UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CARLOS FRANCISCO HABEKOST DOS SANTOS

Incrementando a codificação da Notação e Modelo de Processo de Negócio

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr[a]. Lucinéia Heloisa Thom

Porto Alegre
2016

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Santos, Carlos Francisco Habekost dos

Incrementando a codificação da Notação e Modelo de Processo de Negócio / Carlos Francisco Habekost dos Santos. – 2016.

67 f.: il.

Orientador: Lucinéia Heloisa Thom.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR–RS, 2016.

1. Meta-algoritmos. 2. BPMN. 3. XML. 4. Elementos notacionais. I. Thom, Lucinéia Heloisa. II.Incrementando a codificação da Notação e Modelo de Processo de Negócio.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Luis da Cunha Lamb
Coordenador do PPGC: Prof. Luigi Carro
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“Queira!
Basta ser sincero e desejar profundo
Você será capaz de sacudir o mundo!”
— Raul Seixas

AGRADECIMENTOS

Agradeço inicialmente ao meus pais, João e Sueli, por todo apoio e força dados durante esse tempo. Com certeza, sem eles, não teria chegado onde estou chegando agora. Em especial também a minha tia Sueli por todo o apoio dado.

Gostaria de agradecer também a minha orientadora, profª Lucinéia Thom por todo auxílio e ensinamentos dados. Posso afirmar que evolui muito desde a minha chegada ao mestrado.

Ao professor Marcelo Fantinato, pelas contribuições dadas nos artigos escritos, sempre com suas valiosas observações. Agradeço também a profª Erika Cota, pela contribuição na etapa de validação do trabalho.

Aos meus amigos Jonatan, Didi, Fabiano e Lázaro pelos momentos engraçados e descontraídos que passamos. Nada melhor do que reunir o nosso grupo.

Aos colegas do lab 213: Renato, Gabriel, Guilherme, Diego, Simone, Lorayne e os demais que já passaram pelo laboratório, pelas conversas e experiências vividas.

E por fim agradeço todos aqueles que direta ou indiretamente estiveram apoiando essa caminha a qual estou concluindo agora.

RESUMO

O gerenciamento de processos de negócio (BPM) objetiva a melhor documentação e padronização dos processos de negócio, além do aumento da eficiência e qualidade na execução dos processos. Um processo de negócio pode ser representado graficamente, através da Notação e Modelo de Processo de Negócio – BPMN (*Business Process Model and Notation*), a qual é um padrão da OMG (*Object Management Group*) para modelagem de processos. A BPMN provê um extenso conjunto de elementos de modelagem, tais como atividades, eventos e desvios, que possibilitam a representação de uma grande variedade de processos de negócio. É uma notação com alto poder de expressão que permite capturar a relação lógica e temporal entre atividades, objeto de dados e recursos. Contudo, considerando a especificação da BPMN, existem limitações entre a definição dos elementos notacionais e sua respectiva codificação (no formato XML). Por exemplo, não está expresso no XML do elemento *fluxo de mensagem* alguma definição que permita conectar apenas elementos posicionados em *piscinas* distintas, tal como descrito na definição conceitual da notação. O principal enfoque deste trabalho é o desenvolvimento de uma lógica que permita expressar a definição textual para cada elemento notacional. Tal lógica foi chamada neste trabalho de *meta-algoritmo*. Para verificar a expressividade dos meta-algoritmos propostos, foi utilizado técnicas de Teste de Software, baseado em tabelas de decisão e grafos de fluxo de controle. Além disso, foi aplicado um questionário com o objetivo de verificar a aceitação perante os usuários. Como resultado, destaca-se que os meta-algoritmos tiveram aceitação de 73,33% dos participantes do questionário. A principal contribuição deste trabalho é prover uma lógica mais aderente às definições textuais dos elementos notacionais, além de evidenciar que os usuários podem ter um maior entendimento, conforme verificado no questionário aplicado.

Palavras-chave: Meta-algoritmos. BPMN. XML. Elementos notacionais.

Increasing the coding of the Business Process Model and Notation

ABSTRACT

Business process management (BPM) enables the documentation and standardization of business processes, increasing efficiency and quality in their execution. A business process can be represented graphically by Business Process Model and Notation – BPMN, which is an OMG (Object Management Group) standard for process modeling. BPMN provides an extensive set of modeling elements, such as activities, events and gateways, which enables the representation of a wide variety of business processes. It presents high expression power that captures both temporal and logical relations between activities, data object and resources. However, considering the BPMN specification, there exists a lack of conformity between the conceptual BPMN elements definition and their respective codification (in XML format). For example, it is not expressed in the XML Message Flow element any definition to connect only elements from different pools, as described in the conceptual element definition. The main goal of this work is to develop a logic that enables to express the rules described in the conceptual element definition, for each notational element. In this work, *meta-algorithm* is the term used to refer to this logic. Techniques of Software Testing, such as decision tables and graph coverage were used to check expressiveness of the proposed meta-algorithms. In order to verify the acceptance of users, a survey was applied, to verify the acceptance with users. As result, meta-algorithms had accepted by 73.33% participants. As main contribution this work provides a more adherent logic, compared to conceptual element definition, as well an evidence that users can have an increasing of understanding, like verified in the survey.

