso V - Complemento Salarial - Problema - Iteração 4

na Seção 4.2.5. As instâncias dos tipos algébricos desse produto final incorreto estão em Apêndice, na Subseção D.5.3.

Agora explicaremos a correção para o problema mencionado. As Figuras 4.46, 4.47 e 4.48 apresentam ilustrações iterativas para a correção do problema a partir do uso da função *selectEval* apresentada no CK ajustado, Figura 4.38. Essa função se propõe a resolver o problema de composição em escopo global, similarmente ao Caso II (Subseção 4.2.2) para tratar variabilidade de valor. Neste caso, estamos tratando variabilidade em trechos de processos de negócios.

Apresentamos, em seguida, a transformação criada. A estrutura interna dessa função é similar à *selectBind*, com a diferença de executar *evaluateAdvice* ao invés de *bindParameter*. E, quanto os parâmetros de entrada só diferem da *evaluateAdvice* por causa do acréscimo do *bpId*, que serve para realizar a restrição de escopo, ou seja, viabilizar a seleção local (linha 2) do processo que se deseja transformar.

```
1 selectEval :: Id → Id → BusinessProcessModel → BusinessProcessModel →
      BusinessProcessModel
2 selectEval bpId advId spl product =
3   let process1 = selectBusinessProcess bpId spl (BPM {processes = []});
4       process2 = evaluateAdvice advId spl process1
5   in case (processes process2) of
6     [process] → process2
7     otherwise → BPM {processes = []}
```

A Figura 4.46 ilustra o que ocorre quando se executa a primeira transformação vinculada à feature Ajuda de Custo do CK ajustado, Figura 4.38. Vemos que essa transformação, *selectEval "bpAjudaCustoComum" "advAjudaCusto1" spl product*, realiza três operações: (i) seleciona o processo comum às *features* Ajuda de Custo e Auxílio Moradia (primeiro processo miniaturizado da spl), mostrado expandido na Figura 4.35; (ii) realiza a composição do primeiro processo-advice de Ajuda de Custo com o processo comum (primeiro processo miniaturizado da spl); e (iii) adiciona o resultado da operação (ii) no

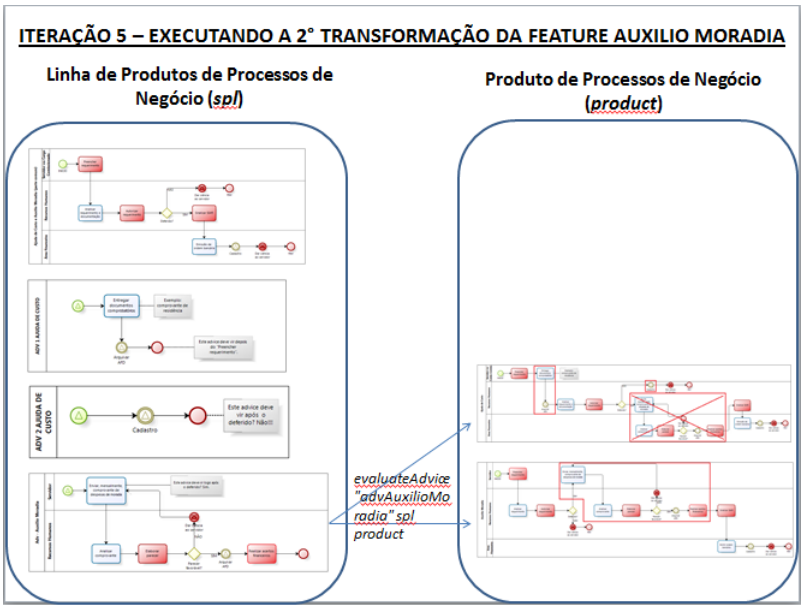


Figura 4.45: Caso V - Complemento Salarial - Problema - Iteração 5

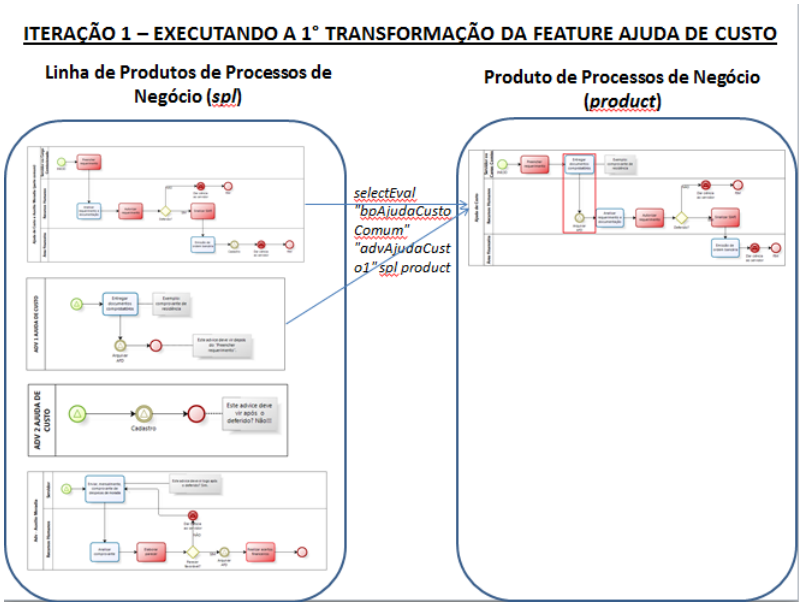


Figura 4.46: Caso V - Complemento Salarial - Solução - Iteração 1

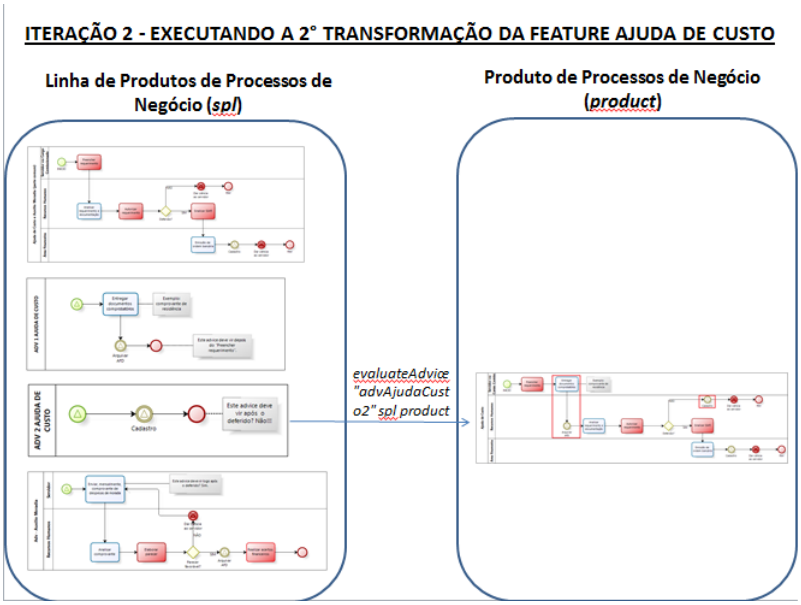


Figura 4.47: Caso V - Complemento Salarial - Solução - Iteração 2

produto (lado direito da figura). Como se pode perceber, no processo adicionado no produto, o trecho que foi composto com a parte comum está destacado para evidenciar o que foi feito.

Já a Figura 4.47 ilustra o que ocorre quando se executa a segunda transformação vinculada à feature Ajuda de Custo do CK ajustado (Figura 4.38). Vemos que essa transformação, *evaluateAdvice "advAjudaCusto2" spl product*, realiza duas operações: (i) seleciona o segundo processo-advice da *features* Ajuda de Custo contido na spl (lado esquerdo da figura); e (ii) realiza a composição desse processo-advice com o(s) processo(s) contido(s) no produto, no caso, só há um processo, o mostrado na iteração anterior. Como se pode perceber, no processo que foi atualizado no produto, o trecho que foi composto com o processo da iteração anterior está destacado para evidenciar o que foi feito (atividade "Cadastro"). O resultado é idêntico com o processo Ajuda de Custo original, mostrado expandido na Figura 4.33, e, portanto, corresponde ao esperado.

A Figura 4.48 ilustra o que ocorre quando se executa a única transformação vinculada à feature Auxílio Moradia do CK ajustado, Figura 4.38. Vemos que essa transformação, *selectEval "bpAjudaCustoComum" "advAuxilioMoradia" spl product*, realiza três operações: (i) seleciona o processo comum às *features* Ajuda de Custo e Auxílio Moradia (primeiro processo miniaturizado da spl), mostrado expandido na Figura 4.35; (ii) realiza a composição do único processo-advice de Auxílio Moradia com o processo comum; e (iii) adiciona o resultado da operação (ii) no produto (lado direito da figura). Como se pode perceber no processo adicionado no produto, o trecho que foi composto com a parte comum está destacado para evidenciar o que foi feito. As instâncias dos tipos algébricos (produto final desejado) deste caso está em Apêndice, na Subseção D.5.4.

Na Figura 4.49, mostramos a configuração com as duas *features* selecionadas, mas ilustrando o problema corrigido. Como se pode perceber nessa figura, o símbolo "+" continua sendo utilizado mesmo a transformação *"evaluateAdvice"* tendo sido substituída pela *"selectEval"*. Mantivemo-lo porque essa nova transformação executa *"evaluateAdvice"*

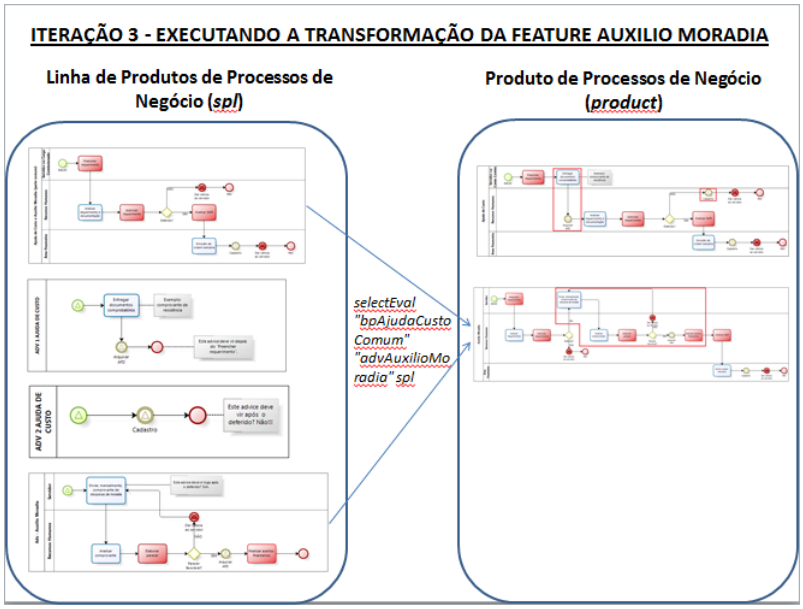


Figura 4.48: Caso V - Complemento Salarial - Solução - Iteração 3

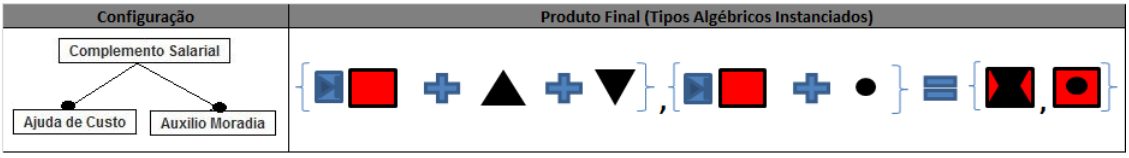


Figura 4.49: Configuração do Produto - Ilustração de PC com as duas *features* selecionadas - problema corrigido

internamente.

Análise do Resultado

Identificamos neste caso problema similar ao identificado no caso Caso II (Seção 4.2.2) e foi resolvido com o mesmo conceito de restrição de escopo de aplicação da transformação. Portanto, a transformação *selectEval* foi criada e sua estrutura interna é semelhante à proposta no Caso II.

4.3 Considerações Finais

Identificamos que os 12 casos avaliados poderiam ser classificados em quatro casos ortogonais, apresentados na Figura 4.50. Há casos em que os processos são iguais, exceto o nome dos processos (primeiro caso ortogonal) e, além disso, possui variação de valor (segundo caso ortogonal). Há outros casos (terceiro caso ortogonal) em que um dos processos desempenha o papel de processo-comum para o par de processos. Em ambos casos um dos processos desempenhará o papel do processo-base (processo-comum que será referência para resolver as variabilidades, quando houver). Uma decisão de engenharia do domínio Czarnecki e Eisenecker (2000) foi eleger como processo-base, para o primeiro e segundo caso ortogonal, um dos processos para compor o *BusinessProcessModel*. Para o

Descrição de Ortogonalidade	Casos Avaliados
Processos Idênticos - Exceto Nome	Caso I - Transferência - Cessão e Requisição
Processos Idênticos - Exceto Nome e Variação de Valor	Caso II - Radiação - Raio-X ou Substância Radioativa e Irradiação Ionizante Caso VII - Adicional - Periculosidade e Insalubridade
Um Processo Comum e o Outro Variante	Caso III - Desligamento - Demissão e Aposentadoria Caso IV - Ouvidoria - Reclamação e Elogio Caso VIII - Auxílio - Auxílio Alimentação e Auxílio Transporte Caso IX - Auxílio Filiação - Auxílio Natalidade e Auxílio Pré-Escolar
Extraído o Processo Comum e os Outros São Variantes	Caso V - Complemento Salarial - Ajuda de Custo e Auxílio Moradia Caso VI - Licença - Licença Incentivada sem Remuneração e Licença para Capacitação Caso X - Férias - Marcar e Férias e Remarcar Férias Caso XI - Serviço - Serviço Extraordinário e Tempo de Serviço Quinquênio Caso XII - Indenização - Indenização de Transporte e Transporte de Mobiliário e Bagagens

Figura 4.50: Visão Ortogonal dos Casos Avaliados

terceiro caso ortogonal, o processo-base é o processo-comum, também inserido no *BusinessProcessModel*.

Constatamos que a solução proposta inicialmente fornecia suporte a 58% (7 de 12) dos problemas relacionados a tratamento de variabilidade em processos de negócio. Foi necessário criar mais duas transformações para que os problemas retratados pelos demais casos - 42% (5 de 12) - fossem resolvidos, que é o último caso ortogonal apresentado na Figura 4.50. Essas duas novas transformações implementaram o conceito de composição ou parametrização em escopo local devido aos problemas de composição ou parametrização em escopo global descritos nos Casos 4.2.2 e 4.2.5. Além de estender a ferremanta *Hephaestus*, a implementação desse conceito é uma contribuição a esse trabalho, conforme documentado em trabalhos relacionados.

Mostramos neste capítulo apenas os casos ortogonais, exceto o caso IV. Apresentamos apenas esse caso a mais porque queríamos ressaltar as alternativas descritas a seguir. Os outros casos avaliados, que são os que se encaixam na ortogonalidade apresentada, estão no Apêndice B.

Ressaltamos que, no *Caso IV - Ouvidoria - Reclamação e Elogio* (Subseção 4.2.4), poderíamos ter resolvido a variabilidade de duas formas: (i) considerar um dos processos como o processo comum e fazer a composição; ou (ii) extrair o processo comum aos dois processos (Reclamação e Elogio) e realizar as composições. A primeira alternativa se encaixa na terceira visão ortogonal e a segunda alternativa na quarta visão. A vantagem da primeira alternativa (nossa escolha) é de termos menor esforço de composição já que possui um núcleo comum maior do que o da segunda alternativa. No entanto, a desvantagem é que o núcleo comum maior não é totalmente reaproveitado já que tivemos que substituir as atividades *"Enviar Elogio"* e *"Incluir Elogio"* por processos-advices que continham apenas uma atividade cada, "Enviar Reclamação" e "Incluir Reclamação". Já com a segunda alternativa, temos como desvantagem um núcleo comum menor, maior esforço de composição, mas sua vantagem é que teríamos um processo-advice de granularidade mais grossa já que as atividades "Enviar Reclamação" e "Incluir Reclamação" estariam em um único processo-advice. Ou seja, a vantagem e desvantagem dessas duas alternativas são inversas. Com a primeira alternativa, modelamos quatro processos para compor o *BusinessProcessoModel* - o processo comum (*"bpElogioComum"* - reaproveitado integralmente) e três processos-advices (*"advReclamacao1"*, *"advReclamacao2"* e *"advReclamacao3"*). Com a segunda alternativa, teríamos três processos compondo o *BusinessProcessModel* - o processo

comum (que seria extraído dos dois - trabalho adicional) mais dois processos-advices (um que condensaria os advices *"advReclamacao1"* e *"advReclamacao2"* em um só e o *"advReclamacao3"*).

Não foi tratado quantificação Filman e Friedman (2000) nos casos avaliados. No entanto, percebemos que ela existe, quanto às atividades "Realizar acertos financeiros" (ocorre em 9 dos 12 casos) e "Elaborar portaria" (ocorreu em 4 dos 12 casos) isoladamente, ou seja, não há qualquer vínculo entre elas. Não se justificou criarmos processo-advice para cada uma dessas atividades dado o baixo grau de reuso. Apesar de não termos aplicado o princípio de quantificação para esses casos, as transformações propostas (Seção 3.2.2) dão suporte a ele. Mas, dado o domínio avaliado (processos de negócio de Recursos Humanos da Administração Pública Federal), há um tipo de quantificação que pode ser explorada: processo(s)-advice(s) do Tribunal de Contas da União (TCU) e/ou da Controladoria Geral da União (CGU) poderi(am), por meio de seus *pointcuts descriptors* (PCDs), interceptar o(s) processo(s) de negócio de seus interesses para realizar auditoria. Isso seria possível se houvesse posição(ões) regulare(s) nos processos de negócio do domínio de RH que pudesse(m) ser anotada(s) *(joinpoints)*. Comentamos "posição(ões) regulare(s)" porque há posições dos processos base que não podem ser anotadas. É o caso dos elementos *Start* e *End* do tipo algébrico *FlowObject* (Seção 3.2.1) que não têm o campo *annotations*. Não permitir esse recurso para esses elementos foi uma decisão de projeto, visto que não havíamos identificado essa necessidade no domínio de RH e acrescentá-los, sem justificativa, exigiria mais controles, como, por exemplo, não permitir composição de um processo-advice do tipo *before* em um *Start* anotado no processo-base. Considerando que os demais elementos do *FlowObject*, *Activity* e *Gateway*, permitem anotação, ainda que não houvesse posições regulares (elementos específicos dos processos-base) de interesse para se realizar uma auditoria, como o *pointcut* proposto é semântico (Seção 3.2.3), poder-se-ia anotar qualquer um desses elementos e, via os PCDs dos órgãos citados, ocorreria a quantificação. Isso daria o trabalho manual de anotar cada ponto de interesse, mas seria viável. Apesar de estar fora do escopo deste trabalho, citamos esse exemplo apenas para ilustrar outra possível aplicação de nossa abordagem. Decidimos não explorar quantificação referente às atividades "Realizar acertos financeiros" e "Elaborar portaria" porque o grau de reuso dos processos-advice seria baixo no conjunto como um todo dos casos avaliados. Esse baixo aproveitamento identificado ocorre porque, dado a não regularidade da posição em que essas atividades ocorrem nos processos-base, teríamos que criar *advices* específicos. Por exemplo, há casos a atividade "Realizar acertos financeiros" ocorre antes de determinado objeto e em outros ocorre após. Ou seja, para certos casos seria exigido *advice*, com a mesma estrutura, do tipo *before* e, para outros, do tipo *after*. Portanto, o reuso que identificamos ficou restrito entre os processos que fazem parte de cada caso (par de processos de negócio) - 46% (24 de 52, conforme descrito na Seção 4.2). Nesses 12 pares avaliados, constatamos reuso de granularidade baixa (até três objetos em comum) - em dois casos, Casos IX(Seção B.4) e Caso XII(Seção B.7), médio (de quatro a seis objetos em comum) - em dois casos, Casos VI(Seção B.1) e X(Seção B.5) - e alta (acima de seis objetos em comum) - nos demais Casos (oito), Casos I-V(Seção 4.2.1- 4.2.5), Casos VII-VIII(Seção B.2- B.3) e Caso XI(Seção B.6).

Como mencionado na Seção 2.1, um dos desafios da Linha de Produtos de Software (LPS) é gerenciar as variabilidades da Linha de Produtos (LP). Especificamente, esse desafio se desdobra em dois: (i) melhor forma de documentar as variabilidades; e (ii) a

melhor forma de compor tais variabilidades com os *core assets* da LP. O Desenvolvimento de Software Orientado a Aspectos (DSOA) é uma das opções para apoiar nesses desafios. Nesse contexto, diversas contribuições Sullivan et al. (2005); Griswold et al. (2006); Kulesza et al. (2006a,b) foram propostas para aperfeiçoar o DSOA quanto à modularidade. Esses quatro trabalhos estão centrados na proposta de *Crosscut Programming Interfaces* (XPIs), que propõem uma interface de ocultamento de informação (*information hiding*) para DSOA. Genericamente, no contexto de LPS, EJPs (implementação de XPIs para LP) é uma interface que expõe os pontos variantes da LP aos artefatos-variantes dela. Ou seja, ela serve como um contrato entre esses dois tipos de artefatos. Há vários benefícios, conforme Griswold et al. (2006) elicita. Um dos benefícios centrais é o de reduzir o acoplamento (dependência) entre essas duas classes de artefatos. No contexto deste trabalho, EJP seria uma interface entre os processos-de-negócio-base (que contêm os (*jointpoints*)) e os processos-variantes (*advices*, que contêm os PCDs), que formam a LP de Processos de Negócios (*BusinessProcessModel*), (*core assets*) desta proposta. Como um dos principais benefícios apontados na literatura de LPS é o de reuso estratégico de artefatos e reuso está intrinsicamente ligado a uma boa proposta de modularidade, EJP é uma potencial solução para viabilizar esse último conceito visto que promove a diminuição do acoplamento entre os artefatos da LP. Essas considerações se fazem importantes neste trabalho visto que faz parte de trabalhos futuros melhorar a modularidade entre os artefatos do *BusinessProcessModel*. Apesar de termos implementado o conceito de *pointcuts* semânticos(Capítulo 3, Seção 3.2.3), para evitar a quebra dos PCDs após alguma manutenção nos processos-base, e assim tornar o reuso mais consistente, percebemos que implementar EJPs se faz necessário em nossa proposta. Além disso, baseado na imposição de restrições (*constraints*) por meio de *Extension Join Points (EJPs)* Kulesza et al. (2006a,b), poderíamos vincular restrições aos processos-advices, que implementam EJPs, já que sua aplicação nos processos-base é contextual/posicional (não regularidade descrita anteriormente). Isso geraria maior consistência na modularidade, tornaria os processos-advices mais reutilizáveis, possibilitando assim sua quantificação. Por fim, não é opinião conclusiva que a modularidade proposta seja viável apenas para pares de processos - já que assim identificamos para o domínio estudado (RH), ou que ela seja genérica o suficiente para tratar boa parte das variabilidades em processos de negócios. Acreditamos que, para outros domínios de processos de negócio, pode-se constatar maior amplitude de reuso do que em pares de processos.

Quanto ao *Caso XI - Serviço - Serviço Extraordinário e Tempo de Serviço Quinquênio* obtemos baixo grau de reuso (apenas duas atividades) por causa da decisão de projeto quanto à simpificação na estrutura do processo-advice, descrito na Seção 3.2.3). A simplificação que fizemos foi de permitir que os processos-advices tenham apenas um elemento de início (*Start*) e de fim(*End*). Esses elementos representam os vínculos(antecessor e sucessor) do processo-advice quanto ao processo-base, tendo como referência o *FlowObject* anotado(*jointpoint*) no processo-base, para efeitos de composição entre processos (para todos os tipos de advice - *before*, *after* e *around*). Se quiséssemos aumentar a granularidade do processo-advice de cada processo desse caso, teríamos que: (i) encontrar uma forma de remodelar os processos originais sem alterar a semântica(não percebemos essa viabilidade para esse caso, apesar dela poder existir); ou (ii) permitir que a estrutura dos processos-advices tenham mais de um objeto de início e/ou de fim (fora do escopo deste trabalho). Portanto, para tratar a variabilidade deste caso, nossa única alternativa

foi modelar processos-variantes de granularidade alta, mas que contêm em si elementos comuns aos dois processos - “Analisar processo”, “Elaborar parecer”, “Parecer favoravel?”, “Dar ciência ao servidor” e “Cadastro”. Dessa forma, obtivemos um baixo grau de reuso do processo-comum, mas obtivemos composições mais simples (apenas um processo-advice para cada processo de negócio).

Capítulo 5

Conclusões, Trabalhos Relacionados e Trabalhos Futuros

Neste capítulo resumimos as contribuições desta dissertação (Seção 5.1), apresentamos os trabalhos relacionados (Seção 5.2) e elencamos os trabalhos futuros (Seção 5.3).

5.1 Contribuições

Nesta dissertação propomos uma técnica e ferramenta para tratar variabilidade em uma Linha de Produtos de Processos de Negócio (LPN) utilizando uma abordagem paramétrica e composicional com OA Machado et al. (2011). Ela aproveita e estende uma ferramenta existente. Essa proposta foi avaliada a partir de processos de negócio do domínio de Recursos Humanos (RH). Tal avaliação foi uma Pesquisa-Ação (*Action Research*) utilizando método empírico de investigação e validação e, para guiá-la, assim como à extensão mencionada, utilizamos um método. Os resultados obtidos ao aplicar esse método bem como à avaliação foram:

1. Caracterizamos comunalidade e variabilidade em processos de negócio do domínio de RH utilizando processo extrativo. Isso viabilizou a elaboração do modelo de domínio (FM).

2. Doze casos representados pelas *features* do FM foram avaliados. Foi apresentada uma classificação ortogonal para eles.

3. Constatamos que a solução proposta inicialmente fornecia suporte a 42% (5 de 12) dos problemas relacionados a tratamento de variabilidade em processos de negócio. Foi necessário criar mais duas transformações para que os problemas retratados pelos demais casos - 58% (7 de 12) - fossem resolvidos. Essas soluções contribuiram para o principal trabalho relacionado Bonifácio e Borba (2009) para, não só estender sua proposta, mas propor uma solução para uma limitação já descrita nesse trabalho relacionado.

4. Não foi tratado quantificação para os casos avaliados. No entanto, percebemos que ela existe para algumas atividades dos processos do domínio avaliado. Escolhemos

não tratá-la dado o baixo grau de reuso que teríamos. Apesar de não termos aplicado o princípio de quantificação para esses casos, as transformações propostas dão suporte a ele. Destacamos que, apesar de fora do escopo deste trabalho, nossa abordagem poderia ser utilizada, por exemplo, para órgãos de controle da administração federal (TCU e/ou CGU) interceptarem, por meio de *PCDs*, os processos de negócio dos órgãos que fossem de interesse para uma auditoria.

5. O reuso que identificamos ficou restrito entre os processos que fazem parte de cada caso (par de processos de negócio) - 46% (24 de 52). Nesses 12 pares avaliados, constatamos reuso de granularidade baixa (em nível de *FlowObjects* - para composições, e em nível de parâmetros internos aos *FlowObjects* no caso de parametrização).

6. Identificamos que, para melhorar a modularidade entre os artefatos do *BusinessProcessModel* e proporcionar maior consistência na composição entre eles, EJP é uma potencial solução. Sua implementação pode melhorar a modularidade já que promove a diminuição do acoplamento entre os artefatos da LP bem como fornece recursos de imposição de restrições *(constraints)* entre os elementos participantes de composição (processos-base e processos-advice).

7. Na avaliação, identificamos que o reuso das variabilidades nos processos de negócio do domínio de RH se restringiu entre processos que fazem parte de par de processos (12 par de processos = 24 processos) - 46% (24 de 52). No entanto, não é opinião conclusiva que o reuso que pode ser proporcionado por nossa abordagem se resuma a esse padrão. Logo, outras avaliações se fazem necessárias para termos maior precisão quanto ao grau de reuso de nossa abordagem.

5.2 Trabalhos Relacionados

Nesta seção, são apresentados os trabalhos relacionados com esta pesquisa. Iniciamos apresentando a contribuição realizada na infraestrutura estendida por este trabalho (Seção 5.2.1). Apresentamos as contribuições em relação ao trabalho de configuração de processos de negócio (Seção 5.2.2), uma proposta de extensão de BPMN utilizando DSOA (Seção 5.2.3) e uma abordagem para tratar variabilidade em uma linha de produtos de contratos eletrônicos (Seção 5.2.4).

5.2.1 Modelando Variabilidade em uma Linha de Produto de Software de Cenários de Caso de Uso - Uma Abordagem Baseada em Mecanismos Transversais

Bonifácio et al. Bonifácio e Borba (2009) propõe a ferramenta *Hephaestus*, que utiliza uma abordagem paramétrica e composicional baseada em abordagem orientada a aspectos para modelar variabilidade em uma LPS de cenários de caso de uso. Estender essa ferramenta foi um dos objetivos deste trabalho. Mas, além disso, realizamos uma contri-

buição referente ao problema registrado na Seção *Specifying Configuration Knowledge*[1]) de Bonifácio. Nessa seção, descrito previamente, que mesmo considerando que a ordem do CK em *Hephaestus* é relevante para a montagem do produto final, é informado que essa ferramenta pode gerar produtos inconsistentes quando executa transformações vinculadas a *features* não-alternativas *(or-inclusive)* (ambas selecionadas na configuradas produto) onde os *advices* referenciados por essas transformações compartilham o mesmo cenário de caso de uso (vinculado a uma *feature* também selecionada no PC). Para essa limitação, nossa contribuição foi de restringir o escopo da composição e da parametrização, conforme descrito no Capítulo 4. Viabilizamos isso com a criação das transformações *selectEval...* e *selectBind...*, respectivamente. Quando avaliamos nossa abordagem, constatamos que essa contruição resolveu 42% (5 de 12) dos problemas causados pela limitação descrita.

5.2.2 Configuração de Processos de Negócio

La Rosa et al. Rosa et al. (2011) propõe um conjunto de métodos e ferramentas para o desenvolvimento de processos baseados no modelo de processo configurável. Essa proposta aborda a variabilidade nos conectores, decisões *(gateways)*, atividades e recursos organizacionais (pessoas ou objetos). Ao contrário da nossa abordagem, que é paramétrica e composicional, a sua abordagem é anotativa, basicamente envolvendo a variabilidade com *gateways* adicionais direcionados pelo modelo de decisão, tumultuando o modelo de processo com detalhes de mapeamento entre expressões de *features* e transformações - *Configuration Knowledge (CK)* - e, portanto, não fornecendo meios para separar os espaços de solução e configuração Czarnecki e Eisenecker (2000) e impedindo a reutilização da variabilidade. Desconsiderando as excepcionalidades descritas no Capítulo 4, na Seção 4.3, foi possível reutilizar 100% das variabilidades entre os pares de processos avaliados.

5.2.3 Abordagem Orientada a Aspectos para Modelar Processos de Negócio

Cappelli et al Cappelli et al. (2009) propõe uma abordagem orientada a aspectos para modularizar interesses transversais no processo de modelagem de negócios. Uma extensão de BPMN é proposta para expressar interesses transversais através da inserção de processos transversais (representado por atividades) e as relações transversais (representada por fluxos entre as atividades). Ao contrário de nossa abordagem, a sua abordagem não contempla questões de variabilidade em uma LPN, portanto, não concentrando em CK nem no modelo do domínio. Dessa forma, essa limitação não viabiliza o reuso das variabilidades de artefatos de uma LPS. Nós fornecemos uma semântica operacional dos tipos de variações *(advices types)*, aproveitando e ampliando uma infraestrutura existente, incluindo uma ferramenta. Apesar de o reuso das variabilidades que identificamos, ao avaliarmos os processos de negócio do domínio de RH (Capítulo 4), terem se restringido entre processos que fazem parte par de processos (12 par de processos = 24 processos) - 46% (24

[1]"However, if two non alternative advice share join points, we have to declare precedence among them. We have decided to declare this kind of precedence as an independent asset, which is taken into account when generating products, because the precedence might also depend on each product. In addition, the notion of provided and required interfaces for the configuration knowledge, as proposed by Teixeira [91], could be used to identify CK errors."

de 52, conforme descrito na Seção 4.2), não é opnião conclusiva (conforme argumentado na Seção 4.3) que o reuso que pode ser proporcionado por nossa abordagem se resuma a esse padrão.

5.2.4 Uma Linha de Produto para Gerenciamento de Processo de Negócio de Elaboração de e-Contratos

Gimenes et al de Souza Gimenes et al. (2008) propõe uma abordagem para tratar variabilidade em uma linha de produtos de contratos eletrônicos (*WS-Contract*). Sua proposta contempla o uso dos artefatos centrais preconizados em LPS - *Model (Ativos da organização), FM, CK e PC* e considera um processo de negócio para a elaboração desses contratos, estabelecimento ("assinatura") deles eletronicamente bem como cumprimento de acordos de níveis de serviço *(SLAs)* também de forma eletrônica. No entanto, está fora do escopo dessa proposta o gerenciamento de variabilidade e comunalidade em artefatos de processos de negócio.

5.3 Trabalhos Futuros

Como trabalhos futuros, temos as seguintes atividades:

(a) Tratar variabilidade em demais elementos de BPMN (raias, notas/texto, *MessageFlow*, entre outros) nos processos de negócio.

(b) Realizar avaliação da abordagem proposta em outros domínios. Nesses momentos, usar as funções de *parser* para converter processos de negócio modelados em BPMN para os tipos algébricos propostos.

(c) Implementar verificador de tipos para garantir a consistência na elaboração dos processos de uma LPN bem como para os processos finais contidos no produto.

(d) Implementar EJPs para melhorar a modularidade entre os artefatos do *BusinessProcessModel*.

(e) Realizar avaliação empírica quantitativa utilizando e propondo métrica, tais como: Métricas de Modularidade (MM); Espalhamento do Conhecimento de Configuração (ESCC); Entrelaçamento do Conhecimento de Configuração (ENCC); Métricas de Complexidade (MC); Métricas de Estabilidade (ME)

(f) Implementar função *show* para o tipo *transition* para ocultar a lista de parâmetros de seus objetos.

(g) Integrar à *Hephaestus* as novas transformações *(selectBind e selectEval)*.

Referências

Victor R. Basili, Gianluigi Caldiera, e H. Dieter Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

Rodrigo Bonifácio e Paulo Borba. Modeling scenario variability as crosscutting mechanisms. In *Proc. of AOSD '09*, pages 125–136. ACM, 2009.

Claudia Cappelli, Julio Leite, Thais Batista, e Lyrene Silva. An aspect-oriented approach to business process modeling. In *Proc. Early Aspects*, pages 7–12. ACM, 2009.

Paul Clements e Linda M. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.

Krysztof Czarnecki e Ulrich Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley Professional, June 2000. ISBN 0201309777.

Thomas H. Davenport. *Process Innovation: Reengineering Work Through Information Technology*. 1993.

Itana Maria de Souza Gimenes, Marcelo Fantinato, e Maria Beatriz Felgar de Toledo. A product line for business process management. In *SPLC*, pages 265–274, 2008.

Marlon Dumas, Wil van der Aalst, e Arthur H. ter Hofstede. *Process-aware information systems: bridging people and software through process technology*. John Wiley and Sons, 2005.

Steve Easterbrook. Empirical research methods for software engineering. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, ASE '07, pages 574–574, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-882-4. doi: http://doi.acm.org/10.1145/1321631.1321749. URL `http://doi.acm.org/10.1145/1321631.1321749`.

Robert E. Filman e Daniel P. Friedman. Aspect-oriented programming is quantification and obliviousness. pages 21–35. Addison-Wesley, 2000.

William G. Griswold, Kevin J. Sullivan, Yuanyuan Song, Macneil Shonle, Nishit Tewari, Yuanfang Cai, e Hridesh Rajan. Modular software design with crosscutting interfaces. *IEEE Software*, 23:51–60, 2006. doi: 10.1109/MS.2006.24.

Jilles Van Gurp, Jan Bosch, e Mikael Svahnberg. On the notion of variability in software product lines. In *Proceedings of the Working IEEE/IFIP Conference on Software*

Architecture, WICSA '01, pages 45–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1360-3. doi: http://dx.doi.org/10.1109/WICSA.2001.948406. URL http://dx.doi.org/10.1109/WICSA.2001.948406.

Michael Hammer e James Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness, April 1994a. ISBN 088730687X. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/088730687X.

Michael Hammer e James Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness, April 1994b. ISBN 088730687X. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/088730687X.

Paul Harmon. *Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes, Morgan Kaufmann*. Morgan Kaufmann, January 2003. ISBN 1558607587. URL http://www.amazon.com/Paul-Harmon/e/B001IGJRWW/ref=ntt_athr_dp_pel_1.

Paul Harmon, Michael Rosen, e Michael Guttman. *Developing E-Business Systems & Architectures: A Manager's Guide*. Morgan Kaufmann, November 2000. ISBN 1558606653.

Michael Havey. *Essential Business Process Modeling*. O'Reilly Media, Inc., 2005. ISBN 0596008430.

Paul Hudak. Building domain-specific embedded languages. *ACM Comput. Surv.*, 28, December 1996. ISSN 0360-0300.

John Jeston e Johan Nelis. *Business process management: practical guidelines to successful implementations*. Butterworth-Heinemann, 2006.

Simon Peyton Jones et al. The haskell 98 report (revised). Technical report, Cambridge University Press, 2002.

Christian Kästner, Sven Apel, e Martin Kuhlemann. Granularity in software product lines. In *Proceedings of the 30th international conference on Software engineering*, ICSE '08, pages 311–320, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-079-1. doi: http://doi.acm.org/10.1145/1368088.1368131. URL http://doi.acm.org/10.1145/1368088.1368131.

G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J. M. Loingtier, e J. Irwin. Aspect-oriented programming. In *ECOOP '97: Proceedings of European Conference on Object-Oriented Programming*, pages 220–242. Springer-Verlag, 1997.

Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, e William G. Griswold. An overview of aspectj. In *Proceedings of the 15th European Conference on Object-Oriented Programming*, ECOOP '01, pages 327–353, London, UK, UK, 2001a. Springer-Verlag. ISBN 3-540-42206-4. URL http://portal.acm.org/citation.cfm?id=646158.680006.

Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, e William G. Griswold. Getting started with aspectj. *Communications of the ACM*, 44:59–65, 2001b.

Charles Krueger. Easing the transition to software mass customization. In *Proceedings of the 4th International Workshop on Software Product-Family Engineering.*, pages 282–293, Bilbao, Spain, October 2001.

Charles W. Krueger. Easing the transition to software mass customization. In *PFE '01: Revised Papers from the 4th International Workshop on Software Product-Family Engineering*, pages 282–293, London, UK, 2002. Springer. ISBN 3-540-43659-6. URL `http://portal.acm.org/citation.cfm?id=748909`.

Uirá Kulesza, Vander Alves, Alessandro F. Garcia, Carlos José Pereira De Lucena, e Paulo Borba. Improving extensibility of object-oriented frameworks with aspect-oriented programming. In *International Conference on Software Reuse*, pages 231–245, 2006a. doi: 10.1007/11763864_17.

Uirá Kulesza, Roberta Coelho, Vander Alves, Alberto Costa Neto, Alessandro Garcia, Carlos Lucena, Arndt von Sta, e Paulo Borba. Implementing framework crosscutting extensions with ejps and aspectj. 2006b.

Idarlan Machado, Rodrigo Bonifácio, Vander Alves, Lucinéia Turnes, e Giselle Machado. Managing variability in business processes: an aspect-oriented approach. In *Proceedings of the 2011 international workshop on Early aspects*, EA '11, pages 25–30, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0645-4. doi: http://doi.acm.org/10.1145/1960502.1960508. URL `http://doi.acm.org/10.1145/1960502.1960508`.

Business process modeling notation (bpmn) specification. Object Management Grou, 03 2009.

M. La Rosa, M. Dumas, A.H.M. ter Hofstede, e J. Mendling. Configurable multi-perspective business process models. *Information Systems*, 26(2), 2011.

O. A. El Sawy. *Redesignig Enterprise Process for e-business.* McGraw-Hill Higher Education, 2001.

Maximilian Störzer e Christian Koppen. Pcdiff: Attacking the fragile pointcut problem, abstract. In *European Interactive Workshop on Aspects in Software*, Berlin, Germany, September 2004. URL `http://pp.info.uni-karlsruhe.de/uploads/publikationen/stoerzer04eiwas.pdf`.

Kevin Sullivan, William G. Griswold, Yuanyuan Song, Yuanfang Cai, Macneil Shonle, Nishit Tewari, e Hridesh Rajan. Information hiding interfaces for aspect-oriented design. In *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, ESEC/FSE-13, pages 166–175, New York, NY, USA, 2005. ACM. ISBN 1-59593-014-0. doi: http://doi.acm.org/10.1145/1081706.1081734. URL `http://doi.acm.org/10.1145/1081706.1081734`.

Apêndice A

Código Fonte

module BusinessProcess where

```
import List

type Id = String
type Name = String
type Condition = String
type Annotation = String
type Parameter = (Name,Value)
type Transition = (FlowObject, FlowObject, Condition)

data BusinessProcessModel =
  BPM { processes :: [BusinessProcess] } deriving(Show)

data BusinessProcess = BusinessProcess {
   pid :: Id,
   ptype :: ProcessType,
   objects :: [FlowObject],
   transitions :: [Transition]
} deriving(Show)

data ProcessType =
  BasicProcess
  | Advice {advType :: AdviceType, pc :: Pointcut}
  deriving(Show)

data AdviceType = Before | After | Around
                  deriving(Eq, Show)

data Pointcut = Pointcut String | PCut String [String] |
                         Empty
                deriving(Show)

data FlowObject =
```

```
  FlowObject {id' :: Id, type' :: FlowObjectType,
              annotations' :: [Annotation],
              parameters' :: [Parameter]}
  | Start
  | End
  | Proceed

data FlowObjectType = Activity | Gateway
  deriving(Eq, Show)

data Value =
  Unbound | Value String
  deriving(Eq)

idflow :: FlowObject -> String
idflow (FlowObject i _ _ _) = i
idflow Start = "start"
idflow End = "end"
idflow Proceed = "proceed"

annotations :: FlowObject -> [Annotation]
annotations (FlowObject _ _ as _) = as
annotations Start = []
annotations End = []
annotations Proceed = []

parameters :: FlowObject -> [Parameter]
parameters (FlowObject _ _ _ p) = p
parameters Start = []
parameters End = []
parameters Proceed = []

instance Eq BusinessProcess where
 BusinessProcess i _ _ _  ==
   BusinessProcess j _ _ _ = i == j

instance Eq FlowObject where
 FlowObject i _ _ _ == FlowObject j _ _ _ = i == j
 Start == Start = True
 End == End = True
 Proceed == Proceed = True
 _ == _  = False

pointcut :: BusinessProcess -> Pointcut
pointcut adv =
 case ptype adv of
```

```
   (Advice _ p) -> p
   otherwise -> Empty

selectBusinessProcess :: Id -> BusinessProcessModel
                            -> BusinessProcessModel
                            -> BusinessProcessModel
selectBusinessProcess id spl product =
 let bps = [bp | bp <- (processes spl),
                      (pid bp == id),
                      bp `notElem` (processes product)]
 in case bps of
   [bp] -> BPM ([bp] ++ (processes product))
   [] -> product

selectBind :: Id -> BusinessProcessModel -> Name -> Value
                 -> BusinessProcessModel
selectBind bpId spl np vp =
 let process1 = selectBusinessProcess bpId spl (BPM {
                  processes = []});
     process2 = bindParameter np vp process1
 in case (processes process2) of
   [process] -> process2
   otherwise -> BPM {processes = []}


bindParameter :: Name -> Value -> BusinessProcessModel
                 -> BusinessProcessModel
bindParameter np vp product =
 BPM ( map (bindParameter' np vp) (processes product) )
 where
  bindParameter' np vp bp =
    bp {objects = (map (applyParm np vp) (objects bp))}
  applyParm np vp fo =
    fo {parameters' = ([(np,vp) | (i,j) <- (parameters fo),
                        np == i ] ++
                       [(i,j)   | (i,j) <- (parameters fo),
                        np /= i ])}

typeFlow :: FlowObject -> FlowObjectType
typeFlow (FlowObject _ t _ _) = t
typeFlow Start = EmptyFlow
typeFlow End = EmptyFlow
typeFlow Proceed = EmptyFlow

selectEval :: Id -> Id -> BusinessProcessModel
              -> BusinessProcessModel
```

```
              -> BusinessProcessModel
selectEval bpId advId spl product =
 let
  process1 = selectBusinessProcess bpId spl
              (BPM {processes = []});
  process2 = evaluateAdvice advId spl process1
 in case (processes process2) of
   [process] -> process2
   otherwise -> BPM {processes = []}

evaluateAdvice :: Id -> BusinessProcessModel
                     -> BusinessProcessModel
                     -> BusinessProcessModel
evaluateAdvice advId spl product =
 let advs = [a | a <- (processes spl),
   (pid a) == advId]
 in case advs of
      [adv] -> BPM (map
        (evaluateAdvice' adv) (processes product))
      otherwise -> product

evaluateAdvice' :: BusinessProcess -> BusinessProcess
                   -> BusinessProcess
evaluateAdvice' adv bp =
 let exist = ([obj | obj <- (objects bp),
   matches obj "" (pointcut adv)] /= [])
 in case ptype adv of
   BasicProcess      -> bp
   (Advice After  _) ->
     if exist then (evaluateAfterAdvice adv bp) else bp
   (Advice Before _) ->
     if exist then (evaluateBeforeAdvice adv bp) else bp
   (Advice Around _) ->
     if exist then (evaluateAroundAdvice adv bp) else bp

evaluateAfterAdvice :: BusinessProcess -> BusinessProcess
                       -> BusinessProcess
evaluateAfterAdvice adv bp =
 bp {
  objects = nub ((objects bp) ++ (objects adv)),
  transitions = ([(i,j,k) | (i,j,k) <- (transitions bp),
                   not (matches i k (pointcut adv))] ++
                 [(i,y,k) | (i,j,k) <- (transitions bp),
                   (x,y,z) <- startTransitions adv,
                   matches i k (pointcut adv)] ++
                 [(x,j,z) | (i,j,k) <- (transitions bp),
```

```
                    (x,y,z) <- endTransitions adv,
                    matches i k (pointcut adv)] ++
                  [(x,y,z) | (x,y,z) <- (transitions adv),
                    (x, y, z) `notElem`
                    ((startTransitions adv) ++
                      (endTransitions adv))])
 }

evaluateBeforeAdvice :: BusinessProcess -> BusinessProcess
-> BusinessProcess
evaluateBeforeAdvice adv bp =
 bp {
  objects = nub ((objects bp) ++ (objects adv)),
  transitions = ([(i,j,k) | (i,j,k) <- (transitions bp),
                 not (matches j k (pointcut adv))] ++
                 [(i,y,k) | (i,j,k) <- (transitions bp),
                 (x,y,z) <- startTransitions adv,
                 matches j k (pointcut adv)] ++
                 [(x,j,z) | (i,j,k) <- (transitions bp),
                 (x,y,z) <- endTransitions adv,
                 matches j k (pointcut adv)] ++
                 [(x,y,z) | (x,y,z) <- (transitions adv),
                 (x, y, z) `notElem` (
                   (startTransitions adv) ++
                   (endTransitions adv))])
 }

evaluateAroundAdvice :: BusinessProcess -> BusinessProcess
                        -> BusinessProcess
evaluateAroundAdvice adv bp =
 let objAnnot = [ obj | obj <- (objects bp),
                  matches obj "" (pointcut adv)];
     transitionsTemp = ([(i,j,k) |
     (i,j,k) <- (transitions bp),
     not (matches i k (pointcut adv))] ++
                         [(i,y,k) | (i,j,k) <-
                         (transitions bp),
                         (x,y,z) <- startTransitions adv,
                         matches j k (pointcut adv)] ++
                         [(x,j,z) | (i,j,k) <-
                         (transitions bp),
                         (x,y,z) <- endTransitions adv,
                           matches i k (pointcut adv)] ++
                         [(x,y,z) | (x,y,z) <-
                           (transitions adv),
                         (x, y, z) `notElem` (
```

```
                        (startTransitions adv) ++
                        (endTransitions adv))]);
     objProceed      = [ obj | obj <- (objects adv),
     obj == Proceed ]
 in case objProceed of
     -- com Proceed
     [objProceed] -> bp {
                         objects = (objects bp) ++
                         [obj|obj <- (objects adv),
                         not (obj `elem` [Start, End,
                         Proceed]) ],
                         transitions = ([(i,obj,k) |
                         (i,j,k) <- transitionsTemp, obj
                         <- objAnnot, j == Proceed ] ++
                         [(obj,j,k) |
                          (i,j,k) <- transitionsTemp, obj
                          <- objAnnot, i == Proceed ] ++
                         [(i,j,k) |
                          (i,j,k) <- transitionsTemp,
                          i /= Proceed, j /= Proceed ])
                        }
     -- sem Proceed
     otherwise ->
       bp {
        objects = [obj|obj <- (objects bp),
         obj `notElem` objAnnot] ++ [obj|obj <-
           (objects adv), not (obj `elem` [Start, End]) ],
        transitions = [(i,j,k) | (i,j,k) <- transitionsTemp]
       }

-- Função raiz, de primeiro nível: Avaliar o advice
-- tratando quantificação.
-- Chama a função de segundo nível.
evaluateAdviceQ :: Id -> BusinessProcessModel
                      -> BusinessProcessModel
                      -> BusinessProcessModel
evaluateAdviceQ advId spl product =
 let advs = [a | a <- (processes spl), (pid a) == advId]
 in case advs of
      [adv] -> BPM (map (evaluateAdviceQ' adv)
      (processes product))
      otherwise -> product

-- Função de segundo nível: Selecionar os objetos anotados
-- do bp-base e
-- chamar a função de terceiro nível.
```

```
evaluateAdviceQ' :: BusinessProcess -> BusinessProcess
                    -> BusinessProcess
evaluateAdviceQ' adv bpProduct =
 let objs = [obj | obj <- (objects bpProduct),
   (matches obj "" (pointcut adv))]
 in case objs of
      [] -> bpProduct
      otherwise -> evaluateAdviceQ'' adv objs bpProduct

-- Função de terceiro nível: Avalia o advice
-- de forma recursiva.
-- Resolve a variabilidade (evaluateAdvice ...) a
-- cada FlowObject
-- anotado do bp-base.
evaluateAdviceQ'' :: BusinessProcess
                     -> [FlowObject]
                     -> BusinessProcess
                     -> BusinessProcess
evaluateAdviceQ'' _ [] bpProduct = bpProduct
evaluateAdviceQ'' adv (o:os) bpProduct =
 case ptype adv of
   BasicProcess -> bpProduct
   (Advice After  _) ->
     evaluateAdviceQ'' advRen os
     (evaluateAfterAdviceQ adv bpProduct o)
   (Advice Before _) ->
     evaluateAdviceQ'' advRen os
     (evaluateBeforeAdviceQ adv bpProduct o)
   (Advice Around _) ->
     evaluateAdviceQ'' advRen os
     (evaluateAroundAdviceQ adv bpProduct o)
   where
   advRen =
    adv {objects = [rename oa | oa <- objects adv],
         transitions = [(rename i, rename j,k)|
         (i,j,k) <- transitions adv]}
   rename obj
     | (obj `notElem` [Start, End, Proceed]) =
         obj {id' = (idflow obj) ++ "'"}
     | otherwise = obj

evaluateAfterAdviceQ :: BusinessProcess
                        -> BusinessProcess
                        -> FlowObject
                        -> BusinessProcess
evaluateAfterAdviceQ adv bp objAnn =
```

```
 BusinessProcess {
  pid = pid bp,
  ptype = ptype bp,
  objects = (objects bp) ++ [obj | obj <- (objects adv),
                             obj `notElem` [Start, End]],
  transitions = ([(i,j,k) | (i,j,k) <- (transitions bp),
                  i /= objAnn] ++
                 [(i,y,z) | (i,j,k) <- (transitions bp),
                  (x,y,z) <- startTransitions adv,
                  i == objAnn] ++
                 [(x,j,z) | (i,j,k) <- (transitions bp),
                  (x,y,z) <- endTransitions adv,
                  i == objAnn] ++
                 [(x,y,z) | (x,y,z) <- (transitions adv),
                  (x, y, z) `notElem`
                  ((startTransitions adv) ++
                  (endTransitions adv))])
}

evaluateBeforeAdviceQ :: BusinessProcess -> BusinessProcess
                                         -> FlowObject
                                         -> BusinessProcess
evaluateBeforeAdviceQ adv bp objAnn =
 BusinessProcess {
  pid = pid bp,
  ptype = ptype bp,
  objects =
    (objects bp) ++ [obj | obj <- (objects adv),
                     obj `notElem` [Start, End]] ,
     transitions = ([(i,j,k) | (i,j,k) <- (transitions bp),
       j /= objAnn] ++
                    [(i,y,z) | (i,j,k) <- (transitions bp),
                     (x,y,z) <- startTransitions adv,
                     j == objAnn] ++
                    [(x,j,z) | (i,j,k) <- (transitions bp),
                     (x,y,z) <- endTransitions adv,
                     j == objAnn] ++
                    [(x,y,z) | (x,y,z) <-
                    (transitions adv),
                     (x, y, z) `notElem` (
                      (startTransitions adv) ++
                      (endTransitions adv))])
}

evaluateAroundAdviceQ :: BusinessProcess -> BusinessProcess
                         -> FlowObject
```

```
                          -> BusinessProcess
evaluateAroundAdviceQ adv bp objAnn =
 let objsAnnot = [ o | o <- (objects bp), o == objAnn];
     transitionsTemp = ([(i,j,k) | (i,j,k) <-
     (transitions bp),
                         i /= objAnn, j /= objAnn] ++
                         [(i,y,z) | (i,j,k) <-
                         (transitions bp),
                          (x,y,z) <- startTransitions adv,
                          j == objAnn] ++
                         [(x,j,z) | (i,j,k) <-
                         (transitions bp),
                          (x,y,z) <- endTransitions adv,
                          i == objAnn] ++
                         [(x,y,z) | (x,y,z) <-
                         (transitions adv),
                          (x, y, z) `notElem` (
                           (startTransitions adv) ++
                           (endTransitions adv))]);
     objProceed      = [ obj | obj <- (objects adv),
      obj == Proceed ]
 in case objProceed of
     -- com Proceed
     [objProceed] ->
       bp {
           objects = (objects bp) ++
             [obj | obj <- (objects adv),
             obj `notElem`
             [Start, End, Proceed]],
           transitions = ([(i,obj,k) | (i,j,k) <-
                           transitionsTemp,
                          obj <- objsAnnot, j ==
                            Proceed ] ++
                          [(obj,j,k) | (i,j,k)
                            <- transitionsTemp,
                          obj <- objsAnnot,
                          i == Proceed ] ++
                          [(i,j,k) | (i,j,k)
                          <- transitionsTemp,
                          i /= Proceed, j /= Proceed ])
           }
     -- sem Proceed
     otherwise ->
       bp {
           objects = [obj | obj <- (objects bp),
                      obj `notElem` objsAnnot] ++
```

```
                    [obj | obj <- (objects adv),
                    obj `notElem` [Start, End]],
            transitions = [(i,j,k) | (i,j,k) <-
             transitionsTemp]
            }

matches :: FlowObject -> Condition -> Pointcut -> Bool
matches fo _ (Pointcut pc) = pc `elem` (annotations fo)
matches fo c (PCut pc cs) = (pc `elem` (annotations fo)) &&
                            (c `elem` cs)
matches fo _ (Empty) = False

startTransitions :: BusinessProcess -> [Transition]
startTransitions bp = [(i, j, k) | (i, j, k) <-
                       (transitions bp), i == Start]

endTransitions :: BusinessProcess -> [Transition]
endTransitions bp = [(i, j, k) | (i, j, k) <-
                     (transitions bp), j == End]

(|=>) :: FlowObject -> FlowObject -> Transition
(|=>) a b = (a, b, "")

print' :: Transition -> String
print' (a, b, c) = (show a) ++ "->" ++ (show b)

instance Show FlowObject where
 show object = "(" ++  (idflow object) ++
                (showParameters (parameters object)) ++ ")"

showParameters :: [Parameter] -> String
showParameters [] = ""
showParameters ps = " - Parameters: " ++
 (init (concat (map (++",")
 (map showParameter ps))))

showParameter :: Parameter -> String
showParameter p = (fst p) ++ " - " ++ (value (snd p))

value :: Value -> String
value Unbound = "unbound"
value (Value v) = v
```

Observação: A função show, implementada para mostrar em tela instâncias do tipo algébrico *FlowObject* (Apêndice A), omite seus detalhes (seus *fields*) para não tornar a apresentação dos processos de negócios demasiadamente extensas, já que cada processo de negócio é composto por uma lista de objetos - *FlowObjects* - e uma lista de transições

- e cada transição é composta por dois objetos. Mas, nos casos em que trata variabilidade de processos de negócio com abordagem paramétrica (Casos 4.2.2 e B.2), foi necessário adaptar essa função para mostrar a lista de parâmetros (*field* de *FlowObject*) e seus valores para expor o resultado da execução das transformações que tratam esse tipo de variabilidade. No entanto, a atividade "Realizar acertos financeiros" está presente também em algumas transições (*transitions*), mas, os parâmetros estão sem valor vinculado (*Unbound*). Faz parte de trabalhos futuros definir uma função *show* específica para mostrar cada transição de modo que sejam omitidos os detalhes de seu par de objetos quando um deles, ou ambos, tiver parâmetros.

Apêndice B

Avaliação - Demais Casos

B.1 Caso VI - Licença - Licença Incentivada sem Remuneração e Licença para Capacitação

B.1.1 Processos Originais

Os processos originais estão ilustrados nas Figuras B.1 e B.2 que descrevem os passos para conceder licença incentivada sem remuneração ou licença para capacitação a um servidor/empregado.

B.1.2 *BusinessProcessModel* - Processos Comuns e Variantes

Os tipos algébricos do processo-comum (Figura B.3) e dos processos-advice (B.4) estão em Apêndice, na Seção C.6.

B.1.3 *Configuration Knowledge*

Na Figura B.5, apresentamos o CK deste caso.

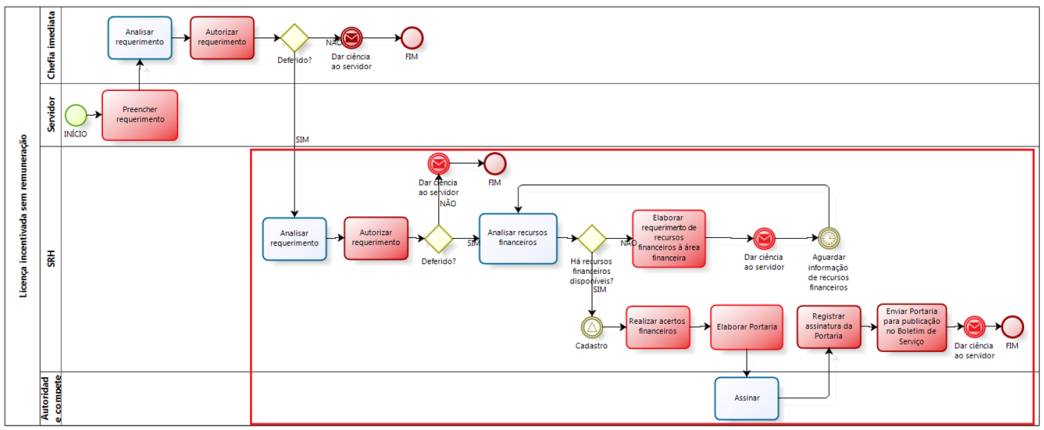


Figura B.1: Processo Original - Licença Incentivada sem Remuneração

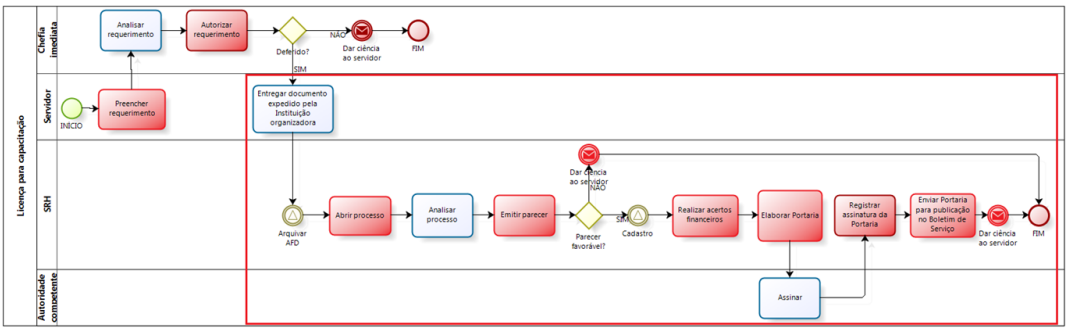


Figura B.2: Processo Original - Licença para Capacitação

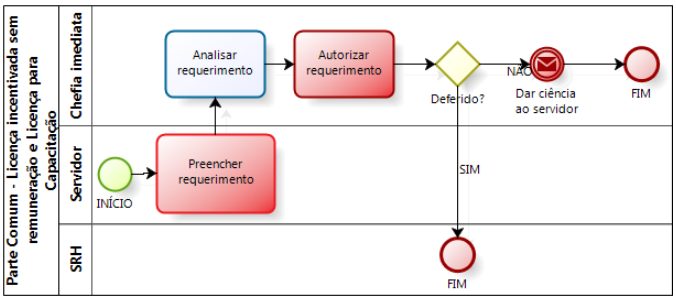


Figura B.3: Processo Comum aos Processos Licença Incentivada sem Remuneração e Licença para Capacitação

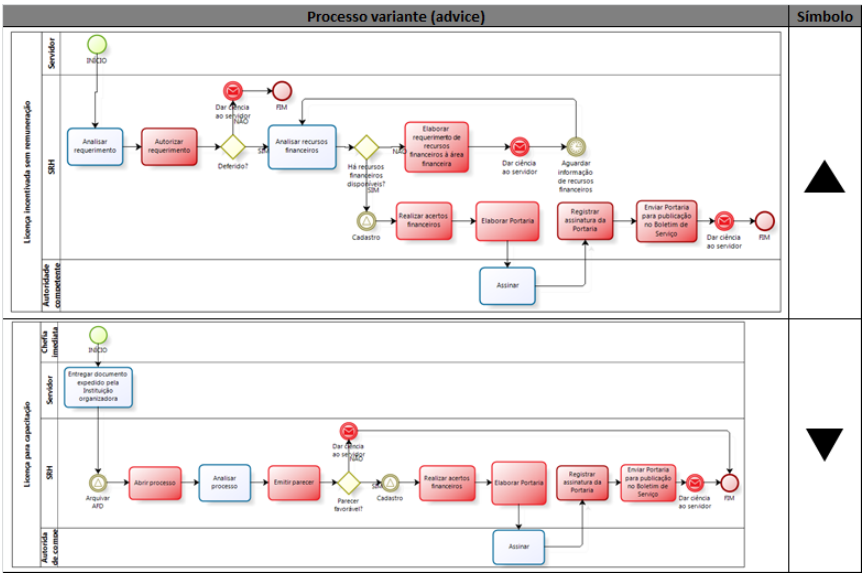


Figura B.4: Processos Variantes (advices) do Processo Comum de Licença Incentivada sem Remuneração e Licença para Capacitação

CK	
Feature Expressions	**Transformations**
Capacitacao	selectEval "bpLicIncSemRemLicParaCapacitacaoComum" "advLicParaCapacitacao" spl product
Incentivada sem Remuneracao	selectEval "bpLicIncSemRemLicParaCapacitacaoComum" "advLicIncSemRemuneracao" spl product

Figura B.5: CK - Licença - Licença Incentivada sem Remuneração e Licença para Capacitação

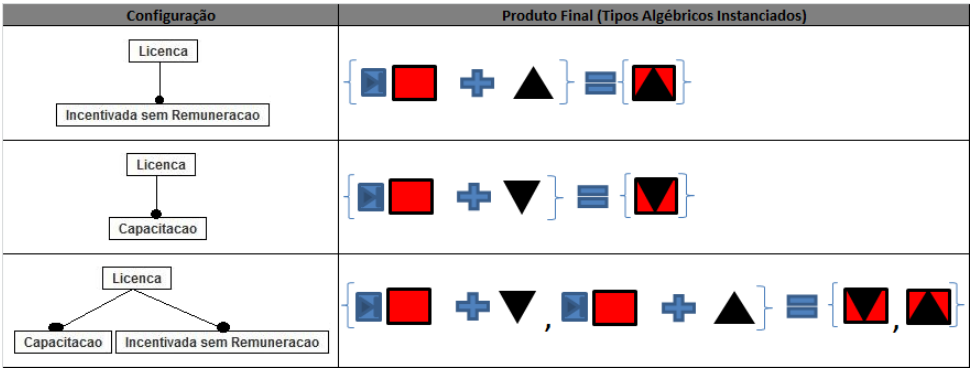


Figura B.6: Caso 6 - Licença - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

B.1.4 Configuração do Produto

Na Figura B.6, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.6.

B.2 Caso VII - Adicional - Periculosidade e Insalubridade

B.2.1 Processos Originais

Os processos originais estão ilustrados nas Figuras B.7 e B.8, que descrevem os passos para indenizar um servidor ou empregado com um adicional no salário devido à trabalhar em alguma atividade que tenha algum grau de periculosidade e/ou insalubridade.

B.2.2 *BusinessProcessModel* - Processos Comuns e Variantes

Como ambos processos são idênticos, variando apenas o nome e o valor interno ao executar a atividade "Realizar acertos financeiros", não temos ilustração para esta seção. Apenas mostramos o tipo algébrico do processo-base (processo-comum eleito) deste caso em Apêndice, na Seção C.7.

B.2.3 *Configuration Knowledge*

Na Figura B.9, apresentamos o CK deste caso.

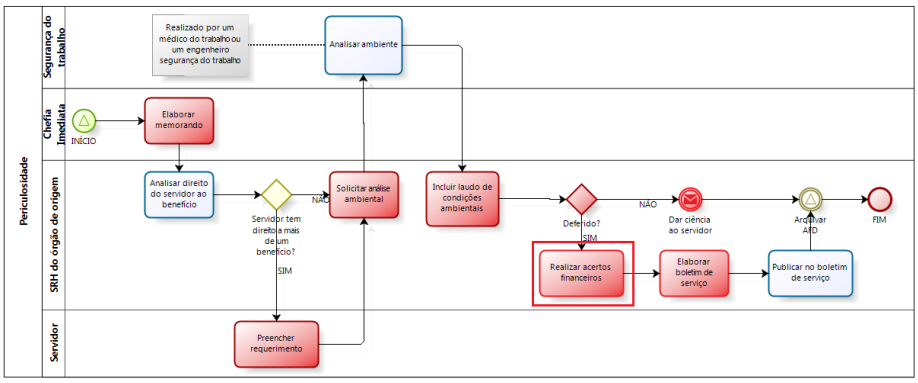


Figura B.7: Processo Original - Periculosidade

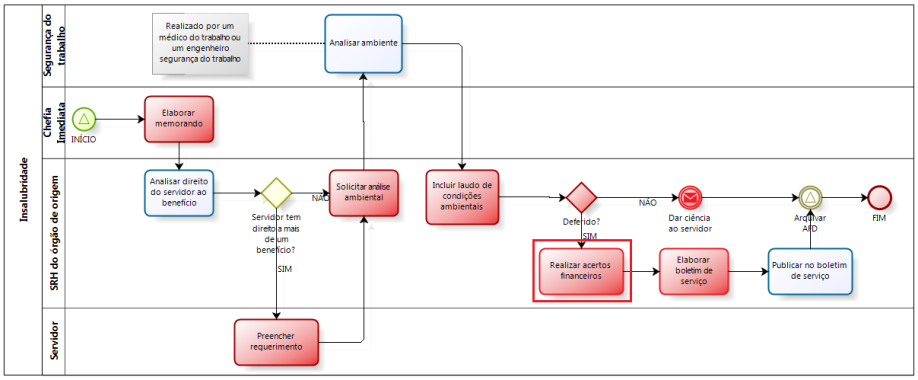


Figura B.8: Processo Original - Insalubridade

CK	
Feature Expressions	**Transformations**
pA = 15%	selectBind "bpPericulosidade" spl "pA" (Value "15%")
pA = 30%	selectBind "bpPericulosidade" spl "pA" (Value "30%")

Figura B.9: CK - Adicional - Periculosidade e Insalubridade

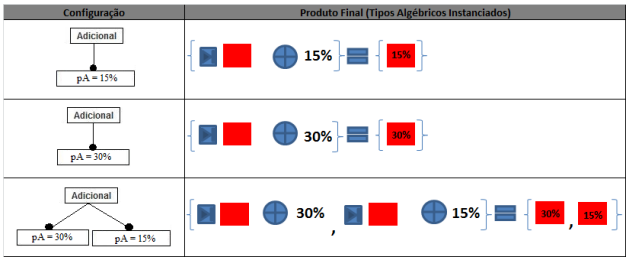


Figura B.10: Caso VII - Adicional - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

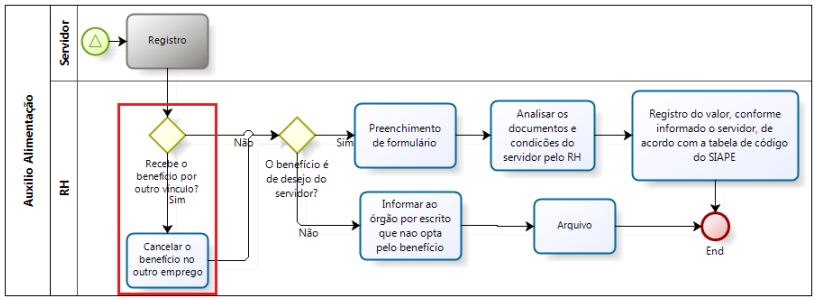


Figura B.11: Processo Original - Auxílio Alimentação

B.2.4 Configuração do Produto

Na Figura B.10, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.7.

B.3 Caso VIII - Auxílio - Auxílio Alimentação e Auxílio Transporte

B.3.1 Processos Originais

Os processos originais estão ilustrados nas Figuras B.11 e B.12, que descrevem os passos para conceder os auxílios alimentação e/ou transporte a um servidor ou empregado.

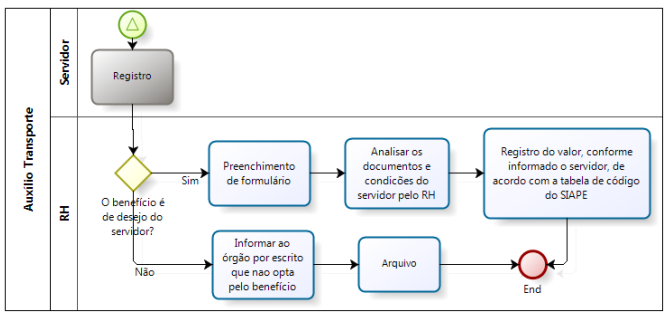


Figura B.12: Processo Original - Auxílio Transporte

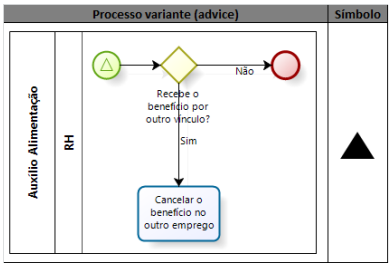


Figura B.13: Processo Variante - Auxílio Alimentação

CK	
Feature Expressions	**Transformations**
Alimentacao	selectBusinessProcess "bpAuxilioComum" spl product evaluateAdvice "advAuxilioAlimentacao" spl product
Transporte	selectBusinessProcess "bpAuxilioComum" spl product

Figura B.14: CK - Auxílio - Auxílio Alimentação e Auxílio Transporte

B.3.2 *BusinessProcessModel* - Processos Comuns e Variantes

Os tipos algébricos do processo-base (Figura B.12) e do processo-advice (Figura B.13) estão em Apêndice, na Seção C.8.

B.3.3 *Configuration Knowledge*

Na Figura B.14, apresentamos o CK deste caso.

B.3.4 Configuração do Produto

Na Figura B.15, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.8.

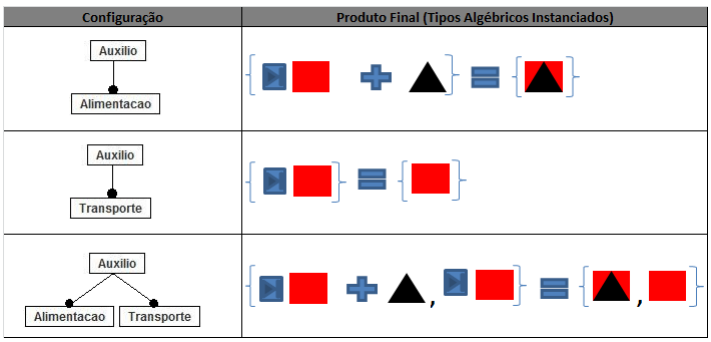


Figura B.15: Caso VIII - Auxílio - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

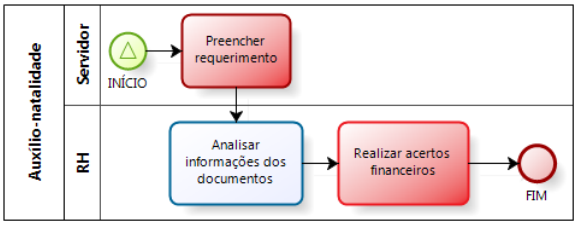


Figura B.16: Processo Original - Auxílio Natalidade

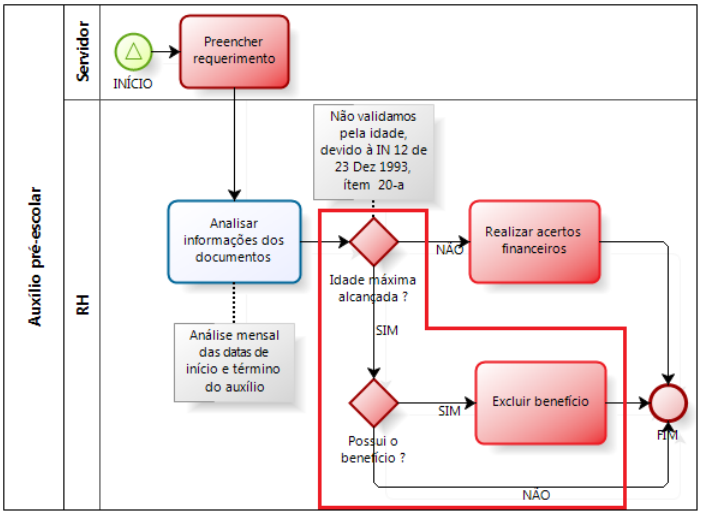


Figura B.17: Processo Original - Auxílio Pré-Escolar

B.4 Caso IX - Auxílio Filiação - Auxílio Natalidade e Auxílio Pré-Escolar

B.4.1 Processos Originais

Os processos originais estão ilustrados nas Figuras B.16 e B.17, que descrevem os passos para conceder auxílio natalidade e/ou auxílio pré-escolar a um servidor ou empregado.

B.4.2 *BusinessProcessModel* - Processos Comuns e Variantes

Os tipos algébricos do processo-comum (Figura B.16) e do processo-advice (Figura B.18) estão em Apêndice, na Seção C.9.

B.4.3 *Configuration Knowledge*

Na Figura B.19, apresentamos o CK deste caso.

B.4.4 Configuração do Produto

Na Figura B.20, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.9.

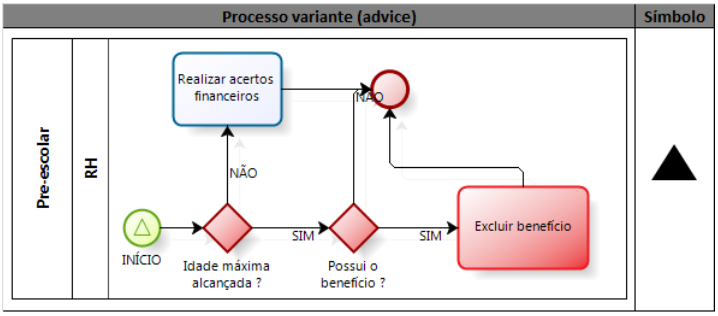


Figura B.18: Processo Variante - Auxílio Pré-Escolar

CK	
Feature Expressions	**Transformations**
Auxilio Pre Escolar	selectBusinessProcess "bpAuxilioNatalidade" spl product evaluateAdvice "advAuxilioPreescolar" spl product
Auxilio Natalidade	selectBusinessProcess "bpAuxilioNatalidade" spl product

Figura B.19: CK - Auxílio Filiação - Auxílio Natalidade e Auxílio Pré-Escolar

Configuração	Produto Final (Tipos Algébricos Instanciados)
Auxilio Filiacao Auxilio Natalidade	
Auxilio Filiacao Auxilio Pre-Escolar	
Auxilio Filiacao Auxilio Natalidade Auxilio Pre-Escolar	

Figura B.20: Caso IX - Auxílio Filiação - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

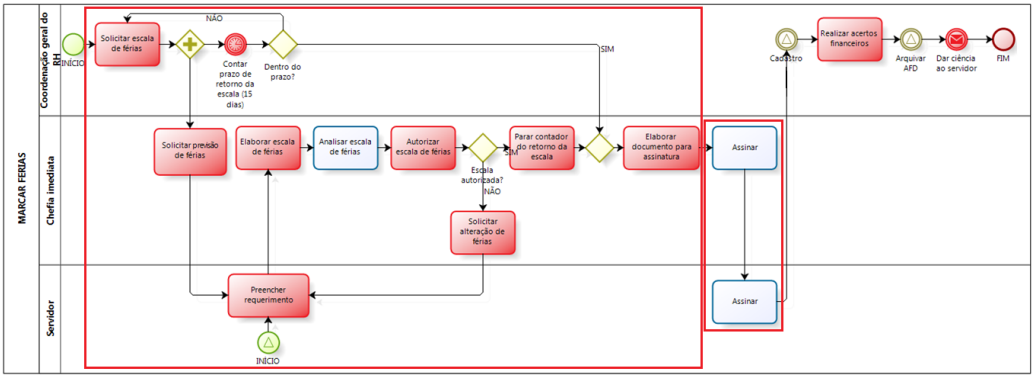


Figura B.21: Processo Original - Marcar Férias

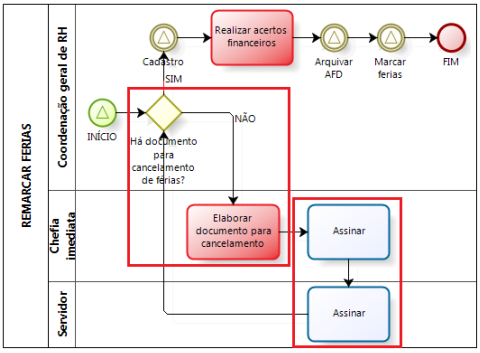


Figura B.22: Processo Original - Remarcar Féris

B.5 Caso X - Férias - Marcar e Férias e Remarcar Férias - Produto Final

B.5.1 Processos Originais

Os processos originais estão ilustrados nas Figuras B.21 e B.22, que descrevem os passos para um servidor ou empregado para marcar ou remarcar férias.

B.5.2 *BusinessProcessModel* - Processos Comuns e Variantes

Os tipos algébricos do processo-comum (Figura B.23) e do processo-advice (Figura B.24) estão em Apêndice, na Seção C.10.

B.5.3 *Configuration Knowledge*

Na Figura B.25, apresentamos o CK deste caso.

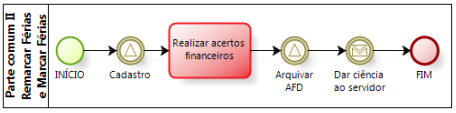


Figura B.23: Processo Comum aos Processos Marcar Férias e Remarcar Férias

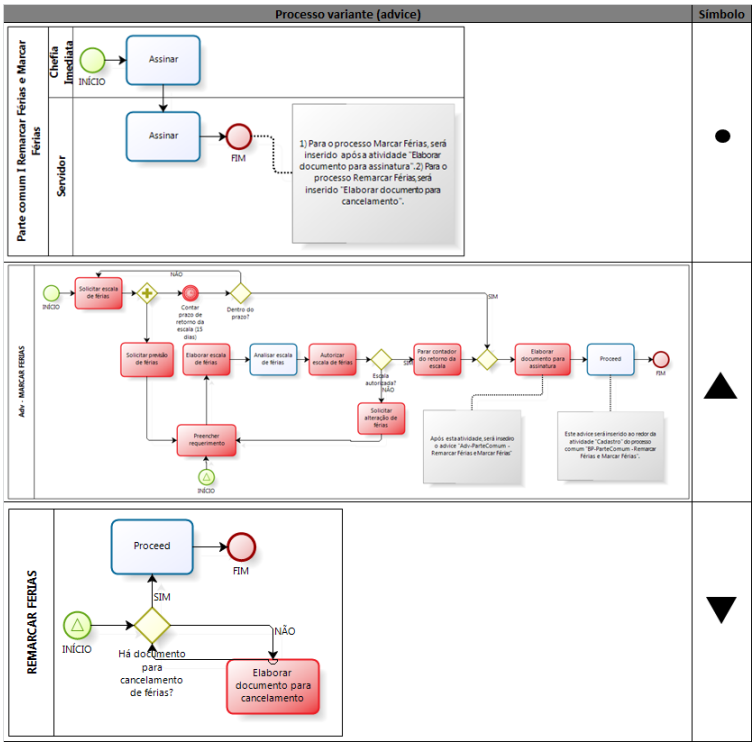


Figura B.24: Processo Variantes do Processo Comum de Marcar Férias e Remarcar Féris

CK	
Feature Expressions	**Transformations**
Marcar Ferias	selectEval "bpFeriasComum" "advMarcarFerias" spl product, evaluateAdvice "advFeriasComum" spl product
Remarcar Ferias	selectEval "bpFeriasComum" "advRemarcarFerias" spl product, evaluateAdvice "advFeriasComum" spl product

Figura B.25: CK - Férias - Marcar Férias e Remarcar Férias

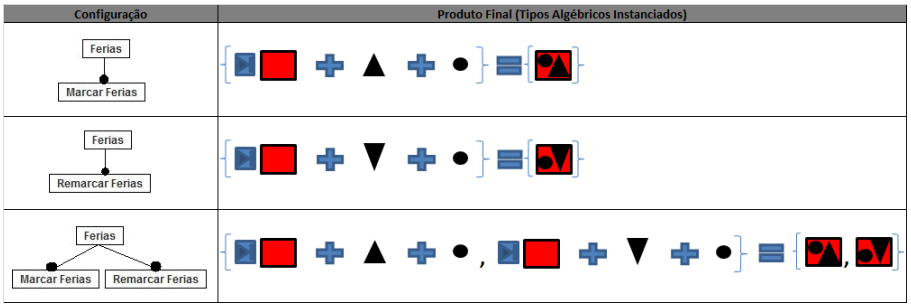


Figura B.26: Caso X - Férias - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

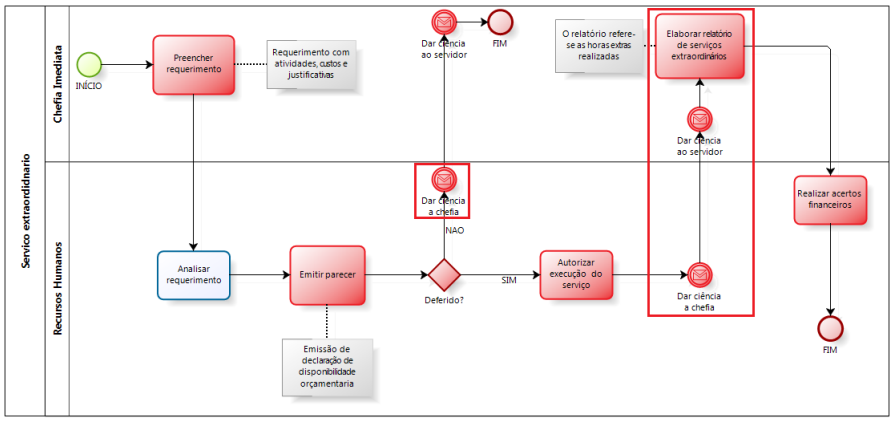


Figura B.27: Processo Original - Serviço Extraordinário

B.5.4 Configuração do Produto

Na Figura B.26, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.10.

B.6 Caso XI - Serviço - Serviço Extraordinário e Tempo de Serviço Quinquênio

B.6.1 Processos Originais

Os processos originais estão ilustrados nas Figuras B.27 e B.28, que descrevem os passos para aprovar/reprovar à convocação a um servidor/empregado para realizar serviço extraordinário ou conceder-lhe benefício oriundo de tempo de serviço (quinquênio).

B.6.2 *BusinessProcessModel* - Processos Comuns e Variantes

Os tipos algébricos do processo-comum (Figura B.29) e do processo-advice (Figura B.30) estão em Apêndice, na Seção C.11.

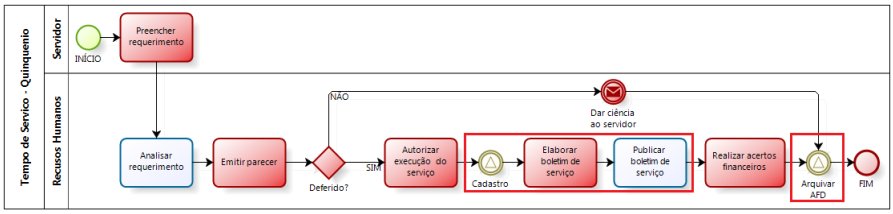


Figura B.28: Processo Original - Tempo de Serviço Quinquênio

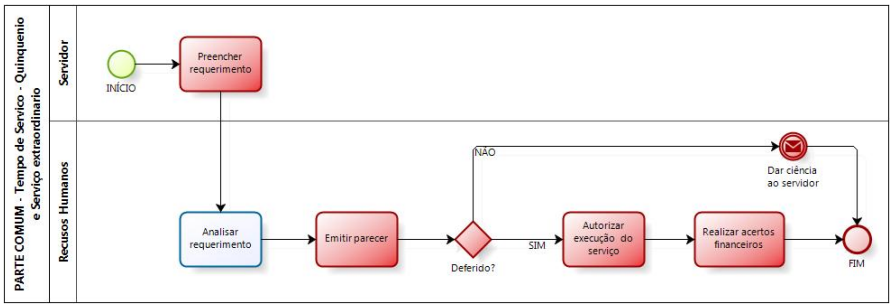


Figura B.29: Processo Comum dos Processos Serviço Extraordinário e Tempo de Serviço Quinquênio

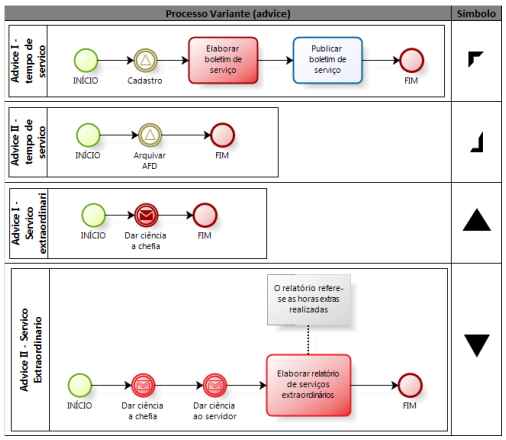


Figura B.30: Processos Variantes do Processo Comum dos Processos Serviço Extraordinário e Tempo de Serviço Quinquênio

CK	
Feature Expressions	**Transformations**
Servico Extraordinario	selectEval "bpServicoComum" "advServicoExtraordinario1" spl product evaluateAdvice "advServicoExtraordinario2" spl product
Tempo de Servico	selectEval "bpServicoComum" "advTempoDeServico1" spl product selectEval "bpServicoComum" "advTempoDeServico2" (BPM { processes = (processes product) ++ [advTempoDeServico2] }) product

Figura B.31: CK - Serviço - Serviço Extraordinário e Tempo de Serviço Quinquênio

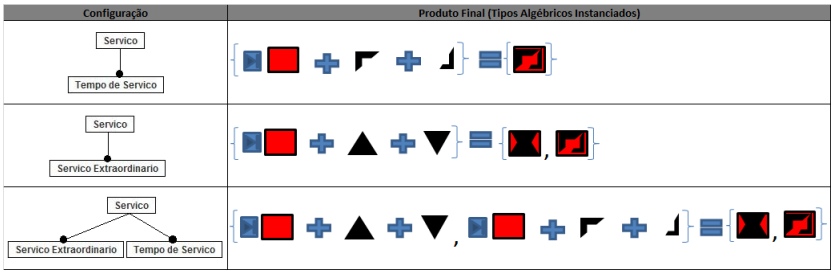


Figura B.32: Caso XI - Serviço - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

B.6.3 *Configuration Knowledge*

Na Figura B.31, apresentamos o CK deste caso.

B.6.4 Configuração do Produto

Na Figura B.32, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.11.

B.7 Caso XII - Indenização - Indenização de Transporte e Transporte de Mobiliário e Bagagens

B.7.1 Processos Originais

Os processos originais estão ilustrados nas Figuras B.33 e B.34, que descrevem os passos para indenizar um servidor/empregado quanto aos gastos em uma viagem à trabalho ou quanto aos gastos de uma mudança de endereço originada de uma transferência por interesse da Administração Pública.

B.7.2 *BusinessProcessModel* - Processos Comuns e Variantes

Os tipos algébricos do processo-base (Figura B.35) e do processo-advice (Figura B.36) estão em Apêndice, na Seção C.12.

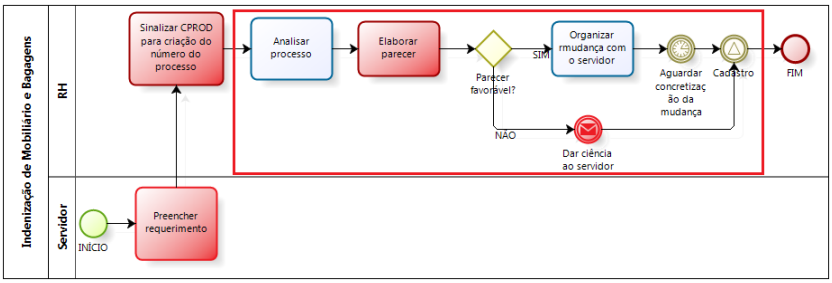


Figura B.33: Processo Original - Indenização de Transporte

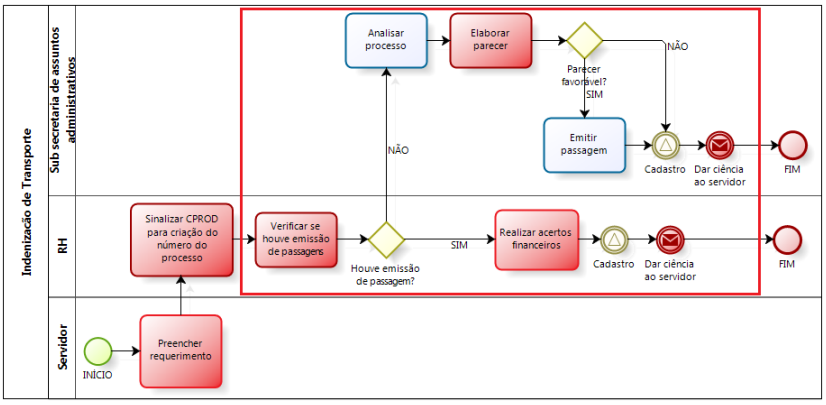


Figura B.34: Processo Original - Indenização de Transporte de Mobiliário e Bagagens

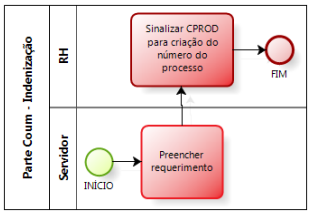


Figura B.35: Processo Comum aos Processos Indenização de Transporte e Indenização de Transporte de Mobiliário e Bagagens

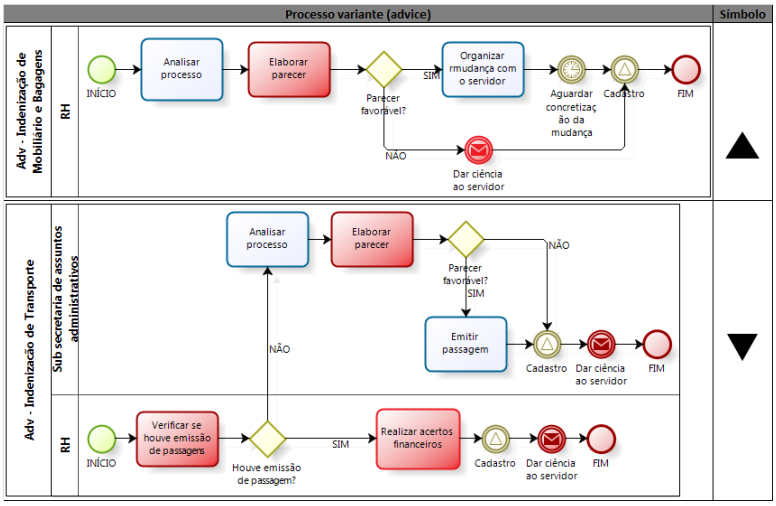


Figura B.36: Processos Variantes do Processo Comum aos Processos Indenização de Transporte e Indenização de Transporte de Mobiliário e Bagagens

CK	
Feature Expressions	**Transformations**
Mobiliario e Bagagens	selectEval "bpIndenizacaoComum" "advMobBag" spl product
Transporte	selectEval "bpIndenizacaoComum" "advTransporte" spl product

Figura B.37: CK - Indenização - Indenização de Transporte e Transporte de Mobiliário e Bagagens

B.7.3 *Configuration Knowledge*

Na Figura B.37, apresentamos o CK deste caso.

B.7.4 Configuração do Produto

Na Figura B.38, mostramos as configurações possíveis e os resultados deste caso. Os tipos algébricos instanciados de cada produto final deste caso estão em Apêndice, na Seção D.12.

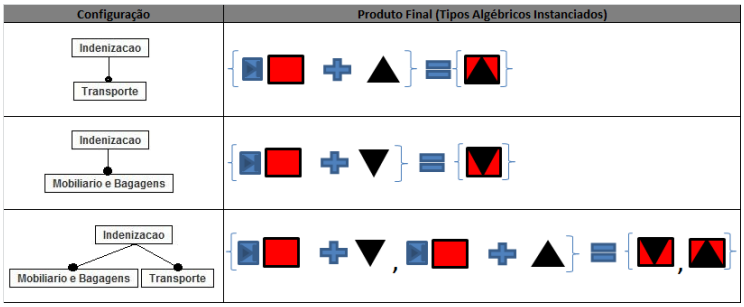


Figura B.38: Caso XI - Indenização - Configuração do Produto - Ilustração de cada PC e seu respectivo resultado

Apêndice C

BusinessProcessModel - Tipos Algébricos

C.1 Caso I - Transferência - Cessão e Requisição - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Receber requerimento", type' = Acti
vity, annotations' = [], parameters' = []}

cc :: FlowObject
cc = FlowObject { id' = "Encaminhar para SRH", type' = Activ
ity, annotations' = [], parameters' = []}

dd :: FlowObject
dd = FlowObject { id' = "Analisar", type' = Activity, annota
tions' = [], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Deferido?", type' = Gateway, annota
tions' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Dar ciencia ao orgao  requerente",
type' = Activity, annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Analisar liberacao", type' = Activi
ty, annotations' = [], parameters' = []}
```

```
hh :: FlowObject
hh = FlowObject { id' = "Concorda?", type' = Gateway, annota
tions' = [], parameters' = []}

ii :: FlowObject
ii = FlowObject { id' = "Concorda?", type' = Gateway, annota
tions' = [], parameters' = []}

jj :: FlowObject
jj = FlowObject { id' = "Incluir anuencia", type' = Activity
, annotations' = [], parameters' = []}

mm :: FlowObject
mm = FlowObject { id' = "Elaborar Portaria", type' = Activit
y, annotations' = [], parameters' = []}

nn :: FlowObject
nn = FlowObject { id' = "Publicacao", type' = Activity, anno
tations' = [], parameters' = []}

oo :: FlowObject
oo = FlowObject { id' = "Cadastro", type' = Activity, annota
tions' = [], parameters' = []}

pp :: FlowObject
pp = FlowObject { id' = "Elaborar oficio de apresentacao do
servidor", type' = Activity, annotations' = [], parameters'
= []}

qq :: FlowObject
qq = FlowObject { id' = "Receber oficio de apresentacao", ty
pe' = Activity, annotations' = [], parameters' = []}


bpParteComumCessaoRequisicao = BusinessProcess {
 pid = "bpParteComumCessaoRequisicao",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, ff, gg, hh, ii, jj, m
 m, nn, oo, pp, qq, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, cc |=> d
 d, dd |=> ee,
 gateyV ee ff "NAO", ff |=> End, gateyV ee gg "SIM", gg |=>
 hh, gateyV hh ff "NAO", gateyV hh ii "SIM", gateyV ii ff "N
 AO", gateyV ii jj "SIM", jj |=> mm, mm |=> nn, nn |=> oo, o
 o |=> pp, pp |=> qq, qq |=> End]
```

```
}

splModel = BPM {processes = [bpParteComumCessaoRequisicao]}
```

C.2 Caso II - Radiação - Raio-X ou Substância Radioativa e Irradiação Ionizante - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Ha necessidade de realizar exame me
dico?", type' = Gateway, annotations' = [], parameters' = []}

cc :: FlowObject
cc = FlowObject { id' = "Contar prazo (6 meses)", type' = Ac
tivity, annotations' = [], parameters' = []}

dd :: FlowObject
dd = FlowObject { id' = "Elaborar requisicao de exame medico
", type' = Activity, annotations' = [], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Receber requisicao de exame medico"
, type' = Activity, annotations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Realizar exame medico", type' = Act
ivity, annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Entregar exame medico", type' = Act
ivity, annotations' = [], parameters' = []}

hh :: FlowObject
hh = FlowObject { id' = "Receber exame medico", type' = Acti
vity, annotations' = [], parameters' = []}

ii :: FlowObject
ii = FlowObject { id' = "Incluir laudo medico", type' = Acti
vity, annotations' = [], parameters' = []}

jj :: FlowObject
```

```
jj = FlowObject { id' = "Servidor esta apto?", type' = Gatew
ay, annotations' = [], parameters' = []}

ll :: FlowObject
ll = FlowObject { id' = "Realizar acertos finaceiros", type'
 = Activity, annotations' = [], parameters' = [("pR",Unbound
)]}

mm :: FlowObject
mm = FlowObject { id' = "Elaborar  Portaria", type' = Activi
ty, annotations' = [], parameters' = []}

nn :: FlowObject
nn = FlowObject { id' = "Publicacao", type' = Activity, anno
tations' = [], parameters' = []}

oo :: FlowObject
oo = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

pp :: FlowObject
pp = FlowObject { id' = "Reconducao", type' = Activity, anno
tations' = [], parameters' = []}

bpParteComumRaioXRadiacaoIonizante = BusinessProcess {
 pid = "bpParteComumRaioXRadiacaoIonizante",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, ff, gg, hh, ii, jj, l
 l, mm, nn, oo, pp, End],
 transitions = [Start |=> aa, aa |=> bb, gateyV bb cc "NAO",
  gateyV bb dd "SIM", dd |=> ee, ee |=> ff, ff |=> gg, gg |=
  > hh, hh |=> ii, ii |=> jj, gateyV jj ll "SIM",
                ll |=> mm, mm |=> nn, nn |=> oo, oo |=> End,
                 gateyV jj pp "NAO", pp |=> End]
}

splModel = BPM {processes = [bpParteComumRaioXRadiacaoIoniza
nte]}
```

C.3 Caso III - Desligamento - Demissão e Aposentadoria - Produto Final - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Receber processo", type' = Activity
```

```
, annotations' = ["pcDemissao"], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Elaborar Portaria", type' = Activit
y, annotations' = [], parameters' = []}

cc :: FlowObject
cc = FlowObject { id' = "Publicacao", type' = Activity, anno
tations' = [], parameters' = []}

dd :: FlowObject
dd = FlowObject { id' = "Cadastro", type' = Activity, annota
tions' = [], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Realizar acertos financeiros", type
' = Activity, annotations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Sinalizar SISAC", type' = Activity,
 annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

bpDemissao = BusinessProcess {
 pid = "bpDemissao",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, ff, gg, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, cc |=> d
 d, dd |=> ee, ee |=> ff, ff |=> gg, gg |=> End]
}

hh :: FlowObject
hh = FlowObject { id' = "Analisar requisitos legais do proce
sso", type' = Activity, annotations' = [], parameters' = []}

advAposentadoria = BusinessProcess {
 pid = "advAposentadoria",
 ptype = Advice {advType = After, pc = Pointcut "pcDemissao"
 },
 objects = [Start, hh, End],
 transitions = [Start |=> hh, hh |=> End]
}
```

```
spl = BPM {processes = [bpDemissao, advAposentadoria]}
```

C.4 Caso IV - Ouvidoria - Reclamação e Elogio - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Enviar Elogio", type' = Activity, a
nnotations' = ["pcReclamacao1"], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Incluir Elogio", type' = Activity,
annotations' = ["pcReclamacao2"], parameters' = []}

cc :: FlowObject
cc = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

dd :: FlowObject
dd = FlowObject { id' = "O setor do elogio eh externo?", typ
e' = Gateway, annotations' = ["pcAjudaCusto2", "pcAuxilioMor
adia"], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Encaminhar elogio para setor ou org
ao externo", type' = Activity, annotations' = [], parameters
' = []}

ff :: FlowObject
ff = FlowObject { id' = "Elaborar correspondencia com elogio
", type' = Activity, annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Enviar elogio ao servidor", type' =
 Activity, annotations' = [], parameters' = []}

hh :: FlowObject
hh = FlowObject { id' = "Receber elogio", type' = Activity,
annotations' = ["pcReclamacao3"], parameters' = []}

bpElogioComum = BusinessProcess {
 pid = "bpElogioComum",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, ff, gg, hh, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, cc |=> d
 d, gateyV dd ee "SIM",
```

```
gateyV dd ff "NAO", ff |=> gg, gg |=> hh, hh |=> End]
}

ii :: FlowObject
ii = FlowObject { id' = "Enviar Reclamacao", type' = Activit
y, annotations' = [], parameters' = []}

advReclamacao1 = BusinessProcess {
 pid = "advReclamacao1",
 ptype = Advice {advType = Around, pc = Pointcut "pcReclamac
 ao1"},
 objects = [Start, ii, End],
 transitions = [Start |=> ii, ii |=> End]
}

jj :: FlowObject
jj = FlowObject { id' = "Incluir Reclamacao", type' = Activi
ty, annotations' = [], parameters' = []}

advReclamacao2 = BusinessProcess {
 pid = "advReclamacao2",
 ptype = Advice {advType = Around, pc = Pointcut "pcReclamac
 ao2"},
 objects = [Start, jj, End],
 transitions = [Start |=> jj, jj |=> End]}

kk :: FlowObject
kk = FlowObject { id' = "Responder critica", type' = Activit
y, annotations' = [], parameters' = []}

mm :: FlowObject
mm = FlowObject { id' = "Analisar resposta", type' = Activit
y, annotations' = [], parameters' = []}

nn :: FlowObject
nn = FlowObject { id' = "Incluir satisfacao da resposta", ty
pe' = Activity, annotations' = [], parameters' = []}

oo :: FlowObject
oo = FlowObject { id' = "Sinalizar SisOuvidor", type' = Acti
vity, annotations' = [], parameters' = []}

advReclamacao3 = BusinessProcess { pid = "advReclamacao3", p
type = Advice {advType = After, pc = Pointcut "pcReclamacao3
"}, objects = [Start, kk, mm, nn, oo, End], transitions = [S
tart |=> kk, kk |=> mm, mm |=> nn, nn |=> oo, oo |=> End]}
```

```
spl = BPM {processes = [bpElogioComum, advReclamacao1, advRe
clamacao2, advReclamacao3]}
```

C.5 Caso V - Complemento Salarial - Ajuda de Custo e Auxílio Moradia - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = ["pcAjudaCusto1"], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Analisar requerimento e documentaca
o", type' = Activity, annotations' = [], parameters' = []}

cc :: FlowObject
cc = FlowObject { id' = "Autorizar requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

dd :: FlowObject
dd = FlowObject { id' = "Deferido?", type' = Gateway, annota
tions' = ["pcAjudaCusto2", "pcAuxilioMoradia"], parameters'
= []}

ee :: FlowObject
ee = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Sinalizar SIAFI", type' = Activity,
 annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Emissao de ordem bancaria", type' =
 Activity, annotations' = [], parameters' = []}

hh :: FlowObject
hh = FlowObject { id' = "Cadastro", type' = Activity, annota
tions' = [], parameters' = []}

ii :: FlowObject
ii = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}
```

```
bpAjudaCustoComum = BusinessProcess {
 pid = "bpAjudaCustoComum",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, ff, gg, hh, ii, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, cc |=> d
 d, gateyV dd ee "NAO", ee |=> End, gateyV dd ff "SIM", ff |
 => gg, gg |=> hh, hh |=> ii, ii |=> End]
}

jj :: FlowObject
jj = FlowObject { id' = "Entregar documentos comprobatorios"
, type' = Activity, annotations' = [], parameters' = []}

kk :: FlowObject
kk = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

advAjudaCusto1 = BusinessProcess {
 pid = "advAjudaCusto1",
 ptype = Advice {advType = After, pc = Pointcut "pcAjudaCust
 o1"},
 objects = [Start, jj, kk, End],
 transitions = [Start |=> jj, jj |=> kk, kk |=> End]
}

ll :: FlowObject
ll = FlowObject { id' = "Cadastro", type' = Activity, annota
tions' = [], parameters' = []}

advAjudaCusto2 = BusinessProcess {
 pid = "advAjudaCusto2",
 ptype = Advice {advType = After, pc = PCut "pcAjudaCusto2"
 ["NAO"]},
 objects = [Start, ll, End],
 transitions = [Start |=> ll, ll |=> End]
}

mm :: FlowObject
mm = FlowObject { id' = "Enviar, mensalmente, comprovante de
 despesas de moradia", type' = Activity, annotations' = [],
 parameters' = []}

nn :: FlowObject
nn = FlowObject { id' = "Analisar comprovante", type' = Acti
vity, annotations' = [], parameters' = []}
```

```
oo :: FlowObject
oo = FlowObject { id' = "Elaborar parecer", type' = Activity
, annotations' = [], parameters' = []}

pp :: FlowObject
pp = FlowObject { id' = "Parecer favoravel?", type' = Gatewa
y, annotations' = [], parameters' = []}

qq :: FlowObject
qq = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

rr :: FlowObject
rr = FlowObject { id' = "Realizar acertos financeiros", type
' = Activity, annotations' = [], parameters' = []}

ss :: FlowObject
ss = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

advAuxilioMoradia = BusinessProcess {
 pid = "advAuxilioMoradia",
 ptype = Advice {advType = After, pc = PCut "pcAuxilioMoradi
 a" ["SIM"]},
 objects = [Start, mm, nn, oo, pp, qq, rr, ss, End],
 transitions = [Start |=> mm, mm |=> nn, nn |=> oo, oo |=> p
 p, gateyV pp qq "SIM", qq |=> rr, rr |=> End, gateyV pp ss
 "NAO", ss |=> mm]
}

spl = BPM {processes = [bpAjudaCustoComum, advAjudaCusto1, a
dvAjudaCusto2, advAuxilioMoradia]}
```

C.6 Caso VI - Licença - Licença Incentivada sem Remuneração e Licença para Capacitação - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Analisar requerimento", type' = Act
ivity, annotations' = [], parameters' = []}
```

```
cc :: FlowObject
cc = FlowObject { id' = "Autorizar requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

dd :: FlowObject
dd = FlowObject { id' = "Deferido?", type' = Gateway, annota
tions' = ["pcLicIncSemRem", "pcLicParaCapacitacao"], paramet
ers' = []}

ee :: FlowObject
ee = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

bpLicIncSemRemLicParaCapacitacaoComum = BusinessProcess {
 pid = "bpLicIncSemRemLicParaCapacitacaoComum",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, cc |=> d
 d, gateyV dd ee "NAO", ee |=> End, gateyV dd End "SIM"]
}

ff2 :: FlowObject
ff2 = FlowObject { id' = "Entregar documento expedido pela I
nstituicao organizadora", type' = Activity, annotations' = [
], parameters' = []}

ff3 :: FlowObject
ff3 = FlowObject { id' = "Arquivar AFD", type' = Activity, a
nnotations' = [], parameters' = []}

gg2 :: FlowObject
gg2 = FlowObject { id' = "Abrir processo", type' = Activity,
 annotations' = [], parameters' = []}

hh2 :: FlowObject
hh2 = FlowObject { id' = "Analisar processo", type' = Activi
ty, annotations' = [], parameters' = []}

jj2 :: FlowObject
jj2 = FlowObject { id' = "Emitir parecer", type' = Activity,
 annotations' = [], parameters' = []}

kk2 :: FlowObject
kk2 = FlowObject { id' = "Parecer favoravel?", type' = Gatew
ay, annotations' = [], parameters' = []}
```

```
ll2 :: FlowObject
ll2 = FlowObject { id' = "Dar ciencia ao servidor", type' =
Activity, annotations' = [], parameters' = []}

advLicParaCapacitacao = BusinessProcess {
 pid = "advLicParaCapacitacao",
 ptype = Advice {advType = After, pc = PCut "pcLicParaCapaci
 tacao" ["SIM"]},
 objects = [Start, ff2, ff3, gg2, hh2, jj2, kk2, ll2, mm3, n
 n, oo, pp, qq, rr, ss, End],
 transitions = [Start |=> ff2, ff2 |=> ff3, ff3 |=> gg2, gg2
  |=> hh2, hh2 |=> jj2, jj2 |=> kk2, gateyV kk2 ll2 "NAO", l
  l2 |=> End,
                gateyV kk2 mm3 "SIM", mm3 |=> nn, nn |=> oo,
                 oo |=> pp, pp |=> qq, qq |=> rr, rr |=> ss,
                  ss |=> End]
}

ff :: FlowObject
ff = FlowObject { id' = "Analisar requerimento", type' = Act
ivity, annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Autorizar requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

hh :: FlowObject
hh = FlowObject { id' = "Deferido?", type' = Gateway, annota
tions' = [], parameters' = []}

ii :: FlowObject
ii = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

jj :: FlowObject
jj = FlowObject { id' = "Analisar recursos financeiros", typ
e' = Activity, annotations' = [], parameters' = []}

kk :: FlowObject
kk = FlowObject { id' = "Ha recursos financeiros disponiveis
?", type' = Gateway, annotations' = [], parameters' = []}

ll :: FlowObject
ll = FlowObject { id' = "Elaborar requerimento de recursos f
inanceiros a area financeira", type' = Activity, annotations
' = [], parameters' = []}
```

```
mm :: FlowObject
mm = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

mm2 :: FlowObject
mm2 = FlowObject { id' = "Aguardar informacao de recursos fi
nanceiros", type' = Activity, annotations' = [], parameters'
 = []}

mm3 :: FlowObject
mm3 = FlowObject { id' = "Cadastro", type' = Activity, annot
ations' = [], parameters' = []}

nn :: FlowObject
nn = FlowObject { id' = "Realizar acertos financeiros", type
' = Activity, annotations' = [], parameters' = []}

oo :: FlowObject
oo = FlowObject { id' = "Elaborar Portaria", type' = Activit
y, annotations' = [], parameters' = []}

pp :: FlowObject
pp = FlowObject { id' = "Assinar", type' = Activity, annotat
ions' = [], parameters' = []}

qq :: FlowObject
qq = FlowObject { id' = "Registrar assinatura da Portaria",
type' = Activity, annotations' = [], parameters' = []}

rr :: FlowObject
rr = FlowObject { id' = "Enviar Portaria para publicacao no
Boletim de Servico", type' = Activity, annotations' = [], pa
rameters' = []}

ss :: FlowObject
ss = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

advLicIncSemRemuneracao = BusinessProcess {
 pid = "advLicIncSemRemuneracao",
 ptype = Advice {advType = After, pc = PCut "pcLicIncSemRem"
  ["SIM"]},
 objects = [Start, ff, gg, hh, ii, jj, kk, ll, mm, mm2, mm3,
  nn, oo, pp, qq, rr, ss, End],
 transitions = [Start |=> ff, ff |=> gg, gg |=> hh, gateyV h
```

```
 h ii "NAO", ii |=> End, gateyV hh jj "SIM", jj |=> kk,
                gateyV kk ll "NAO", ll |=> mm, mm |=> mm2, m
                m2 |=> jj, gateyV kk mm3 "SIM", mm3 |=> nn,
                nn |=> oo,
                oo |=> pp, pp |=> qq, qq |=> rr, rr |=> ss,
                ss |=> End]
}

spl = BPM {processes = [bpLicIncSemRemLicParaCapacitacaoComu
m, advLicParaCapacitacao, advLicIncSemRemuneracao]}
```

C.7 Caso VII - Adicional - Periculosidade e Insalubridade - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Elaborar memorando", type' = Activi
ty, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Analisar direito do servidor ao ben
eficio", type' = Activity, annotations' = [], parameters' =
[]}

cc :: FlowObject
cc = FlowObject { id' = "Servidor tem direito a mais de um b
eneficio?", type' = Gateway, annotations' = [], parameters'
= []}

dd :: FlowObject
dd = FlowObject { id' = "Solicitar analise ambiental", type'
 = Activity, annotations' = [], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Analisar ambiente", type' = Activit
y, annotations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Incluir laudo de condicoes ambienta
is", type' = Activity, annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Deferido?", type' = Gateway, annota
tions' = [], parameters' = []}
```

```
hh :: FlowObject
hh = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

ii :: FlowObject
ii = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

jj :: FlowObject
jj = FlowObject { id' = "Realizar acertos financeiros", type
' = Activity, annotations' = [], parameters' = [("pA",Unboun
d)]}

ll :: FlowObject
ll = FlowObject { id' = "Elaborar  boletim de servico", type
' = Activity, annotations' = [], parameters' = []}

mm :: FlowObject
mm = FlowObject { id' = "Publicar no boletim de servico", ty
pe' = Activity, annotations' = [], parameters' = []}

nn :: FlowObject
nn = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

bpPericulosidade = BusinessProcess {
 pid = "bpPericulosidade",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, ff, gg, hh, ii, jj, l
 l, mm, nn, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, gateyV c
 c dd "NAO", dd |=> ee, ee |=> ff, ff |=> gg, gg |=> hh, hh
 |=> ii, ii |=> End, gg |=> jj, jj |=> ll, ll |=> mm, mm |=>
  ii, gateyV cc nn "SIM", nn |=> dd]
}

spl = BPM {processes = [bpPericulosidade]}
```

C.8 Caso VIII - Auxílio - Auxílio Alimentação e Auxílio Transporte - BusinessProcessModel - Tipos Algébricos

```
a :: FlowObject
a = FlowObject { id' = "Registro", type' = Activity, annotat
ions' = ["pcAuxilio"], parameters' = []}
```

```
b :: FlowObject
b = FlowObject { id' = "O beneficio eh de desejo do servidor
?", type' = Gateway, annotations' = [], parameters' = []}

c :: FlowObject
c = FlowObject { id' = "Preenchimento de formulario", type'
= Activity, annotations' = [], parameters' = []}

d :: FlowObject
d = FlowObject { id' = "Analisar os documentos e condicoes d
o servidor pelo RH", type' = Activity, annotations' = [], pa
rameters' = []}

e :: FlowObject
e = FlowObject { id' = "Registro do valor, conforme informad
o o servidor, de acordo com a tabela de codigo do SIAPE", ty
pe' = Activity, annotations' = [], parameters' = []}

f :: FlowObject
f = FlowObject { id' = "Informar ao orgao por escrito que na
o opta pelo beneficio", type' = Activity, annotations' = [],
 parameters' = []}

g :: FlowObject
g = FlowObject { id' = "Arquivo", type' = Activity, annotati
ons' = [], parameters' = []}

bpAuxilioComum = BusinessProcess {
 pid = "bpAuxilioComum",
 ptype = BasicProcess,
 objects = [Start, a, b, c, d, e, f, g, End],
 transitions = [Start |=> a, a |=> b, gateyV b c "SIM", c |=
 > d, d |=> e, gateyV b f "NAO", f |=> g, g |=> End]
}

h :: FlowObject
h = FlowObject { id' = "Recebe o beneficio por outro vinculo
? ", type' = Gateway, annotations' = [], parameters' = []}

i :: FlowObject
i = FlowObject { id' = "Cancelar o beneficio no outro empreg
o", type' = Activity, annotations' = [], parameters' = []}

advAuxilioAlimentacao = BusinessProcess {
 pid = "advAuxilioAlimentacao",
```

```
 ptype = Advice {advType = After, pc = Pointcut "pcAuxilio"}
 ,
 objects = [Start, h, i, End],
 transitions = [Start |=> h, gateyV h End "NAO", gateyV h i
 "SIM", i |=> End]
}

spl = BPM {processes = [bpAuxilioComum, advAuxilioAlimentaca
o]}
```

C.9 Caso IX - Auxílio Filiação - Auxílio Natalidade e Auxílio Pré-Escolar - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Analisar informacoes dos documentos
", type' = Activity, annotations' = [], parameters' = []}

cc :: FlowObject
cc = FlowObject { id' = "Realizar acertos financeiros", type
' = Activity, annotations' = ["pcAuxilioPreescolar"], parame
ters' = []}

bpAuxilioNatalidade = BusinessProcess {
 pid = "bpAuxilioNatalidade",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, cc |=> E
 nd]
}

dd :: FlowObject
dd = FlowObject { id' = "Idade maxima alcancada?", type' = G
ateway, annotations' = [], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Possui o beneficio?", type' = Gatew
ay, annotations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Excluir beneficio", type' = Activit
```

```
y, annotations' = [], parameters' = []}

advAuxilioPreescolar = BusinessProcess {
 pid = "advAuxilioPreescolar",
 ptype = Advice {advType = Around, pc = Pointcut "pcAuxilioP
 reescolar"},
 objects = [Start, dd, ee, ff, cc, End],
 transitions = [Start |=> dd, gateyV dd ee "SIM", gateyV dd
 cc "NAO", gateyV ee ff "SIM", gateyV ee End "NAO", ff |=> E
 nd, cc |=> End]
}

spl = BPM {processes = [bpAuxilioNatalidade, advAuxilioPrees
colar]}
```

C.10 Caso X - Férias - Marcar e Férias e Remarcar Férias - Produto Final - Instância dos Tipos Algébricos

```
cc :: FlowObject
cc = FlowObject { id' = "Cadastrar", type' = Activity, annot
ations' = ["pcRemarcarFerias", "pcMarcarFerias"], parameters
' = []}

dd :: FlowObject
dd = FlowObject { id' = "Realizar acertos financeiros", type
' = Gateway, annotations' = ["pcServicoExtraordinario1"], pa
rameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

bpFeriasComum = BusinessProcess {
 pid = "bpFeriasComum",
 ptype = BasicProcess,
 objects = [Start, cc, dd, ee, ff, End],
 transitions = [Start |=> cc, cc |=> dd, dd |=> ee, ee |=> f
 f, ff |=> End]
}
```

```
ii :: FlowObject
ii = FlowObject { id' = "Ha documento para cancelamento  de
ferias?", type' = Gateway, annotations' = [], parameters' =
[]}

jj :: FlowObject
jj = FlowObject { id' = "Elaborar documento para cancelament
o", type' = Activity, annotations' = ["pcFeriasComum"], para
meters' = []}

pp = Proceed

advRemarcarFerias = BusinessProcess {
 pid = "advRemarcarFerias",
 ptype = Advice {advType = Around, pc = Pointcut "pcRemarcar
 Ferias"},
 objects = [Start, ii, jj, pp, End],
 transitions = [Start |=> ii, gateyV ii jj "NAO", jj |=> ii,
  gateyV ii pp "SIM", pp |=> End]
}

qq :: FlowObject
qq = FlowObject { id' = "Solicitar escala de ferias", type'
= Activity, annotations' = [], parameters' = []}

rr :: FlowObject
rr = FlowObject { id' = "+", type' = Gateway, annotations' =
 [], parameters' = []}

ss :: FlowObject
ss = FlowObject { id' = "Solicitar previsao de ferias", type
' = Activity, annotations' = [], parameters' = []}

tt :: FlowObject
tt = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

uu :: FlowObject
uu = FlowObject { id' = "Elaborar escala de ferias", type' =
 Activity, annotations' = [], parameters' = []}

zz :: FlowObject
zz = FlowObject { id' = "Analisar escala de ferias", type' =
 Activity, annotations' = [], parameters' = []}

xx :: FlowObject
```

```
xx = FlowObject { id’ = "Autorizar escala de ferias", type’
= Activity, annotations’ = [], parameters’ = []}

zz2 :: FlowObject
zz2 = FlowObject { id’ = "Escala autorizada?", type’ = Gatew
ay, annotations’ = [], parameters’ = []}

ww :: FlowObject
ww = FlowObject { id’ = "Solicitar alteracao de ferias", typ
e’ = Activity, annotations’ = [], parameters’ = []}

ww1 :: FlowObject
ww1 = FlowObject { id’ = "Parar contador do retorno da escal
a", type’ = Activity, annotations’ = [], parameters’ = []}

ww11 :: FlowObject
ww11 = FlowObject { id’ = "++", type’ = Gateway, annotations
’ = [], parameters’ = []}

ww2 :: FlowObject
ww2 = FlowObject { id’ = "Elaborar documento para assinatura
", type’ = Activity, annotations’ = ["pcFeriasComum"], param
eters’ = []}

ww3 :: FlowObject
ww3 = FlowObject { id’ = "Contar prazo de retorno da escala
(15 dias)", type’ = Activity, annotations’ = [], parameters’
 = []}

ww4 :: FlowObject
ww4 = FlowObject { id’ = "Dentro do prazo?", type’ = Gateway
, annotations’ = [], parameters’ = []}

pp2 = Proceed

advMarcarFerias = BusinessProcess {
 pid = "advMarcarFerias",
 ptype = Advice {advType = Around, pc = Pointcut "pcMarcarFe
 rias"},
 objects = [Start, qq, rr, ss, Start, tt, uu, zz, xx, zz2, w
 w, ww1, ww11, ww2, ww3, ww4, pp2, End],
 transitions = [Start |=> qq, qq |=> rr, rr |=> ss, ss |=> t
 t, tt |=> uu, uu |=> zz, zz |=> xx,
                xx |=> zz2, gateyV zz2 ww "NAO", ww |=> tt,
                Start |=> tt, gateyV zz2 ww1 "SIM",
                ww1 |=> ww11, ww11 |=> ww2, rr |=> ww3, ww3
```

```
                |=> ww4, gateyV ww4 ww11 "SIM",
                ww2 |=> pp2, pp2 |=> End]
}

aa :: FlowObject
aa = FlowObject { id' = "Assinar (Chefia imediata)", type' =
 Activity, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Assinar (Servidor)", type' = Activi
ty, annotations' = [], parameters' = []}

advFeriasComum = BusinessProcess {
 pid = "advFeriasComum",
 ptype = Advice {advType = After, pc = Pointcut "pcFeriasCom
 um"},
 objects = [Start, aa, bb, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> End]
}

spl = BPM {processes = [bpFeriasComum, advRemarcarFerias, ad
vMarcarFerias, advFeriasComum]}
```

C.11 Caso XI - Serviço - Serviço Extraordinário e Tempo de Serviço Quinquênio - Produto Final - Instância dos Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Analisar requerimento", type' = Act
ivity, annotations' = [], parameters' = []}

cc :: FlowObject
cc = FlowObject { id' = "Emitir parecer", type' = Activity,
annotations' = [], parameters' = []}

dd :: FlowObject
dd = FlowObject { id' = "Deferido?", type' = Gateway, annota
tions' = ["pcServicoExtraordinario1"], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Dar ciencia ao servidor", type' = A
```

```
ctivity, annotations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Autorizar execucao  do servico", ty
pe' = Activity, annotations' = ["pcServicoExtraordinario2",
"pcTempoDeServico1"], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Realizar acertos financeiros", type
' = Activity, annotations' = ["pcTempoDeServico2"], paramete
rs' = []}

bpServicoComum = BusinessProcess {
 pid = "bpServicoComum",
 ptype = BasicProcess,
 objects = [Start, aa, bb, cc, dd, ee, ff, gg, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> cc, cc |=> d
 d, gateyV dd ee "NAO", ee |=> End, gateyV dd ff "SIM", ff |
 => gg, gg |=> End]
}

hh :: FlowObject
hh = FlowObject { id' = "Dar ciencia a chefia", type' = Acti
vity, annotations' = [], parameters' = []}

advServicoExtraordinario1 = BusinessProcess {
 pid = "advServicoExtraordinario1",
 ptype = Advice {advType = After, pc = PCut "pcServicoExtrao
 rdinario1" ["NAO"]},
 objects = [Start, hh, End],
 transitions = [Start |=> hh, hh |=> End]
}

ii :: FlowObject
ii = FlowObject { id' = "Dar ciencia a chefia", type' = Acti
vity, annotations' = [], parameters' = []}

jj :: FlowObject
jj = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

kk :: FlowObject
kk = FlowObject { id' = "Elaborar relatorio de servicos extr
aordinarios", type' = Activity, annotations' = [], parameter
s' = []}
```

```
advServicoExtraordinario2 = BusinessProcess {
 pid = "advServicoExtraordinario2",
 ptype = Advice {advType = After, pc = Pointcut "pcServicoEx
 traordinario2"},
 objects = [Start, ii, jj, kk, End],
 transitions = [Start |=> ii, ii |=> jj, jj |=> kk, kk |=> E
 nd]
}

ll :: FlowObject
ll = FlowObject { id' = "Cadastro", type' = Activity, annota
tions' = [], parameters' = []}

mm :: FlowObject
mm = FlowObject { id' = "Elaborar boletim de  servico", type
' = Activity, annotations' = [], parameters' = []}

nn :: FlowObject
nn = FlowObject { id' = "Publicar boletim de servico", type'
 = Activity, annotations' = [], parameters' = []}

advTempoDeServico1 = BusinessProcess {
 pid = "advTempoDeServico1",
 ptype = Advice {advType = After, pc = Pointcut "pcTempoDeSe
 rvico1"},
 objects = [Start, ll, mm, nn, End],
 transitions = [Start |=> ll, ll |=> mm, mm |=> nn, nn |=> E
 nd]
}

oo :: FlowObject
oo = FlowObject { id' = "Arquivar AFD", type' = Activity, an
notations' = [], parameters' = []}

advTempoDeServico2 = BusinessProcess {
 pid = "advTempoDeServico2",
 ptype = Advice {advType = After, pc = Pointcut "pcTempoDeSe
 rvico2"},
 objects = [Start, oo, End],
 transitions = [Start |=> oo, oo |=> End]
}

spl = BPM {processes = [bpServicoComum, advServicoExtraordin
ario1, advServicoExtraordinario2, advTempoDeServico1, advTem
poDeServico2]}
```

C.12 Caso XII - Indenização - Indenização de Transporte e Transporte de Mobiliário e Bagagens - BusinessProcessModel - Tipos Algébricos

```
aa :: FlowObject
aa = FlowObject { id' = "Preencher requerimento", type' = Ac
tivity, annotations' = [], parameters' = []}

bb :: FlowObject
bb = FlowObject { id' = "Sinalizar CPROD para criacao do num
ero do processo", type' = Activity, annotations' = ["pcInden
izacao"], parameters' = []}

bpIndenizacaoComum = BusinessProcess {
 pid = "bpIndenizacaoComum",
 ptype = BasicProcess,
 objects = [Start, aa, bb, End],
 transitions = [Start |=> aa, aa |=> bb, bb |=> End]
}

cc :: FlowObject
cc = FlowObject { id' = "Verificar se houve emissao de passa
gens", type' = Activity, annotations' = [], parameters' = []
}

dd :: FlowObject
dd = FlowObject { id' = "Houve emissao de passagem?", type'
= Gateway, annotations' = [], parameters' = []}

ee :: FlowObject
ee = FlowObject { id' = "Analisar processo", type' = Activit
y, annotations' = [], parameters' = []}

ff :: FlowObject
ff = FlowObject { id' = "Elaborar parecer", type' = Activity
, annotations' = [], parameters' = []}

gg :: FlowObject
gg = FlowObject { id' = "Parecer favoravel?", type' = Gatewa
y, annotations' = [], parameters' = []}

hh :: FlowObject
hh = FlowObject { id' = "Cadastro", type' = Activity, annota
tions' = [], parameters' = []}
```

```
ii :: FlowObject
ii = FlowObject { id' = "Dar ciencia ao servidor", type' = A
ctivity, annotations' = [], parameters' = []}

jj :: FlowObject
jj = FlowObject { id' = "Realizar acertos financeiros", type
' = Activity, annotations' = [], parameters' = []}

kk :: FlowObject
kk = FlowObject { id' = "Cadastro (RH)", type' = Activity, a
nnotations' = [], parameters' = []}

ll :: FlowObject
ll = FlowObject { id' = "Dar ciencia ao servidor (RH)", type
' = Activity, annotations' = [], parameters' = []}

mm :: FlowObject
mm = FlowObject { id' = "Emitir passagem", type' = Activity,
 annotations' = [], parameters' = []}

advTransporte = BusinessProcess {
 pid = "advTransporte",
 ptype = Advice {advType = After, pc = Pointcut "pcIndenizac
 ao"},
 objects = [Start, cc, dd, ee, ff, gg, hh, ii, jj, kk, ll, m
 m, End],
 transitions = [Start |=> cc, cc |=> dd, gateyV dd ee "NAO",
  ee |=> ff, ff |=> gg, gateyV gg hh "NAO", hh |=> ii,
                ii |=> End, gateyV dd jj "SIM", jj |=> kk, k
                k |=> ll, ll |=> End, gateyV gg mm "SIM", mm
                 |=> hh]
}

ee2 :: FlowObject
ee2 = FlowObject { id' = "Analisar processo", type' = Activi
ty, annotations' = [], parameters' = []}

ff2 :: FlowObject
ff2 = FlowObject { id' = "Elaborar parecer", type' = Activit
y, annotations' = [], parameters' = []}

gg2 :: FlowObject
gg2 = FlowObject { id' = "Parecer favoravel?", type' = Gatew
ay, annotations' = [], parameters' = []}

hh2 :: FlowObject
```

```
hh2 = FlowObject { id' = "Dar ciencia ao servidor", type' =
Activity, annotations' = [], parameters' = []}

ii2 :: FlowObject
ii2 = FlowObject { id' = "Cadastro", type' = Activity, annot
ations' = [], parameters' = []}

jj2 :: FlowObject
jj2 = FlowObject { id' = "Organizar rmudanca com o servidor"
, type' = Activity, annotations' = [], parameters' = []}

kk2 :: FlowObject
kk2 = FlowObject { id' = "Aguardar concretizacao da mudanca"
, type' = Activity, annotations' = [], parameters' = []}

advMobBag = BusinessProcess {
 pid = "advMobBag",
 ptype = Advice {advType = After, pc = Pointcut "pcIndenizac
 ao"},
 objects = [Start, ee2, ff2, gg2, hh2, ii2, jj2, kk2, End],
 transitions = [Start |=> ee2, ee2 |=> ff2, ff2 |=> gg2, gat
 eyV gg2 hh2 "NAO", hh2 |=> ii2, ii2 |=> End, gateyV gg2 jj2
  "SIM", jj2 |=> kk2, kk2 |=> ii2]
}

spl = BPM {processes = [bpIndenizacaoComum, advTransporte, a
dvMobBag]}
```

Apêndice D

Resultados de cada PC - Tipos Algébricos Instanciados

D.1 Caso I - Transferência - Cessão e Requisição - Produto Final - Instância dos Tipos Algébricos

D.1.1 Instância do Tipo Algébrico quando somente feature Cessao for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpParteComumCessao
Requisicao",ptype = BasicProcess, objects = [(start),(Preenc
her requerimento),(Receber requerimento),(Encaminhar para SR
H),(Analisar),(Deferido?),(Dar ciencia ao orgao  requerente)
,(Analisar liberacao),(Concorda?),(Concorda?),(Incluir anuen
cia),(Elaborar Portaria),(Publicacao),(Cadastro),(Elaborar o
ficio de apresentacao do servidor),(Receber oficio de aprese
ntacao),(end)], transitions = [((start),(Preencher requerime
nto),""),((Preencher requerimento),(Receber requerimento),""
),((Receber requerimento),(Encaminhar para SRH),""),((Encami
nhar para SRH),(Analisar),""),((Analisar),(Deferido?),""),((
Deferido?),(Dar ciencia ao orgao requerente),"NAO"),((Darcie
ncia ao orgao  requerente),(end),""),((Deferido?),(Analisar
liberacao),"SIM"),((Analisar liberacao),(Concorda?),""),((Co
ncorda?),(Dar ciencia ao orgao requerente),"NAO"),((Concorda
?),(Concorda?),"SIM"),((Concorda?),(D ar ciencia ao orgao  r
equerente),"NAO"),((Concorda?),(Incluir anue ncia), "SIM"),(
(Incluir anuencia),(Elaborar Portaria),""), ((Elabo rar Port
aria),(Publicacao),""),((Publicacao), (Cadastro),""), ((Ca d
astro),(Elaborar oficio de apresentacao do  servidor),""),((
Elaborar oficio de apresentacao do servidor), (Receber ofici
o de apresentacao),""),((Receber oficio de  apresentacao),(e
nd),"")]}]}
```

D.1.2 Instância do Tipo Algébrico quando somente feature Requisicao for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpParteComumCessao
Requisicao",ptype = BasicProcess, objects = [(start),(Preenc
her requerimento),(Receber requerimento),(Encaminhar para SR
H),(Analisar),(Deferido?),(Dar ciencia ao orgao  requerente)
,(Analisar liberacao),(Concorda?),(Concorda?),(Incluir anuen
cia),(Elaborar Portaria),(Publicacao),(Cadastro),(Elaborar o
ficio de apresentacao do servidor),(Receber oficio de aprese
ntacao),(end)], transitions = [((start),(Preencher requerime
nto),""),((Preencher requerimento),(Receber requerimento),""
),((Receber requerimento),(Encaminhar para SRH),""),((Encami
nhar para SRH),(Analisar),""),((Analisar),(Deferido?),""),((
Deferido?),(Dar ciencia ao orgao requerente),"NAO"),((Darcie
ncia ao orgao  requerente),(end),""),((Deferido?),(Analisar
liberacao),"SIM"),((Analisar liberacao),(Concorda?),""),((Co
ncorda?),(Dar ciencia ao orgao requerente),"NAO"),((Concorda
?),(Concorda?),"SIM"),((Concorda?),(D ar ciencia ao orgao  r
equerente),"NAO"),((Concorda?),(Incluir anue ncia), "SIM"),(
(Incluir anuencia),(Elaborar Portaria),""), ((Elabo rar Port
aria),(Publicacao),""),((Publicacao), (Cadastro),""), ((Ca d
astro),(Elaborar oficio de apresentacao do  servidor),""),((
Elaborar oficio de apresentacao do servidor), (Receber ofici
o de apresentacao),""),((Receber oficio de  apresentacao),(e
nd),"")]}]}
```

D.1.3 Instância do Tipo Algébrico quando as features Cessao e Requisicao forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpParteComumCessao
Requisicao",ptype = BasicProcess, objects = [(start),(Preenc
her requerimento),(Receber requerimento),(Encaminhar para SR
H),(Analisar),(Deferido?),(Dar ciencia ao orgao  requerente)
,(Analisar liberacao),(Concorda?),(Concorda?),(Incluir anuen
cia),(Elaborar Portaria),(Publicacao),(Cadastro),(Elaborar o
ficio de apresentacao do servidor),(Receber oficio de aprese
ntacao),(end)], transitions = [((start),(Preencher requerime
nto),""),((Preencher requerimento),(Receber requerimento),""
),((Receber requerimento),(Encaminhar para SRH),""),((Encami
nhar para SRH),(Analisar),""),((Analisar),(Deferido?),""),((
Deferido?),(Dar ciencia ao orgao requerente),"NAO"),((Darcie
ncia ao orgao  requerente),(end),""),((Deferido?),(Analisar
liberacao),"SIM"),((Analisar liberacao),(Concorda?),""),((Co
ncorda?),(Dar ciencia ao orgao requerente),"NAO"),((Concorda
?),(Concorda?),"SIM"),((Concorda?),(D ar ciencia ao orgao  r
```

```
equerente),"NAO"),((Concorda?),(Incluir anue ncia), "SIM"),(
(Incluir anuencia),(Elaborar Portaria),""), ((Elabo rar Port
aria),(Publicacao),""),((Publicacao), (Cadastro),""), ((Ca d
astro),(Elaborar oficio de apresentacao do  servidor),""),((
Elaborar oficio de apresentacao do servidor), (Receber ofici
o de apresentacao),""),((Receber oficio de  apresentacao),(e
nd),"")]},BusinessProcess {pid = "bpParteComumCessao
Requisicao",ptype = BasicProcess, objects = [(start),(Preenc
her requerimento),(Receber requerimento),(Encaminhar para SR
H),(Analisar),(Deferido?),(Dar ciencia ao orgao  requerente)
,(Analisar liberacao),(Concorda?),(Concorda?),(Incluir anuen
cia),(Elaborar Portaria),(Publicacao),(Cadastro),(Elaborar o
ficio de apresentacao do servidor),(Receber oficio de aprese
ntacao),(end)], transitions = [((start),(Preencher requerime
nto),""),((Preencher requerimento),(Receber requerimento),""
),((Receber requerimento),(Encaminhar para SRH),""),((Encami
nhar para SRH),(Analisar),""),((Analisar),(Deferido?),""),((
Deferido?),(Dar ciencia ao orgao requerente),"NAO"),((Darcie
ncia ao orgao  requerente),(end),""),((Deferido?),(Analisar
liberacao),"SIM"),((Analisar liberacao),(Concorda?),""),((Co
ncorda?),(Dar ciencia ao orgao requerente),"NAO"),((Concorda
?),(Concorda?),"SIM"),((Concorda?),(D ar ciencia ao orgao  r
equerente),"NAO"),((Concorda?),(Incluir anue ncia), "SIM"),(
(Incluir anuencia),(Elaborar Portaria),""), ((Elabo rar Port
aria),(Publicacao),""),((Publicacao), (Cadastro),""), ((Ca d
astro),(Elaborar oficio de apresentacao do  servidor),""),((
Elaborar oficio de apresentacao do servidor), (Receber ofici
o de apresentacao),""),((Receber oficio de  apresentacao),(e
nd),"")]}]}
```

D.2 Caso II - Radiação - Raio-X ou Substância Radioativa e Irradiação Ionizante - Produto Final - Instância dos Tipos Algébricos

D.2.1 Instância do Tipo Algébrico quando somente feature Raio-X for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpParteComumRaioXR
adiacaoIonizante", ptype = BasicProcess, objects = [(start),
(Preencher requerimento),(Ha necessidade de realizar exame m
edico?),(Contar prazo (6 meses)),(Elaborar requisicao de exa
me medico),(Receber requisicao de exame medico),(Realizar ex
ame medico),(Entregar exame medico),(Receber exame medico),(
Incluir laudo medico),(Servidor esta apto?),(Realizar acerto
s finaceiros - Parameters: pR - 15%),(Elaborar  Portaria),(P
```

```
ublicacao),(Arquivar AFD),(Reconducao),(end)], transitions =
[((start),(Preencher requerimento),""),((Preencher requerime
nto),(Ha necessidade de realizar exame medico?),""),((Ha nec
essidade de realizar exame medico?),(Contar prazo (6 meses))
,"NAO"),((Ha necessidade de realizar exame medico?),(Elabora
r requisicao de exame medico),"SIM"),((Elaborar requisicao d
e exame medico),(Receber requisicao de exame medico),""),((R
eceber requisicaode exame medico),(Realizar exame medico),""
),((Realizar exame medico),(Entregar exame medico),""),((Ent
regar exame medico),(Receber exame medico),""),((Receber exa
me medico),(Incluir laudo medico),""),((Incluir laudo medico
),(Servidor esta apto?),""),((Servidor esta apto?),(Realizar
 acertos finaceiros - Parameters: pR - unbound),"SIM"),((Rea
lizar acertos finaceiros - Parameters: pR - unbound),(Elabor
ar  Portaria),""),((Elaborar  Portaria),(Publicacao),""),((P
ublicacao),(Arquivar AFD),""),((Arquivar AFD),(end),""),((Se
rvidor esta apto?),(Reconducao),"NAO"),((Reconducao),(end),"
")]}]}
```

D.2.2 Instância do Tipo Algébrico quando somente feature Irradiacao Ionizante for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpParteComumRaioXR
adiacaoIonizante", ptype = BasicProcess, objects = [(start),
(Preencher requerimento),(Ha necessidade de realizar exame m
edico?),(Contar prazo (6 meses)),(Elaborar requisicao de exa
me medico),(Receber requisicao de exame medico),(Realizar ex
ame medico),(Entregar exame medico),(Receber exame medico),(
Incluir laudo medico),(Servidor esta apto?),(Realizar acerto
s finaceiros - Parameters: pR - 30%),(Elaborar  Portaria),(P
ublicacao),(Arquivar AFD),(Reconducao),(end)], transitions =
[((start),(Preencher requerimento),""),((Preencher requerime
nto),(Ha necessidade de realizar exame medico?),""),((Ha nec
essidade de realizar exame medico?),(Contar prazo (6 meses))
,"NAO"),((Ha necessidade de realizar exame medico?),(Elabora
r requisicao de exame medico),"SIM"),((Elaborar requisicao d
e exame medico),(Receber requisicao de exame medico),""),((R
eceber requisicaode exame medico),(Realizar exame medico),""
),((Realizar exame medico),(Entregar exame medico),""),((Ent
regar exame medico),(Receber exame medico),""),((Receber exa
me medico),(Incluir laudo medico),""),((Incluir laudo medico
),(Servidor estaapto?),""),((Servidor esta apto?),(Realizar
acertos finaceiros - Parameters: pR - unbound),"SIM"),((Real
izar acertos finaceiros - Parameters: pR - unbound),(Elabora
r  Portaria),""),((Elaborar  Portaria),(Publicacao),""),((Pu
blicacao),(Arquivar AFD),""),((Arquivar AFD),(end),""),((Ser
```

```
vidor esta apto?),(Reconducao),"NAO"),((Reconducao),(end),""
)]}]}
```

D.2.3 Instância do Tipo Algébrico quando as features Raio-X e Irradiacao Ionizante forem selecionadas no PC

Com Base no CK Original - Instância do Tipo Algébrico quando as features Raio-X e Irradiacao Ionizante forem selecionadas no PC

```
BPM {processes =
[BusinessProcess {pid = "bpParteComumRaioXRadiacaoIonizante"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Ha necessidade de realizar exame medico?),(Contar p
razo (6 meses)),(Elaborar requisicao de exame medico),(Receb
er requisicao de exame medico),(Realizar exame medico),(Entr
egar exame medico),(Receber exame medico),(Incluir laudo med
ico),(Servidor esta apto?),(Realizar acertos finaceiros - Pa
rameters: pR - 30%),(Elaborar  Portaria),(Publicacao),(Arqui
var AFD),(Reconducao),(end)], transitions = [((start),(Preen
cher requerimento),""),((Preencher requerimento),(Ha necessi
dade de realizar exame medico?),""),((Ha necessidade de real
izar exame medico?),(Contar prazo (6 meses)),"NAO"),((Ha nec
essidade de realizar exame medico?),(Elaborar requisicao de
exame medico),"SIM"),((Elaborar requisicao de exame medico),
(Receber requisicao de exame medico),""),((Receber requisica
ode exame medico),(Realizar exame medico),""),((Realizar exa
me medico),(Entregarexame medico),""),((Entregar exame medic
o),(Receber exame medico),""),((Receberexame medico),(Inclui
r laudo medico),""),((Incluir laudo medico),(Servidor estaap
to?),""),((Servidor esta apto?),(Realizar acertos finaceiros
- Parameters: pR- unbound),"SIM"),((Realizar acertos finace
iros - Parameters: pR - unbound),(Elaborar  Portaria),""),((
Elaborar  Portaria),(Publicacao),""),((Publicacao),(Arquivar
AFD),""),((Arquivar AFD),(end),""),((Servidor esta apto?),(R
econducao),"NAO"),((Reconducao),(end),"")]},BusinessProcess
{pid = "bpParteComumRaioXRadiacaoIonizante", ptype =BasicPro
cess, objects = [(start),(Preencher requerimento),(Ha necess
idade de realizar exame medico?),(Contar prazo (6 meses)),(E
laborar requisicao de exame medico),(Receber requisicao de e
xame medico),(Realizar exame medico),(Entregar exame medico)
,(Receber exame medico),(Incluir laudo medico),(Servidor est
a apto?),(Realizar acertos finaceiros - Parameters: pR - 30%
),(Elaborar  Portaria),(Publicacao),(Arquivar AFD),(Reconduc
ao),(end)], transitions =[((start),(Preencher requerimento),
""),((Preencher requerimento),(Ha necessidade de realizar ex
ame medico?),""),((Ha necessidade de realizar exame medico?)
```

```
,(Contar prazo (6 meses)),"NAO"),((Ha necessidade de realiza
r exame medico?),(Elaborar requisicao de exame medico),"SIM"
),((Elaborar requisicao de exame medico),(Receber requisicao
de exame medico),""),((Receber requisicaode exame medico),(R
ealizar exame medico),""),((Realizar exame medico),(Entregar
exame medico),""),((Entregar exame medico),(Receber exame me
dico),""),((Receber exame medico),(Incluir laudo medico),"")
,((Incluir laudo medico),(Servidor estaapto?),""),((Servidor
 esta apto?),(Realizaracertos finaceiros - Parameters: pR -
 unbound),"SIM"),((Realizar acertos finaceiros - Parameters:
 pR - unbound),(Elaborar  Portaria),""),((Elaborar  Portaria
),(Publicacao),""),((Publicacao),(Arquivar AFD),""),((Arquiv
ar AFD),(end),""),((Servidor esta apto?),(Reconducao),"NAO")
,((Reconducao),(end),"")]}]}
```

Com Base no CK Corrigido - Instância do Tipo Algébrico quando as features Raio-X e Irradiacao Ionizante forem selecionadas no PC

```
BPM {processes =
[BusinessProcess {pid = "bpParteComumRaioXRadiacaoIonizante"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Ha necessidade de realizar exame medico?),(Contar p
razo (6 meses)),(Elaborar requisicao de exame medico),(Receb
er requisicao de exame medico),(Realizar exame medico),(Entr
egar exame medico),(Receber exame medico),(Incluir laudo med
ico),(Servidor esta apto?),(Realizar acertos finaceiros - Pa
rameters: pR - 15%),(Elaborar  Portaria),(Publicacao),(Arqui
var AFD),(Reconducao),(end)], transitions = [((start),(Preen
cher requerimento),""),((Preencher requerimento),(Ha necessi
dade de realizar exame medico?),""),((Ha necessidade de real
izar exame medico?),(Contar prazo (6 meses)),"NAO"),((Ha nec
essidade de realizar exame medico?),(Elaborar requisicao de
exame medico),"SIM"),((Elaborar requisicao de exame medico),
(Receber requisicao de exame medico),""),((Receber requisica
ode exame medico),(Realizar exame medico),""),((Realizar exa
me medico),(Entregarexame medico),""),((Entregar exame medic
o),(Receber exame medico),""),((Receberexame medico),(Inclui
r laudo medico),""),((Incluir laudo medico),(Servidor estaap
to?),""),((Servidor esta apto?),(Realizar acertos finaceiros
- Parameters: pR- unbound),"SIM"),((Realizar acertos finace
iros - Parameters: pR - unbound),(Elaborar  Portaria),""),((
Elaborar  Portaria),(Publicacao),""),((Publicacao),(Arquivar
AFD),""),((Arquivar AFD),(end),""),((Servidor esta apto?),(R
econducao),"NAO"),((Reconducao),(end),"")]},BusinessProcess
{pid = "bpParteComumRaioXRadiacaoIonizante", ptype =BasicPro
cess, objects = [(start),(Preencher requerimento),(Ha necess
```

```
idade de realizar exame medico?),(Contar prazo (6 meses)),(E
laborar requisicao de exame medico),(Receber requisicao de e
xame medico),(Realizar exame medico),(Entregar exame medico)
,(Receber exame medico),(Incluir laudo medico),(Servidor est
a apto?),(Realizar acertos finaceiros - Parameters: pR - 30%
),(Elaborar  Portaria),(Publicacao),(Arquivar AFD),(Reconduc
ao),(end)], transitions =[((start),(Preencher requerimento),
""),((Preencher requerimento),(Ha necessidade de realizar ex
ame medico?),""),((Ha necessidade de realizar exame medico?)
,(Contar prazo (6 meses)),"NAO"),((Ha necessidade de realiza
r exame medico?),(Elaborar requisicao de exame medico),"SIM"
),((Elaborar requisicao de exame medico),(Receber requisicao
de exame medico),""),((Receber requisicaode exame medico),(R
ealizar exame medico),""),((Realizar exame medico),(Entregar
exame medico),""),((Entregar exame medico),(Receber exame me
dico),""),((Receber exame medico),(Incluir laudo medico),"")
,((Incluir laudo medico),(Servidor estaapto?),""),((Servidor
 esta apto?),(Realizaracertos finaceiros - Parameters: pR -
 unbound),"SIM"),((Realizar acertos finaceiros - Parameters:
 pR - unbound),(Elaborar  Portaria),""),((Elaborar  Portaria
),(Publicacao),""),((Publicacao),(Arquivar AFD),""),((Arquiv
ar AFD),(end),""),((Servidor esta apto?),(Reconducao),"NAO")
,((Reconducao),(end),"")]}]}
```

D.3 Caso III - Desligamento - Demissão e Aposentadoria - Produto Final - Instância dos Tipos Algébricos

D.3.1 Instância do Tipo Algébrico quando somente feature Aposentadoria for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpDemissao", ptype
= BasicProcess, objects = [(start),(Receber processo),(Elab
orar Portaria),(Publicacao),(Cadastro),(Realizar acertos fi
nanceiros),(Sinalizar SISAC),(Arquivar AFD),(end),(Analisar
requisitos legais do processo)], transitions = [((start),(
Receber processo),""),((Elaborar Portaria),(Publicacao),""
),((Publicacao),(Cadastro),""),((Cadastro),(Realizar acert
os financeiros),""),((Realizar acertos financeiros),(Sinal
izar SISAC),""),((Sinalizar SISAC),(Arquivar AFD),""),((Ar
quivar AFD),(end),""),((Receber processo),(Analisar requis
itos legais do processo),""), ((Analisar requisitos legais
do processo),(Elaborar Portaria),"")]}]}
```

D.3.2 Instância do Tipo Algébrico quando somente feature Demissao for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpDemissao", ptype
= BasicProcess, objects = [(start),(Receber processo),(Elab
orar Portaria),(Publicacao),(Cadastro),(Realizar acertos fi
nanceiros),(Sinalizar SISAC),(Arquivar AFD),(end)], transit
ions = [((start),(Receber processo),""),((Receber processo)
,(Elaborar Portaria),""),((Elaborar Portaria),(Publicacao),
""),((Publicacao),(Cadastro),""),((Cadastro),(Realizar acer
tos financeiros),""),((Realizar acertos financeiros),(Sinal
izar SISAC),""),((Sinalizar SISAC),(Arquivar AFD),""),((Arq
uivar AFD),(end),"")]}]}
```

D.3.3 Com Base no CK Original - Instância do Tipo Algébrico quando as features Aposentadoria e Demissao forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpDemissao", ptype
= BasicProcess, objects = [(start),(Receber processo),(Elab
orar Portaria),(Publicacao),(Cadastro),(Realizar acertos fi
nanceiros),(Sinalizar SISAC),(Arquivar AFD),(end),(Analisar
requisitos legais do processo)], transitions = [((start),(
Receber processo),""),((Elaborar Portaria),(Publicacao),""
),((Publicacao),(Cadastro),""),((Cadastro),(Realizar acert
os financeiros),""),((Realizar acertos financeiros),(Sinal
izar SISAC),""),((Sinalizar SISAC),(Arquivar AFD),""),((Ar
quivar AFD),(end),""),((Receber processo),(Analisar requis
itos legais do processo),""),((Analisar requisitos legais
do processo),(Elaborar Portaria),"")]},BusinessProcess {pi
d = "bpDemissao", ptype = BasicProcess, objects = [(start)
,(Receber processo),(Elaborar Portaria),(Publicacao),(Cada
stro),(Realizar acertos financeiros),(Sinalizar SISAC),(Ar
quivar AFD),(end),(Analisar requisitos legais do processo)
], transitions = [((start),(Receber processo),""),((Elabor
ar Portaria),(Publicacao),""),((Publicacao),(Cadastro),"")
,((Cadastro),(Realizar acertos financeiros),""),((Realizar
acertos financeiros),(Sinalizar SISAC),""),((Sinalizar SI
SAC),(Arquivar AFD),""),((Arquivar AFD),(end),""),((Receb
er processo),(Analisar requisitos legais do processo),"")
,((Analisar requisitos legais do processo),(Elaborar Port
aria),"")]}]}
```

D.3.4 Com Base no CK Corrigido - Instância do Tipo Algébrico quando as features Aposentadoria e Demissao forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpDemissao", ptype
= BasicProcess, objects = [(start),(Receber processo),(Elab
orar Portaria),(Publicacao),(Cadastro),(Realizar acertos fi
nanceiros),(Sinalizar SISAC),(Arquivar AFD),(end)], transit
ions = [((start),(Receber processo),""),((Receber processo)
,(Elaborar Portaria),""),((Elaborar Portaria),(Publicacao),
""),((Publicacao),(Cadastro),""),((Cadastro),(Realizar acer
tos financeiros),""),((Realizar acertos financeiros),(Sinal
izar SISAC),""),((Sinalizar SISAC),(Arquivar AFD),""),((Arq
uivar AFD),(end),"")]},BusinessProcess {pid = "bpDemissao",
ptype = BasicProcess, objects = [(start),(Receber processo
),(Elaborar Portaria),(Publicacao),(Cadastro),(Realizar ac
ertos financeiros),(Sinalizar SISAC),(Arquivar AFD),(end),
(Analisar requisitos legais do processo)], transitions = [
((start),(Receber processo),""),((Elaborar Portaria),(Publ
icacao),""),((Publicacao),(Cadastro),""),((Cadastro),(Real
izar acertos financeiros),""),((Realizar acertos financeir
os),(Sinalizar SISAC),""),((Sinalizar SISAC),(Arquivar AFD
),""),((Arquivar AFD),(end),""),((Receber processo),(Anali
sar requisitos legais do processo),""),((Analisar requisit
os legais do processo),(Elaborar Portaria),"")]}]}
```

D.4 Caso IV - Ouvidoria - Reclamação e Elogio - Produto Final - Instâncias dos Tipos Algébricos

D.4.1 Instância do Tipo Algébrico quando somente feature Reclamacao for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpElogioComum", pt
ype = BasicProcess, objects = [(start),(Arquivar AFD),(O set
or do elogio eh externo?),(Encaminhar elogio para setor ou o
rgao externo),(Elaborar correspondencia com elogio),(Enviar
elogio ao servidor),(Receber elogio),(end),(Enviar Reclamaca
o),(Incluir Reclamacao),(Responder critica),(Analisar respos
ta),(Incluir satisfacao da resposta),(Sinalizar SisOuvidor)]
, transitions = [((Arquivar AFD),(O setor do elogio eh exter
no?),""),((O setor do elogio eh externo?),(Encaminhar elogio
para setor ou orgao externo),"SIM"),((O setor do elogio eh
externo?),(Elaborar correspondencia com elogio),"NAO"),((El
aborar correspondencia com elogio),(Enviar elogio ao servid
or),""),((Enviar elogio ao servidor),(Receber elogio),""),(
```

```
(start),(Enviar Reclamacao),""),((Enviar Reclamacao),(Inclu
ir Reclamacao),""),((Incluir Reclamacao),(Arquivar AFD),"")
,((Receber elogio),(Responder critica),""),((Sinalizar SisO
uvidor),(end),""),((Responder critica),(Analisar resposta),
""),((Analisar resposta),(Incluir satisfacao da resposta),"
"),((Incluir satisfacao da resposta),(Sinalizar SisOuvidor)
,"")]}]}
```

D.4.2 Instância do Tipo Algébrico quando somente feature Elogio for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpElogioComum", pt
ype = BasicProcess,objects = [(start),(Enviar Elogio),(Inclu
ir Elogio),(Arquivar AFD),(O setor do elogio eh externo?),(E
ncaminhar elogio para setor ou orgao externo),(Elaborar corr
espondencia com elogio),(Enviar elogio ao servidor),(Receber
elogio),(end)], transitions = [((start),(Enviar Elogio),"")
,((Enviar Elogio),(Incluir Elogio),""),((Incluir Elogio),(A
rquivar AFD),""),((Arquivar AFD),(O setor do elogio eh exte
rno?),""),((O setor do elogio eh externo?),(Encaminhar elog
io para setor ou orgao externo),"SIM"),((O setor do elogio
eh externo?),(Elaborar correspondencia com elogio),"NAO"),(
(Elaborar correspondencia com elogio),(Enviar elogio ao ser
vidor),""),((Enviar elogio ao servidor),(Receber elogio),""
),((Receber elogio),(end),"")]}]}
```

D.4.3 Instância do Tipo Algébrico quando as features Reclamacao e Elogio forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpElogioComum", pt
ype = BasicProcess,objects = [(start),(Enviar Elogio),(Inclu
ir Elogio),(Arquivar AFD),(O setor do elogio eh externo?),(E
ncaminhar elogio para setor ou orgao externo),(Elaborar corr
espondencia com elogio),(Enviar elogio ao servidor),(Receber
elogio),(end)], transitions = [((start),(Enviar Elogio),"")
,((Enviar Elogio),(Incluir Elogio),""),((Incluir Elogio),(A
rquivar AFD),""),((Arquivar AFD),(O setor do elogio eh exte
rno?),""),((O setor do elogio eh externo?),(Encaminhar elog
io para setor ou orgao externo),"SIM"),((O setor do elogio
eh externo?),(Elaborar correspondencia comelogio),"NAO"),((
Elaborar correspondencia com elogio),(Enviar elogio ao serv
idor),""),((Enviar elogio ao servidor),(Receber elogio),"")
,((Receber elogio),(end),"")]},BusinessProcess {pid = "bpEl
ogioComum", ptype = BasicProcess, objects = [(start),(Arqui
var AFD),(O setor do elogio eh externo?),(Encaminhar elogio
para setor ou orgao externo),(Elaborar correspondencia com
```

```
elogio),(Enviar elogio ao servidor),(Receber elogio),(end
),(Enviar Reclamacao),(Incluir Reclamacao),(Responder cri
tica),(Analisar resposta),(Incluir satisfacao da resposta
),(Sinalizar SisOuvidor)], transitions = [((Arquivar AFD)
,(O setor do elogio eh externo?),""),((O setor do elogio
eh externo?),(Encaminhar elogio para setor ou orgao exter
no),"SIM"),((O setor do elogio eh externo?),(Elaborar cor
respondencia com elogio),"NAO"),((Elaborar correspondenci
a com elogio),(Enviar elogio ao servidor),""),((Enviar el
ogio ao servidor),(Receber elogio),""),((start),(Enviar R
eclamacao),""),((Enviar Reclamacao),(Incluir Reclamacao),
""),((Incluir Reclamacao),(Arquivar AFD),""),((Receber el
ogio),(Responder critica),""),((Sinalizar SisOuvidor),(en
d),""),((Responder critica),(Analisar resposta),""),((Ana
lisar resposta),(Incluir satisfacao da resposta),""),((In
cluir satisfacao da resposta),(Sinalizar SisOuvidor),"")]
}]}
```

D.5 Caso V - Complemento Salarial - Ajuda de Custo e Auxílio Moradia - Produto Final - Instância dos Tipos Algébricos

D.5.1 Instância do Tipo Algébrico quando somente feature Ajuda de Custo for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpAjudaCustoComum"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Analisar requerimento e documentacao),(Autorizar re
querimento),(Deferido?),(Dar ciencia ao servidor),(Sinalizar
SIAFI),(Emissao de ordem bancaria),(Cadastro),(Dar ciencia
ao servidor),(end),(Entregar documentos comprobatorios),(Ar
quivar AFD),(Cadastro)], transitions =[((start),(Preencher
requerimento),""),((Analisar requerimento e documentacao),(
Autorizar requerimento),""),((Autorizar requerimento),(Defe
rido?),""),((Dar ciencia ao servidor),(end),""),((Deferido?
),(Sinalizar SIAFI),"SIM"),((Sinalizar SIAFI),(Emissao de o
rdem bancaria),""),((Emissao de ordem bancaria),(Cadastro),
""),((Cadastro),(Dar ciencia ao servidor),""),((Dar ciencia
ao servidor),(end),""),((Preencher requerimento),(Entregar
documentos comprobatorios),""),((Arquivar AFD),(Analisar
requerimento e documentacao),""),((Entregar documentos co
mprobatorios),(Arquivar AFD),""),((Deferido?),(Cadastro),
"NAO"),((Cadastro),(Dar ciencia ao servidor),"")]}]}
```

D.5.2 Instância do Tipo Algébrico quando somente feature Auxilio Moradia for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpAjudaCustoComum"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Analisar requerimento e documentacao),(Autorizar re
querimento),(Deferido?),(Dar ciencia ao servidor),(Sinalizar
SIAFI),(Emissao de ordem bancaria),(Cadastro),(Dar ciencia
ao servidor),(end),(Enviar, mensalmente, comprovante de des
pesas de moradia),(Analisar comprovante),(Elaborar parecer)
,(Parecer favoravel?),(Arquivar AFD),(Realizar acertos fina
nceiros),(Dar ciencia ao servidor)], transitions = [((start
),(Preencher requerimento),""),((Preencher requerimento),(A
nalisar requerimento e documentacao),""),((Analisar requeri
mento e documentacao),(Autorizar requerimento),""),((Autori
zar requerimento),(Deferido?),""),((Deferido?),(Dar ciencia
ao servidor),"NAO"),((Dar ciencia ao servidor),(end),""),(
(Sinalizar SIAFI),(Emissao de ordem bancaria),""),((Emissa
o de ordem bancaria),(Cadastro),""),((Cadastro),(Dar cienc
ia ao servidor),""),((Dar ciencia ao servidor),(end),""),(
(Deferido?),(Enviar, mensalmente, comprovante de despesas
de moradia),"SIM"),((Realizar acertos financeiros),(Sinali
zar SIAFI),""),((Enviar, mensalmente, comprovante de despe
sas de moradia),(Analisar comprovante),""),((Analisar comp
rovante),(Elaborar parecer),""),((Elaborar parecer),(Parec
er favoravel?),""),((Parecer favoravel?),(Arquivar AFD),"S
IM"),((Arquivar AFD),(Realizar acertos financeiros),""),((
Parecer favoravel?),(Dar ciencia ao servidor),"NAO"),((Dar
ciencia ao servidor),(Enviar, mensalmente, comprovante de
despesas de moradia),"")]}]}
```

D.5.3 Com Base no CK Original - Instância do Tipo Algébrico quando as features Ajuda de Custo e Auxilio Moradia forem selecionadas no PC

```
BPM {processes = [,BusinessProcess {pid = "bpAjudaCustoComum
", ptype = BasicProcess, objects = [(start),(Preencher reque
rimento),(Analisar requerimento e documentacao),(Autorizar r
equerimento),(Deferido?),(Dar ciencia ao servidor),(Sinaliza
r SIAFI),(Emissao de ordem bancaria),(Cadastro),(Dar ciencia
ao servidor),(end),(Entregar documentos comprobatorios),(Ar
quivar AFD),(Cadastro),(Enviar, mensalmente, comprovante de
despesas de moradia),(Analisar comprovante),(Elaborar pare
cer),(Parecer favoravel?),(Arquivar AFD),(Realizar acertos
financeiros),(Dar ciencia ao servidor)], transitions = [(
(start),(Preencher requerimento),""),((Analisar requerime
```

nto e documentacao),(Autorizar requerimento),""),((Autori
zar requerimento),(Deferido?),""),((Dar ciencia ao servid
or),(end),""),((Sinalizar SIAFI),(Emissao de ordem bancar
ia),""),((Emissao de ordem bancaria),(Cadastro),""),((Cad
astro),(Dar ciencia ao servidor),""),((Dar ciencia ao ser
vidor),(end),""),((Preencher requerimento),(Entregar docu
mentos comprobatorios),""),((Arquivar AFD),(Analisar requ
erimento e documentacao),""),((Entregar documentos com pr
obatorios),(Arquivar AFD),""),((Deferido?),(Cadastro),"NA
O"),((Cadastro),(Dar ciencia ao servidor),""),((Deferido?
),(Enviar, mensalmente, comprovante de despesas de moradi
a),"SIM"),((Realizar acertos financeiros),(Sinalizar SIAF
I),""),((Enviar, mensalmente, comprovante de despesas de
moradia),(Analisar comprovante),""),((Analisar comprovant
e),(Elaborar parecer),""),((Elaborar parecer),(Parecer fa
voravel?),""),((Parecer favoravel?),(Arquivar AFD),"SIM")
,((Arquivar AFD),(Realizar acertos financeiros),""),((Par
ecer favoravel?),(Dar ciencia ao servidor),"NAO"),((Dar c
iencia ao servidor),(Enviar, mensalmente, comprovante de
despesas de moradia),"")]},BusinessProcess {pid = "bpAjud
aCustoComum", ptype = BasicProcess, objects = [(start),(P
reencher requerimento),(Analisar requerimento e documenta
cao),(Autorizar requerimento),(Deferido?),(Dar ciencia ao
servidor),(Sinalizar SIAFI),(Emissao de ordem bancaria),
(Cadastro),(Dar ciencia ao servidor),(end),(Enviar, mens
almente, comprovante de despesas de moradia),(Analisar c
omprovante),(Elaborar parecer),(Parecer favoravel?),(Arq
uivar AFD),(Realizar acertos financeiros),(Dar ciencia a
o servidor)], transitions = [((start),(Preencher requeri
mento),""),((Preencher requerimento),(Analisar requerime
nto e documentacao),""),((Analisar requerimento e docume
ntacao),(Autorizar requerimento),""),((Autorizar requeri
mento),(Deferido?),""),((Deferido?),(Dar ciencia ao serv
idor),"NAO"),((Dar ciencia ao servidor),(end),""),((Sina
lizar SIAFI),(Emissao de ordem bancaria),""),((Emissao d
e ordem bancaria),(Cadastro),""),((Cadastro),(Dar cienci
a ao servidor),""),((Dar ciencia ao servidor),(end),""),
((Deferido?),(Enviar, mensalmente, comprovante de despes
as de moradia),"SIM"),((Realizar acertos financeiros),(S
inalizar SIAFI),""),((Enviar, mensalmente, comprovante d
e despesas de moradia),(Analisar comprovante),""),((Anal
isar comprovante),(Elaborar parecer),""),((Elaborar pare
cer),(Parecer favoravel?),""),((Parecer favoravel?),(Arq
uivar AFD),"SIM"),((Arquivar AFD),(Realizar acertos fina
nceiros),""),((Parecer favoravel?),(Dar ciencia ao servi
dor),"NAO"),((Dar ciencia ao servidor),(Enviar, mensalme

```
nte, comprovante de despesas de moradia),"")]}]}
```

D.5.4 Com Base no CK Corrigido - Instância do Tipo Algébrico quando as features Ajuda de Custo e Auxilio Moradia forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpAjudaCustoComum"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Analisar requerimento e documentacao),(Autorizar re
querimento),(Deferido?),(Dar ciencia ao servidor),(Sinalizar
SIAFI),(Emissao de ordem bancaria),(Cadastro),(Dar ciencia
ao servidor),(end),(Entregar documentos comprobatorios),(Ar
quivar AFD),(Cadastro)], transitions = [((start),(Preencher
requerimento),""),((Analisar requerimento e documentacao),
(Autorizar requerimento),""),((Autorizar requerimento),(De
ferido?),""),((Dar ciencia ao servidor),(end),""),((Deferi
do?),(Sinalizar SIAFI),"SIM"),((Sinalizar SIAFI),(Emissao
de ordem bancaria),""),((Emissao de ordem bancaria),(Cadas
tro),""),((Cadastro),(Dar ciencia ao servidor),""),((Dar c
iencia ao servidor),(end),""),((Preencher requerimento),(E
ntregar documentos comprobatorios),""),((Arquivar AFD),(An
alisar requerimento e documentacao),""),((Entregar documen
tos comprobatorios),(Arquivar AFD),""),((Deferido?),(Cadas
tro),"NAO"),((Cadastro),(Dar ciencia ao servidor),"")]},Bu
sinessProcess {pid = "bpAjudaCustoComum", ptype = BasicPro
cess, objects = [(start),(Preencher requerimento),(Analisa
r requerimento e documentacao),(Autorizar requerimento),(D
eferido?),(Dar ciencia ao servidor),(Sinalizar SIAFI),(Emi
ssao de ordem bancaria),(Cadastro),(Dar ciencia ao servido
r),(end),(Enviar, mensalmente, comprovante de despesas de
moradia),(Analisar comprovante),(Elaborar parecer),(Parece
r favoravel?),(Arquivar AFD),(Realizar acertos financeiros
),(Dar ciencia ao servidor)], transitions = [((start),(Pre
encher requerimento),""),((Preencher requerimento),(Analis
ar requerimento e documentacao),""),((Analisar requeriment
o e documentacao),(Autorizar requerimento),""),((Autorizar
requerimento),(Deferido?),""),((Deferido?),(Dar ciencia a
o servidor),"NAO"),((Dar ciencia ao servidor),(end),""),(
(Sinalizar SIAFI),(Emissao de ordem bancaria),""),((Emiss
ao de ordem bancaria),(Cadastro),""),((Cadastro),(Dar cie
ncia ao servidor),""),((Dar ciencia ao servidor),(end),""
),((Deferido?),(Enviar, mensalmente, comprovante de despe
sas de moradia),"SIM"),((Realizar acertos financeiros),(S
inalizar SIAFI),""),((Enviar, mensalmente, comprovante de
despesas de moradia),(Analisar comprovante),""),((Analis
ar comprovante),(Elaborar parecer),""),((Elaborar parece
```

```
r),(Parecer favoravel?),""),((Parecer favoravel?),(Arqui
var AFD),"SIM"),((Arquivar AFD),(Realizar acertos financ
eiros),""),((Parecer favoravel?),(Dar ciencia ao servido
r),"NAO"),((Dar ciencia ao servidor),(Enviar, mensalment
e, comprovante de despesas de moradia),"")]}]}
```

D.6 Caso VI - Licença - Capacitação e Incentivada sem Remuneração - Produto Final - Instância dos Tipos Algébricos

D.6.1 Instância do Tipo Algébrico quando somente feature Capacitacao for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpLicIncSemRemLicP
araCapacitacaoComum"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Analisar re
querimento),(Autorizar requerimento),(Deferido?),(Dar cienci
a ao servidor),(end)
,(Entregar documento expedido pela Instituicao organizadora)
,(Arquivar AFD),(Abr
ir processo),(Analisar processo),(Emitir parecer),(Parecer f
avoravel?),(Dar cien
cia ao servidor),(Cadastro),(Realizar acertos financeiros),(
Elaborar Portaria),(
Assinar),(Registrar assinatura da Portaria),(Enviar Portaria
 para publicacao no
Boletim de Servico),(Dar ciencia ao servidor)], transitions
= [((start),(Preench
er requerimento),""),((Preencher requerimento),(Analisar req
uerimento),""),((Ana
lisar requerimento),(Autorizar requerimento),""),((Autorizar
 requerimento),(Defe
rido?),""),((Deferido?),(Dar ciencia ao servidor),"NAO"),((D
ar ciencia ao servid
or),(end),""),((Deferido?),(Entregar documento expedido pela
 Instituicao organiz
adora),"SIM"),((Dar ciencia ao servidor),(end),""),((Dar cie
ncia ao servidor),(e
nd),""),((Entregar documento expedido pela Instituicao organ
izadora),(Arquivar A
FD),""),((Arquivar AFD),(Abrir processo),""),((Abrir process
o),(Analisar process
o),""),((Analisar processo),(Emitir parecer),""),((Emitir pa
recer),(Parecer favo
```

```
ravel?),""),((Parecer favoravel?),(Dar ciencia ao servidor),
"NAO"),((Parecer fav
oravel?),(Cadastro),"SIM"),((Cadastro),(Realizar acertos fin
anceiros),""),((Real
izar acertos financeiros),(Elaborar Portaria),""),((Elaborar
 Portaria),(Assinar)
,""),((Assinar),(Registrar assinatura da Portaria),""),((Reg
istrar assinatura da
 Portaria),(Enviar Portaria para publicacao no Boletim de Se
 rvico),""),((Enviar
Portaria para publicacao no Boletim de Servico),(Dar ciencia
 ao servidor),"")]}]
}
```

D.6.2 Instância do Tipo Algébrico quando somente feature Incentivada sem remuneracao for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpLicIncSemRemLicP
araCapacitacaoComum"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Analisar re
querimento),(Autorizar requerimento),(Deferido?),(Dar cienci
a ao servidor),(end)
,(Analisar requerimento),(Autorizar requerimento),(Deferido?
),(Deferido?),(Dar c
iencia ao servidor),(Analisar recursos financeiros),(Ha recu
rsos financeiros dis
poniveis?),(Elaborar requerimento de recursos financeiros a
area financeira),(Da
r ciencia ao servidor),(Aguardar informacao de recursos fina
nceiros),(Cadastro),
(Realizar acertos financeiros),(Elaborar Portaria),(Assinar)
,(Registrar assinatu
ra da Portaria),(Enviar Portaria para publicacao no Boletim
de Servico),(Dar cie
ncia ao servidor)], transitions = [((start),(Preencher reque
rimento),""),((Preen
cher requerimento),(Analisar requerimento),""),((Analisar re
querimento),(Autoriz
ar requerimento),""),((Autorizar requerimento),(Deferido?),"
"),((Deferido?),(Dar
 ciencia ao servidor),"NAO"),((Dar ciencia ao servidor),(end
 ),""),((Deferido?),(
Analisar requerimento),"SIM"),((Dar ciencia ao servidor),(en
d),""),((Dar ciencia
 ao servidor),(end),""),((Analisar requerimento),(Autorizar
```

```
 requerimento),""),((
Autorizar requerimento),(Deferido?),""),((Deferido?),(Dar ci
encia ao servidor),"
NAO"),((Deferido?),(Analisar recursos financeiros),"SIM"),((
Analisar recursos fi
nanceiros),(Ha recursos financeiros disponiveis?),""),((Ha r
ecursos financeiros
disponiveis?),(Elaborar requerimento de recursos financeiros
 a area financeira),
"NAO"),((Elaborar requerimento de recursos financeiros a are
a financeira),(Dar c
iencia ao servidor),""),((Dar ciencia ao servidor),(Aguardar
 informacao de recur
sos financeiros),""),((Aguardar informacao de recursos finan
ceiros),(Analisar re
cursos financeiros),""),((Ha recursos financeiros disponivei
s?),(Cadastro),"SIM"
),((Cadastro),(Realizar acertos financeiros),""),((Realizar
acertos financeiros)
,(Elaborar Portaria),""),((Elaborar Portaria),(Assinar),""),
((Assinar),(Registra
r assinatura da Portaria),""),((Registrar assinatura da Port
aria),(Enviar Portar
ia para publicacao no Boletim de Servico),""),((Enviar Porta
ria para publicacao
no Boletim de Servico),(Dar ciencia ao servidor),"")]}]}
```

D.6.3 Instância do Tipo Algébrico quando as features Capacitacao e Incentivada sem Remuneracao forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpLicIncSemRemLicP
araCapacitacaoComum"
, ptype = BasicProcess, objects = [(start),(Preencher requer
imento),(Analisar re
querimento),(Autorizar requerimento),(Deferido?),(Dar cienci
a ao servidor),(end)
,(Entregar documento expedido pela Instituicao organizadora)
,(Arquivar AFD),(Abr
ir processo),(Analisar processo),(Emitir parecer),(Parecer f
avoravel?),(Dar cien
cia ao servidor),(Cadastro),(Realizar acertos financeiros),(
Elaborar Portaria),(
Assinar),(Registrar assinatura da Portaria),(Enviar Portaria
 para publicacao no
Boletim de Servico),(Dar ciencia ao servidor)], transitions
```

```
= [((start),(Preench
er requerimento),""),((Preencher requerimento),(Analisar req
uerimento),""),((Ana
lisar requerimento),(Autorizar requerimento),""),((Autorizar
 requerimento),(Defe
rido?),""),((Deferido?),(Dar ciencia ao servidor),"NAO"),((D
ar ciencia ao servid
or),(end),""),((Deferido?),(Entregar documento expedido pela
 Instituicao organiz
adora),"SIM"),((Dar ciencia ao servidor),(end),""),((Dar cie
ncia ao servidor),(e
nd),""),((Entregar documento expedido pela Instituicao organ
izadora),(Arquivar A
FD),""),((Arquivar AFD),(Abrir processo),""),((Abrir process
o),(Analisar process
o),""),((Analisar processo),(Emitir parecer),""),((Emitir pa
recer),(Parecer favo
ravel?),""),((Parecer favoravel?),(Dar ciencia ao servidor),
"NAO"),((Parecer fav
oravel?),(Cadastro),"SIM"),((Cadastro),(Realizar acertos fin
anceiros),""),((Real
izar acertos financeiros),(Elaborar Portaria),""),((Elaborar
 Portaria),(Assinar)
,""),((Assinar),(Registrar assinatura da Portaria),""),((Reg
istrar assinatura da
 Portaria),(Enviar Portaria para publicacao no Boletim de Se
 rvico),""),((Enviar
Portaria para publicacao no Boletim de Servico),(Dar ciencia
 ao servidor),"")]},
BusinessProcess {pid = "bpLicIncSemRemLicParaCapacitacaoComu
m", ptype = BasicPro
cess, objects = [(start),(Preencher requerimento),(Analisar
requerimento),(Autor
izar requerimento),(Deferido?),(Dar ciencia ao servidor),(en
d),(Analisar requeri
mento),(Autorizar requerimento),(Deferido?),(Deferido?),(Dar
 ciencia ao servidor
),(Analisar recursos financeiros),(Ha recursos financeiros d
isponiveis?),(Elabor
ar requerimento de recursos financeiros a area financeira),(
Dar ciencia ao servi
dor),(Aguardar informacao de recursos financeiros),(Cadastro
),(Realizar acertos
financeiros),(Elaborar Portaria),(Assinar),(Registrar assina
tura da Portaria),(E
nviar Portaria para publicacao no Boletim de Servico),(Dar c
```

```
iencia ao servidor)]
, transitions = [((start),(Preencher requerimento),""),((Pre
encher requerimento)
,(Analisar requerimento),""),((Analisar requerimento),(Autor
izar requerimento),"
"),((Autorizar requerimento),(Deferido?),""),((Deferido?),(D
ar ciencia ao servid
or),"NAO"),((Dar ciencia ao servidor),(end),""),((Deferido?)
,(Analisar requerime
nto),"SIM"),((Dar ciencia ao servidor),(end),""),((Dar cienc
ia ao servidor),(end
),""),((Analisar requerimento),(Autorizar requerimento),""),
((Autorizar requerim
ento),(Deferido?),""),((Deferido?),(Dar ciencia ao servidor)
,"NAO"),((Deferido?)
,(Analisar recursos financeiros),"SIM"),((Analisar recursos
financeiros),(Ha rec
ursos financeiros disponiveis?),""),((Ha recursos financeiro
s disponiveis?),(Ela
borar requerimento de recursos financeiros a area financeira
),"NAO"),((Elaborar
requerimento de recursos financeiros a area financeira),(Dar
 ciencia ao servidor
),""),((Dar ciencia ao servidor),(Aguardar informacao de rec
ursos financeiros),"
"),((Aguardar informacao de recursos financeiros),(Analisar
recursos financeiros
),""),((Ha recursos financeiros disponiveis?),(Cadastro),"SI
M"),((Cadastro),(Rea
lizar acertos financeiros),""),((Realizar acertos financeiro
s),(Elaborar Portari
a),""),((Elaborar Portaria),(Assinar),""),((Assinar),(Regist
rar assinatura da Po
rtaria),""),((Registrar assinatura da Portaria),(Enviar Port
aria para publicacao
 no Boletim de Servico),""),((Enviar Portaria para publicaca
 o no Boletim de Serv
ico),(Dar ciencia ao servidor),"")]}]}
```

D.7 Caso VII - Adicional - Periculosidade e Insalubridade - Produto Final - Instância dos Tipos Algébricos

D.7.1 Instância do Tipo Algébrico quando somente feature Periculosidade for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpPericulosidade",
 ptype = BasicProces
s, objects = [(start),(Elaborar memorando),(Analisar direito
 do servidor ao bene
ficio),(Servidor tem direito a mais de um beneficio?),(Solic
itar analise ambient
al),(Analisar ambiente),(Incluir laudo de condicoes ambienta
is),(Deferido?),(Dar
 ciencia ao servidor),(Arquivar AFD),(Realizar acertos finan
 ceiros - Parameters:
 ParamPericulosidade - unbound,ParamInsalubridade - unbound)
 ,(Elaborar  boletim
de servico),(Publicar no boletim de servico),(Preencher requ
erimento),(end)], tr
ansitions = [((start),(Elaborar memorando),""),((Elaborar me
morando),(Analisar d
ireito do servidor ao beneficio),""),((Analisar direito do s
ervidor ao beneficio
),(Servidor tem direito a mais de um beneficio?),""),((Servi
dor tem direito a ma
is de um beneficio?),(Solicitar analise ambiental),"NAO"),((
Solicitar analise am
biental),(Analisar ambiente),""),((Analisar ambiente),(Inclu
ir laudo de condicoe
s ambientais),""),((Incluir laudo de condicoes ambientais),(
Deferido?),""),((Def
erido?),(Dar ciencia ao servidor),""),((Dar ciencia ao servi
dor),(Arquivar AFD),
""),((Arquivar AFD),(end),""),((Deferido?),(Realizar acertos
 financeiros - Param
eters: ParamPericulosidade - unbound,ParamInsalubridade - un
bound),""),((Realiza
r acertos financeiros - Parameters: ParamPericulosidade - un
bound,ParamInsalubri
dade - unbound),(Elaborar  boletim de servico),""),((Elabora
r  boletim de servic
o),(Publicar no boletim de servico),""),((Publicar no boleti
m de servico),(Arqui
```

```
var AFD),""),((Servidor tem direito a mais de um beneficio?)
,(Preencher requerim
ento),"SIM"),((Preencher requerimento),(Solicitar analise am
biental),"")]}]}
```

D.7.2 Instância do Tipo Algébrico quando somente feature Insalubridade for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpPericulosidade",
 ptype = BasicProces
s, objects = [(start),(Elaborar memorando),(Analisar direito
 do servidor ao bene
ficio),(Servidor tem direito a mais de um beneficio?),(Solic
itar analise ambient
al),(Analisar ambiente),(Incluir laudo de condicoes ambienta
is),(Deferido?),(Dar
 ciencia ao servidor),(Arquivar AFD),(Realizar acertos finan
 ceiros - Parameters:
 ParamPericulosidade - unbound,ParamInsalubridade - unbound)
 ,(Elaborar  boletim
de servico),(Publicar no boletim de servico),(Preencher requ
erimento),(end)], tr
ansitions = [((start),(Elaborar memorando),""),((Elaborar me
morando),(Analisar d
ireito do servidor ao beneficio),""),((Analisar direito do s
ervidor ao beneficio
),(Servidor tem direito a mais de um beneficio?),""),((Servi
dor tem direito a ma
is de um beneficio?),(Solicitar analise ambiental),"NAO"),((
Solicitar analise am
biental),(Analisar ambiente),""),((Analisar ambiente),(Inclu
ir laudo de condicoe
s ambientais),""),((Incluir laudo de condicoes ambientais),(
Deferido?),""),((Def
erido?),(Dar ciencia ao servidor),""),((Dar ciencia ao servi
dor),(Arquivar AFD),
""),((Arquivar AFD),(end),""),((Deferido?),(Realizar acertos
 financeiros - Param
eters: ParamPericulosidade - unbound,ParamInsalubridade - un
bound),""),((Realiza
r acertos financeiros - Parameters: ParamPericulosidade - un
bound,ParamInsalubri
dade - unbound),(Elaborar  boletim de servico),""),((Elabora
r  boletim de servic
o),(Publicar no boletim de servico),""),((Publicar no boleti
m de servico),(Arqui
```

```
var AFD),""),((Servidor tem direito a mais de um beneficio?)
,(Preencher requerim
ento),"SIM"),((Preencher requerimento),(Solicitar analise am
biental),"")]}]}
```

D.7.3 Instância do Tipo Algébrico quando as features Periculosidade e Insalubridade forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpPericulosidade",
 ptype = BasicProces
s, objects = [(start),(Elaborar memorando),(Analisar direito
 do servidor ao bene
ficio),(Servidor tem direito a mais de um beneficio?),(Solic
itar analise ambient
al),(Analisar ambiente),(Incluir laudo de condicoes ambienta
is),(Deferido?),(Dar
 ciencia ao servidor),(Arquivar AFD),(Realizar acertos finan
 ceiros - Parameters:
 ParamPericulosidade - unbound,ParamInsalubridade - unbound)
 ,(Elaborar  boletim
de servico),(Publicar no boletim de servico),(Preencher requ
erimento),(end)], tr
ansitions = [((start),(Elaborar memorando),""),((Elaborar me
morando),(Analisar d
ireito do servidor ao beneficio),""),((Analisar direito do s
ervidor ao beneficio
),(Servidor tem direito a mais de um beneficio?),""),((Servi
dor tem direito a ma
is de um beneficio?),(Solicitar analise ambiental),"NAO"),((
Solicitar analise am
biental),(Analisar ambiente),""),((Analisar ambiente),(Inclu
ir laudo de condicoe
s ambientais),""),((Incluir laudo de condicoes ambientais),(
Deferido?),""),((Def
erido?),(Dar ciencia ao servidor),""),((Dar ciencia ao servi
dor),(Arquivar AFD),
""),((Arquivar AFD),(end),""),((Deferido?),(Realizar acertos
 financeiros - Param
eters: ParamPericulosidade - unbound,ParamInsalubridade - un
bound),""),((Realiza
r acertos financeiros - Parameters: ParamPericulosidade - un
bound,ParamInsalubri
dade - unbound),(Elaborar  boletim de servico),""),((Elabora
r  boletim de servic
o),(Publicar no boletim de servico),""),((Publicar no boleti
m de servico),(Arqui
```

```
var AFD),""),((Servidor tem direito a mais de um beneficio?)
,(Preencher requerim
ento),"SIM"),((Preencher requerimento),(Solicitar analise am
biental),"")]},Busin
essProcess {pid = "bpPericulosidade", ptype = BasicProcess,
objects = [(start),(
Elaborar memorando),(Analisar direito do servidor ao benefic
io),(Servidor tem di
reito a mais de um beneficio?),(Solicitar analise ambiental)
,(Analisar ambiente)
,(Incluir laudo de condicoes ambientais),(Deferido?),(Dar ci
encia ao servidor),(
Arquivar AFD),(Realizar acertos financeiros - Parameters: Pa
ramPericulosidade -
unbound,ParamInsalubridade - unbound),(Elaborar  boletim de
servico),(Publicar n
o boletim de servico),(Preencher requerimento),(end)], trans
itions = [((start),(
Elaborar memorando),""),((Elaborar memorando),(Analisar dire
ito do servidor ao b
eneficio),""),((Analisar direito do servidor ao beneficio),(
Servidor tem direito
 a mais de um beneficio?),""),((Servidor tem direito a mais
 de um beneficio?),(S
olicitar analise ambiental),"NAO"),((Solicitar analise ambie
ntal),(Analisar ambi
ente),""),((Analisar ambiente),(Incluir laudo de condicoes a
mbientais),""),((Inc
luir laudo de condicoes ambientais),(Deferido?),""),((Deferi
do?),(Dar ciencia ao
 servidor),""),((Dar ciencia ao servidor),(Arquivar AFD),"")
 ,((Arquivar AFD),(en
d),""),((Deferido?),(Realizar acertos financeiros - Parame
rs: ParamPericulosid
ade - unbound,ParamInsalubridade - unbound),""),((Realizar a
certos financeiros -
 Parameters: ParamPericulosidade - unbound,ParamInsalubridad
 e - unbound),(Elabor
ar  boletim de servico),""),((Elaborar  boletim de servico),
(Publicar no boletim
 de servico),""),((Publicar no boletim de servico),(Arquivar
  AFD),""),((Servidor
 tem direito a mais de um beneficio?),(Preencher requeriment
 o),"SIM"),((Preenche
r requerimento),(Solicitar analise ambiental),"")]}]}
```

D.8 Caso VIII - Auxílio - Alimentação e Transporte - Produto Final - Instância dos Tipos Algébricos

D.8.1 Instância do Tipo Algébrico quando somente feature Alimentacao for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpAuxilioComum", p
type = BasicProcess,
 objects = [(start),(Registro),(O beneficio eh de desejo do
 servidor?),(Preenchi
mento de formulario),(Analisar os documentos e condicoes do
servidor pelo RH),(R
egistro do valor, conforme informado o servidor, de acordo c
om a tabela de codig
o do SIAPE),(Informar ao orgao por escrito que nao opta pelo
 beneficio),(Arquivo
),(end),(Recebe o beneficio por outro vinculo? ),(Cancelar o
 beneficio no outro
emprego)], transitions = [((start),(Registro),""),((O benefi
cio eh de desejo do
servidor?),(Preenchimento de formulario),"SIM"),((Preenchime
nto de formulario),(
Analisar os documentos e condicoes do servidor pelo RH),""),
((Analisar os docume
ntos e condicoes do servidor pelo RH),(Registro do valor, co
nforme informado o s
ervidor, de acordo com a tabela de codigo do SIAPE),""),((O
beneficio eh de dese
jo do servidor?),(Informar ao orgao por escrito que nao opta
 pelo beneficio),"NA
O"),((Informar ao orgao por escrito que nao opta pelo benefi
cio),(Arquivo),""),(
(Arquivo),(end),""),((Registro),(Recebe o beneficio por outr
o vinculo? ),""),((R
ecebe o beneficio por outro vinculo? ),(O beneficio eh de de
sejo do servidor?),"
NAO"),((Cancelar o beneficio no outro emprego),(O beneficio
eh de desejo do serv
idor?),""),((Recebe o beneficio por outro vinculo? ),(Cancel
ar o beneficio no ou
tro emprego),"SIM")]}]}
```

D.8.2 Instância do Tipo Algébrico quando somente feature Transporte for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpAuxilioComum", p
```

```
type = BasicProcess,
 objects = [(start),(Registro),(O beneficio eh de desejo do
 servidor?),(Preenchi
mento de formulario),(Analisar os documentos e condicoes do
servidor pelo RH),(R
egistro do valor, conforme informado o servidor, de acordo c
om a tabela de codig
o do SIAPE),(Informar ao orgao por escrito que nao opta pelo
 beneficio),(Arquivo
),(end)], transitions = [((start),(Registro),""),((Registro)
,(O beneficio eh de
desejo do servidor?),""),((O beneficio eh de desejo do servi
dor?),(Preenchimento
 de formulario),"SIM"),((Preenchimento de formulario),(Anali
 sar os documentos e
condicoes do servidor pelo RH),""),((Analisar os documentos
e condicoes do servi
dor pelo RH),(Registro do valor, conforme informado o servid
or, de acordo com a
tabela de codigo do SIAPE),""),((O beneficio eh de desejo do
 servidor?),(Informa
r ao orgao por escrito que nao opta pelo beneficio),"NAO"),(
(Informar ao orgao p
or escrito que nao opta pelo beneficio),(Arquivo),""),((Arqu
ivo),(end),"")]}]}
```

D.8.3 Instância do Tipo Algébrico quando as features Alimentacao e Transporte forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpAuxilioComum", p
type = BasicProcess,
 objects = [(start),(Registro),(O beneficio eh de desejo do
 servidor?),(Preenchi
mento de formulario),(Analisar os documentos e condicoes do
servidor pelo RH),(R
egistro do valor, conforme informado o servidor, de acordo c
om a tabela de codig
o do SIAPE),(Informar ao orgao por escrito que nao opta pelo
 beneficio),(Arquivo
),(end)], transitions = [((start),(Registro),""),((Registro)
,(O beneficio eh de
desejo do servidor?),""),((O beneficio eh de desejo do servi
dor?),(Preenchimento
 de formulario),"SIM"),((Preenchimento de formulario),(Anali
 sar os documentos e
condicoes do servidor pelo RH),""),((Analisar os documentos
```

```
e condicoes do servi
dor pelo RH),(Registro do valor, conforme informado o servid
or, de acordo com a
tabela de codigo do SIAPE),""),((O beneficio eh de desejo do
 servidor?),(Informa
r ao orgao por escrito que nao opta pelo beneficio),"NAO"),(
(Informar ao orgao p
or escrito que nao opta pelo beneficio),(Arquivo),""),((Arqu
ivo),(end),"")]},Bus
inessProcess {pid = "bpAuxilioComum", ptype = BasicProcess,
objects = [(start),(
Registro),(O beneficio eh de desejo do servidor?),(Preenchim
ento de formulario),
(Analisar os documentos e condicoes do servidor pelo RH),(Re
gistro do valor, con
forme informado o servidor, de acordo com a tabela de codigo
 do SIAPE),(Informar
 ao orgao por escrito que nao opta pelo beneficio),(Arquivo)
 ,(end),(Recebe o ben
eficio por outro vinculo? ),(Cancelar o beneficio no outro e
mprego)], transition
s = [((start),(Registro),""),((O beneficio eh de desejo do s
ervidor?),(Preenchim
ento de formulario),"SIM"),((Preenchimento de formulario),(A
nalisar os documento
s e condicoes do servidor pelo RH),""),((Analisar os documen
tos e condicoes do s
ervidor pelo RH),(Registro do valor, conforme informado o se
rvidor, de acordo co
m a tabela de codigo do SIAPE),""),((O beneficio eh de desej
o do servidor?),(Inf
ormar ao orgao por escrito que nao opta pelo beneficio),"NAO
"),((Informar ao org
ao por escrito que nao opta pelo beneficio),(Arquivo),""),((
Arquivo),(end),""),(
(Registro),(Recebe o beneficio por outro vinculo? ),""),((Re
cebe o beneficio por
 outro vinculo? ),(O beneficio eh de desejo do servidor?),"N
 AO"),((Cancelar o be
neficio no outro emprego),(O beneficio eh de desejo do servi
dor?),""),((Recebe o
 beneficio por outro vinculo? ),(Cancelar o beneficio no out
 ro emprego),"SIM")]}
]}
```

D.9 Caso IX - Auxílio Filiação - Auxílio Pré-Escolar e Auxílio Natalidade - Produto Final - Instância dos Tipos Algébricos

D.9.1 Instância do Tipo Algébrico quando somente feature Auxilio Pre-Escolar for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpAuxilioNatalidad

e", ptype = BasicPro
cess, objects = [(start),(Preencher requerimento),(Analisar
informacoes dos docu
mentos),(end),(Idade maxima alcancada?),(Possui o beneficio?
),(Excluir beneficio
),(Realizar acertos financeiros)], transitions = [((start),(
Preencher requerimen
to),""),((Preencher requerimento),(Analisar informacoes dos
documentos),""),((An
alisar informacoes dos documentos),(Idade maxima alcancada?)
,""),((Possui o bene
ficio?),(end),"NAO"),((Excluir beneficio),(end),""),((Realiz
ar acertos financeir
os),(end),""),((Idade maxima alcancada?),(Possui o beneficio
?),"SIM"),((Idade ma
xima alcancada?),(Realizar acertos financeiros),"NAO"),((Pos
sui o beneficio?),(E
xcluir beneficio),"SIM")]}]}
```

D.9.2 Instância do Tipo Algébrico quando somente feature Auxilio Natalidade for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpAuxilioNatalidad
e", ptype = BasicPro
cess, objects = [(start),(Preencher requerimento),(Analisar
informacoes dos docu
mentos),(Realizar acertos financeiros),(end)], transitions =
 [((start),(Preenche
r requerimento),""),((Preencher requerimento),(Analisar info
rmacoes dos document
os),""),((Analisar informacoes dos documentos),(Realizar ace
rtos financeiros),""
),((Realizar acertos financeiros),(end),"")]}]}
```

D.9.3 Instância do Tipo Algébrico quando as features Auxilio Pre-Escolar e Auxilio Natalidade forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpAuxilioNatalidad
e", ptype = BasicPro
cess, objects = [(start),(Preencher requerimento),(Analisar
informacoes dos docu
mentos),(Realizar acertos financeiros),(end)], transitions =
 [((start),(Preenche
r requerimento),""),((Preencher requerimento),(Analisar info
rmacoes dos document
os),""),((Analisar informacoes dos documentos),(Realizar ace
rtos financeiros),""
),((Realizar acertos financeiros),(end),"")]},BusinessProces
s {pid = "bpAuxilioN
atalidade", ptype = BasicProcess, objects = [(start),(Preenc
her requerimento),(A
nalisar informacoes dos documentos),(end),(Idade maxima alca
ncada?),(Possui o be
neficio?),(Excluir beneficio),(Realizar acertos financeiros)
], transitions = [((
start),(Preencher requerimento),""),((Preencher requerimento
),(Analisar informac
oes dos documentos),""),((Analisar informacoes dos documento
s),(Idade maxima alc
ancada?),""),((Possui o beneficio?),(end),"NAO"),((Excluir b
eneficio),(end),""),
((Realizar acertos financeiros),(end),""),((Idade maxima alc
ancada?),(Possui o b
eneficio?),"SIM"),((Idade maxima alcancada?),(Realizar acert
os financeiros),"NAO
"),((Possui o beneficio?),(Excluir beneficio),"SIM")]}]}
```

D.10 Caso X - Férias - Marcar e Férias e Remarcar Férias - Produto Final - Instância dos Tipos Algébricos

D.10.1 Instância do Tipo Algébrico quando somente feature Marcar Ferias for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpFeriasComum", pt
ype = BasicProcess,
objects = [(start),(Cadastrar),(Realizar acertos financeiros
```

```
),(Arquivar AFD),(Da
r ciencia ao servidor),(end),(Solicitar escala de ferias),(+
),(Solicitar previsa
o de ferias),(Preencher requerimento),(Elaborar escala de fe
rias),(Analisar esca
la de ferias),(Autorizar escala de ferias),(Escala autorizad
a?),(Solicitar alter
acao de ferias),(Parar contador do retorno da escala),(++),(
Elaborar documento p
ara assinatura),(Contar prazo de retorno da escala (15 dias)
),(Dentro do prazo?)
,(Assinar (Chefia imediata)),(Assinar (Servidor))], transiti
ons = [((Cadastrar),
(Realizar acertos financeiros),""),((Realizar acertos financ
eiros),(Arquivar AFD
),""),((Arquivar AFD),(Dar ciencia ao servidor),""),((Dar ci
encia ao servidor),(
end),""),((start),(Solicitar escala de ferias),""),((start),
(Preencher requerime
nto),""),((Solicitar escala de ferias),(+),""),((+),(Solicit
ar previsao de feria
s),""),((Solicitar previsao de ferias),(Preencher requerimen
to),""),((Preencher
requerimento),(Elaborar escala de ferias),""),((Elaborar esc
ala de ferias),(Anal
isar escala de ferias),""),((Analisar escala de ferias),(Aut
orizar escala de fer
ias),""),((Autorizar escala de ferias),(Escala autorizada?),
""),((Escala autoriz
ada?),(Solicitar alteracao de ferias),"NAO"),((Solicitar alt
eracao de ferias),(P
reencher requerimento),""),((Escala autorizada?),(Parar cont
ador do retorno da e
scala),"SIM"),((Parar contador do retorno da escala),(++),""
),((++),(Elaborar do
cumento para assinatura),""),((+),(Contar prazo de retorno d
a escala (15 dias)),
""),((Contar prazo de retorno da escala (15 dias)),(Dentro d
o prazo?),""),((Dent
ro do prazo?),(++),"SIM"),((Elaborar documento para assinatu
ra),(Assinar (Chefia
 imediata)),""),((Assinar (Servidor)),(Cadastrar),""),((Assi
 nar (Chefia imediata
)),(Assinar (Servidor)),"")]}]}
```

D.10.2 Instância do Tipo Algébrico quando somente feature Remarcar Ferias for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpFeriasComum", pt
ype = BasicProcess,
objects = [(start),(Cadastrar),(Realizar acertos financeiros
),(Arquivar AFD),(Da
r ciencia ao servidor),(end),(Ha documento para cancelamento
  de ferias?),(Elabo
rar documento para cancelamento),(Assinar (Chefia imediata))
,(Assinar (Servidor)
)], transitions = [((Ha documento para cancelamento  de feri
as?),(Cadastrar),"SI
M"),((Cadastrar),(Realizar acertos financeiros),""),((Realiz
ar acertos financeir
os),(Arquivar AFD),""),((Arquivar AFD),(Dar ciencia ao servi
dor),""),((Dar cienc
ia ao servidor),(end),""),((start),(Ha documento para cancel
amento  de ferias?),
""),((Ha documento para cancelamento  de ferias?),(Elaborar
documento para cance
lamento),"NAO"),((Elaborar documento para cancelamento),(Ass
inar (Chefia imediat
a)),""),((Assinar (Servidor)),(Ha documento para cancelament
o  de ferias?),""),(
(Assinar (Chefia imediata)),(Assinar (Servidor)),"")]}]}
```

D.10.3 Instância do Tipo Algébrico quando as features Marcar Ferias e Remarcar Ferias forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpFeriasComum", pt
ype = BasicProcess,
objects = [(start),(Cadastrar),(Realizar acertos financeiros
),(Arquivar AFD),(Da
r ciencia ao servidor),(end),(Solicitar escala de ferias),(+
),(Solicitar previsa
o de ferias),(Preencher requerimento),(Elaborar escala de fe
rias),(Analisar esca
la de ferias),(Autorizar escala de ferias),(Escala autorizad
a?),(Solicitar alter
acao de ferias),(Parar contador do retorno da escala),(++),(
Elaborar documento p
ara assinatura),(Contar prazo de retorno da escala (15 dias)
),(Dentro do prazo?)
,(Assinar (Chefia imediata)),(Assinar (Servidor))], transiti
ons = [((Cadastrar),
```

```
(Realizar acertos financeiros),""),((Realizar acertos financ
eiros),(Arquivar AFD
),""),((Arquivar AFD),(Dar ciencia ao servidor),""),((Dar ci
encia ao servidor),(
end),""),((start),(Solicitar escala de ferias),""),((start),
(Preencher requerime
nto),""),((Solicitar escala de ferias),(+),""),((+),(Solicit
ar previsao de feria
s),""),((Solicitar previsao de ferias),(Preencher requerimen
to),""),((Preencher
requerimento),(Elaborar escala de ferias),""),((Elaborar esc
ala de ferias),(Anal
isar escala de ferias),""),((Analisar escala de ferias),(Aut
orizar escala de fer
ias),""),((Autorizar escala de ferias),(Escala autorizada?),
""),((Escala autoriz
ada?),(Solicitar alteracao de ferias),"NAO"),((Solicitar alt
eracao de ferias),(P
reencher requerimento),""),((Escala autorizada?),(Parar cont
ador do retorno da e
scala),"SIM"),((Parar contador do retorno da escala),(++),""
),((++),(Elaborar do
cumento para assinatura),""),((+),(Contar prazo de retorno d
a escala (15 dias)),
""),((Contar prazo de retorno da escala (15 dias)),(Dentro d
o prazo?),""),((Dent
ro do prazo?),(++),"SIM"),((Elaborar documento para assinatu
ra),(Assinar (Chefia
 imediata)),""),((Assinar (Servidor)),(Cadastrar),""),((Assi
 nar (Chefia imediata
)),(Assinar (Servidor)),"")]},BusinessProcess {pid = "bpFeri
asComum", ptype = Ba
sicProcess, objects = [(start),(Cadastrar),(Realizar acertos
 financeiros),(Arqui
var AFD),(Dar ciencia ao servidor),(end),(Ha documento para
cancelamento  de fer
ias?),(Elaborar documento para cancelamento)], transitions =
 [((Ha documento par
a cancelamento  de ferias?),(Cadastrar),"SIM"),((Cadastrar),
(Realizar acertos fi
nanceiros),""),((Realizar acertos financeiros),(Arquivar AFD
),""),((Arquivar AFD
),(Dar ciencia ao servidor),""),((Dar ciencia ao servidor),(
end),""),((start),(H
a documento para cancelamento  de ferias?),""),((Ha document
o para cancelamento
```

```
 de ferias?),(Elaborar documento para cancelamento),"NAO"),(
 (Elaborar documento
para cancelamento),(Ha documento para cancelamento  de feria
s?),"")]}]}
```

D.11 Caso XI - Serviço - Serviço Extraordinário e Tempo de Serviço Quinquênio - Produto Final - Instância dos Tipos Algébricos

D.11.1 Instância do Tipo Algébrico quando somente feature Servico Extraordinario for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpServicoComum", p
type = BasicProcess,
 objects = [(start),(Preencher requerimento),(Analisar reque
 rimento),(Emitir par
ecer),(Deferido?),(Dar ciencia ao servidor),(Autorizar execu
cao  do servico),(Re
alizar acertos financeiros),(end),(Dar ciencia a chefia),(Da
r ciencia a chefia),
(Dar ciencia ao servidor),(Elaborar relatorio de servicos ex
traordinarios)], tra
nsitions = [((start),(Preencher requerimento),""),((Preenche
r requerimento),(Ana
lisar requerimento),""),((Analisar requerimento),(Emitir par
ecer),""),((Emitir p
arecer),(Deferido?),""),((Dar ciencia ao servidor),(end),"")
,((Deferido?),(Autor
izar execucao  do servico),"SIM"),((Realizar acertos finance
iros),(end),""),((De
ferido?),(Dar ciencia a chefia),"NAO"),((Dar ciencia a chefi
a),(Dar ciencia ao s
ervidor),""),((Autorizar execucao  do servico),(Dar ciencia
a chefia),""),((Elab
orar relatorio de servicos extraordinarios),(Realizar acerto
s financeiros),""),(
(Dar ciencia a chefia),(Dar ciencia ao servidor),""),((Dar c
iencia ao servidor),
(Elaborar relatorio de servicos extraordinarios),"")]}]}
```

D.11.2 Instância do Tipo Algébrico quando somente feature Tempo de Servico for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpServicoComum", p
```

```
type = BasicProcess,
 objects = [(start),(Preencher requerimento),(Analisar reque
 rimento),(Emitir par
ecer),(Deferido?),(Dar ciencia ao servidor),(Autorizar execu
cao  do servico),(Re
alizar acertos financeiros),(end),(Cadastro),(Elaborar bolet
im de  servico),(Pub
licar boletim de servico),(Arquivar AFD)], transitions = [((
start),(Preencher re
querimento),""),((Preencher requerimento),(Analisar requerim
ento),""),((Analisar
 requerimento),(Emitir parecer),""),((Emitir parecer),(Defer
 ido?),""),((Deferido
?),(Dar ciencia ao servidor),"NAO"),((Dar ciencia ao servido
r),(end),""),((Defer
ido?),(Autorizar execucao  do servico),"SIM"),((Autorizar ex
ecucao  do servico),
(Cadastro),""),((Publicar boletim de servico),(Realizar acer
tos financeiros),""
),((Cadastro),(Elaborar boletim de  servico),""),((Elaborar b
oletim de  servico),
(Publicar boletim de servico),""),((Realizar acertos finance
iros),(Arquivar AFD)
,""),((Arquivar AFD),(end),"")]}]}
```

D.11.3 Instância do Tipo Algébrico quando as features Servico Extraordinario e Tempo de Servico forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpServicoComum", p
type = BasicProcess,
 objects = [(start),(Preencher requerimento),(Analisar reque
 rimento),(Emitir par
ecer),(Deferido?),(Dar ciencia ao servidor),(Autorizar execu
cao  do servico),(Re
alizar acertos financeiros),(end),(Dar ciencia a chefia),(Da
r ciencia a chefia),
(Dar ciencia ao servidor),(Elaborar relatorio de servicos ex
traordinarios)], tra
nsitions = [((start),(Preencher requerimento),""),((Preenche
r requerimento),(Ana
lisar requerimento),""),((Analisar requerimento),(Emitir par
ecer),""),((Emitir p
arecer),(Deferido?),""),((Dar ciencia ao servidor),(end),""
),((Deferido?),(Autor
izar execucao  do servico),"SIM"),((Realizar acertos finance
```

```
iros),(end),""),((De
ferido?),(Dar ciencia a chefia),"NAO"),((Dar ciencia a chefi
a),(Dar ciencia ao s
ervidor),""),((Autorizar execucao  do servico),(Dar ciencia
a chefia),""),((Elab
orar relatorio de servicos extraordinarios),(Realizar acerto
s financeiros),""),(
(Dar ciencia a chefia),(Dar ciencia ao servidor),""),((Dar c
iencia ao servidor),
(Elaborar relatorio de servicos extraordinarios),"")]},Busin
essProcess {pid = "b
pServicoComum", ptype = BasicProcess, objects = [(start),(Pr
eencher requerimento
),(Analisar requerimento),(Emitir parecer),(Deferido?),(Dar
ciencia ao servidor)
,(Autorizar execucao  do servico),(Realizar acertos financei
ros),(end),(Cadastro
),(Elaborar boletim de  servico),(Publicar boletim de servic
o),(Arquivar AFD)],
transitions = [((start),(Preencher requerimento),""),((Preen
cher requerimento),(
Analisar requerimento),""),((Analisar requerimento),(Emitir
parecer),""),((Emiti
r parecer),(Deferido?),""),((Deferido?),(Dar ciencia ao serv
idor),"NAO"),((Dar c
iencia ao servidor),(end),""),((Deferido?),(Autorizar execuc
ao  do servico),"SIM
"),((Autorizar execucao  do servico),(Cadastro),""),((Public
ar boletim de servic
o),(Realizar acertos financeiros),""),((Cadastro),(Elaborar
boletim de  servico)
,""),((Elaborar boletim de  servico),(Publicar boletim de se
rvico),""),((Realiza
r acertos financeiros),(Arquivar AFD),""),((Arquivar AFD),(e
nd),"")]}]}
```

D.12 Caso XII - Indenização - Indenização Transporte e Transporte de Mobiliário e Bagagens - Produto Final - Instância dos Tipos Algébricos

D.12.1 Instância do Tipo Algébrico quando somente feature Transporte for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpIndenizacaoComum
", ptype = BasicProc
ess, objects = [(start),(Preencher requerimento),(Sinalizar
CPROD para criacao d
o numero do processo),(end),(Verificar se houve emissao de p
assagens),(Houve emi
ssao de passagem?),(Analisar processo),(Elaborar parecer),(P
arecer favoravel?),(
Cadastro),(Dar ciencia ao servidor),(Realizar acertos financ
eiros),(Cadastro (RH
)),(Dar ciencia ao servidor (RH)),(Emitir passagem)], transi
tions = [((start),(P
reencher requerimento),""),((Preencher requerimento),(Sinali
zar CPROD para criac
ao do numero do processo),""),((Sinalizar CPROD para criacao
 do numero do proces
so),(Verificar se houve emissao de passagens),""),((Dar cien
cia ao servidor),(en
d),""),((Dar ciencia ao servidor (RH)),(end),""),((Verificar
 se houve emissao de
 passagens),(Houve emissao de passagem?),""),((Houve emissao
  de passagem?),(Anal
isar processo),"NAO"),((Analisar processo),(Elaborar parecer
),""),((Elaborar par
ecer),(Parecer favoravel?),""),((Parecer favoravel?),(Cadast
ro),"NAO"),((Cadastr
o),(Dar ciencia ao servidor),""),((Houve emissao de passagem
?),(Realizar acertos
 financeiros),"SIM"),((Realizar acertos financeiros),(Cadast
 ro (RH)),""),((Cadas
tro (RH)),(Dar ciencia ao servidor (RH)),""),((Parecer favor
avel?),(Emitir passa
gem),"SIM"),((Emitir passagem),(Cadastro),"")]}]}
```

D.12.2 Instância do Tipo Algébrico quando somente feature Mobiliario e Bagagens for selecionada no PC

```
BPM {processes = [BusinessProcess {pid = "bpIndenizacaoComum
", ptype = BasicProc
ess, objects = [(start),(Preencher requerimento),(Sinalizar
CPROD para criacao d
o numero do processo),(end),(Analisar processo),(Elaborar pa
recer),(Parecer favo
ravel?),(Dar ciencia ao servidor),(Cadastro),(Organizar rmud
anca com o servidor)
,(Aguardar concretizacao da mudanca)], transitions = [((star
t),(Preencher requer
imento),""),((Preencher requerimento),(Sinalizar CPROD para
criacao do numero do
 processo),""),((Sinalizar CPROD para criacao do numero do p
 rocesso),(Analisar p
rocesso),""),((Cadastro),(end),""),((Analisar processo),(Ela
borar parecer),""),(
(Elaborar parecer),(Parecer favoravel?),""),((Parecer favora
vel?),(Dar ciencia a
o servidor),"NAO"),((Dar ciencia ao servidor),(Cadastro),"")
,((Parecer favoravel
?),(Organizar rmudanca com o servidor),"SIM"),((Organizar rm
udanca com o servido
r),(Aguardar concretizacao da mudanca),""),((Aguardar concre
tizacao da mudanca),
(Cadastro),"")]}]}
```

D.12.3 Instância do Tipo Algébrico quando as features Transporte e Mobiliario e Bagagens forem selecionadas no PC

```
BPM {processes = [BusinessProcess {pid = "bpIndenizacaoComum
", ptype = BasicProc
ess, objects = [(start),(Preencher requerimento),(Sinalizar
CPROD para criacao d
o numero do processo),(end),(Verificar se houve emissao de p
assagens),(Houve emi
ssao de passagem?),(Analisar processo),(Elaborar parecer),(P
arecer favoravel?),(
Cadastro),(Dar ciencia ao servidor),(Realizar acertos financ
eiros),(Cadastro (RH
)),(Dar ciencia ao servidor (RH)),(Emitir passagem)], transi
tions = [((start),(P
reencher requerimento),""),((Preencher requerimento),(Sinali
zar CPROD para criac
```

```
ao do numero do processo),""),((Sinalizar CPROD para criacao
 do numero do proces
so),(Verificar se houve emissao de passagens),""),((Dar cien
cia ao servidor),(en
d),""),((Dar ciencia ao servidor (RH)),(end),""),((Verificar
 se houve emissao de
 passagens),(Houve emissao de passagem?),""),((Houve emissao
  de passagem?),(Anal
isar processo),"NAO"),((Analisar processo),(Elaborar parecer
),""),((Elaborar par
ecer),(Parecer favoravel?),""),((Parecer favoravel?),(Cadast
ro),"NAO"),((Cadastr
o),(Dar ciencia ao servidor),""),((Houve emissao de passagem
?),(Realizar acertos
 financeiros),"SIM"),((Realizar acertos financeiros),(Cadast
 ro (RH)),""),((Cadas
tro (RH)),(Dar ciencia ao servidor (RH)),""),((Parecer favor
avel?),(Emitir passa
gem),"SIM"),((Emitir passagem),(Cadastro),"")]},BusinessProc
ess {pid = "bpIndeni
zacaoComum", ptype = BasicProcess, objects = [(start),(Preen
cher requerimento),(
Sinalizar CPROD para criacao do numero do processo),(end),(A
nalisar processo),(E
laborar parecer),(Parecer favoravel?),(Dar ciencia ao servid
or),(Cadastro),(Orga
nizar rmudanca com o servidor),(Aguardar concretizacao da mu
danca)], transitions
 = [((start),(Preencher requerimento),""),((Preencher requer
 imento),(Sinalizar C
PROD para criacao do numero do processo),""),((Sinalizar CPR
OD para criacao do n
umero do processo),(Analisar processo),""),((Cadastro),(end)
,""),((Analisar proc
esso),(Elaborar parecer),""),((Elaborar parecer),(Parecer fa
voravel?),""),((Pare
cer favoravel?),(Dar ciencia ao servidor),"NAO"),((Dar cienc
ia ao servidor),(Cad
astro),""),((Parecer favoravel?),(Organizar rmudanca com o s
ervidor),"SIM"),((Or
ganizar rmudanca com o servidor),(Aguardar concretizacao da
mudanca),""),((Aguar
dar concretizacao da mudanca),(Cadastro),"")]}]}
```