Keywords: Meta-algorithms, BPMN, XML, Notational elements.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BPM	Gerenciamento de Processos de Negócio – *Business Process Management*
BPMN	Notação para Modelagem de processos de Negócio – *Business Process Management Notation*
BPMS	Sistema de Gerenciamento de Processos de Negócio – *Business process Management System*
OMG	*Object Management Group*
WfMC	*Workflow Management Coalition*
W3C	*World Wide Web Consortium*
TI	Tecnologia da Informação
DP	Diagrama de Processo
CFG	Grafo de Fluxo de Controle

SUMÁRIO

1 INTRODUÇÃO

Independentemente do domínio de aplicação ao qual se referem, as atividades de uma organização estão relacionadas a processos, tais como a compra de um produto, a participação em uma licitação pública, a manufatura de um automóvel, ou tratamento de pacientes.

Um processo de negócio pode ser definido como uma ordem parcial de atividades executadas em uma organização, tendo como objetivo prestar um serviço a um cliente ou manufaturar um produto (DUMAS et al., 2013; WESKE, 2012; WFMC, 1999). A maneira pela qual processos de negócio são projetados e gerenciados influencia sua eficiência na execução e na satisfação do cliente, além de aumentar a competitividade de uma organização (PRIEGO-ROCHE et al., 2012).

Processos de negócio refletem o funcionamento de uma organização e são responsáveis por produzir serviços ou produtos que serão entregues a um cliente da organização. É através da execução de seus processos de negócio que as organizações criam os seus diferenciais competitivos (NASCIMENTO, 2014).

O Gerenciamento de Processos de Negócio (BPM) tem sua base centrada na gestão dos processos de negócio das organizações, possibilitando a sua representação e de suas atividades. Além disso, provê um conjunto de técnicas para a análise e melhoria contínua dos processos de negócio executados nas organizações (WESKE, 2012).

O ciclo de vida de BPM inclui as etapas de modelagem, configuração, execução e validação de um processo de negócio (WEBER; SADIQ; REICHERT, 2009; DUMAS; ROSA; MENDLING, 2012). Em particular, objetiva a melhor documentação e padronização de processos, além do aumento da eficiência e da qualidade na execução destes (THOM; REICHERT; IOCHPE, 2009).

Uma das principais etapas do ciclo de vida de BPM é o levantamento de requisitos, na qual se identifica os processos de negócio executados na organização e se realiza a respectiva modelagem dos processos identificados. Tal modelagem possibilita as organizações conhecer e entender os seus processos de negócio, a partir de uma perspectiva organizacional. Permite também a análise, modelagem, execução e monitoramento dos seus processos de negócio (PRIEGO-ROCHE et al., 2012). A representação destes processos de negócio em uma notação de modelagem pode ser realizada através da Notação e Modelo de Processo de Negócio (BPMN – *Business Process Modeling and Notation*) (OMG, 2013).

A BPMN oferece um amplo conjunto de elementos notacionais, incluindo atividades, eventos, desvios, entre outros. Existem outras notações que também possibilitam representar graficamente processos tais como *Event Process Chain* (EPC) (**??**), Diagrama de Atividades da Linguagem de Modelagem Unificada (UML) (WESKE, 2012) e Redes de Petri (AALST et al., 2003). Porém, a BPMN é um padrão para modelagem de processos, adotada pela *Object Management Group* (OMG) (OMG, 2013; AALST, 2013; THOM; REICHERT; IOCHPE, 2009), utilizada pela grande maioria dos fabricantes de ferramentas de modelagem de processo de negócio, sendo portanto a notação objeto de estudo deste trabalho.

1.1 Motivação e Justificativa

A BPMN dispõe de uma grande quantidade de elementos de notacionais. Para cada elemento notacional, a BPMN propõe uma definição textual, um desenho notacional e um código XML, tal como ilustrado na Figura 1.1 (mais informações sobre a BPMN estão disponíveis na sessão 2.2). O XML proposto na definição da notação está codificado em XSD (*XML Schema Definition*). Conhecido também como *XML Schema*, é uma recomendação oficial da *World Wide Web Consortium* (W3C) para validação, definição de estrutura, conteúdo e semântica de documentos XML [1].

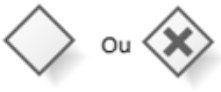
O desafio de pesquisa abordado nesta dissertação foca na limitação de semântica entre o XML de um elemento notacional e a respectiva definição textual, presentes na especificação da BPMN. Em (SANTOS; THOM; FANTINATO, 2015) é abordado esse desafio, cujo exemplo da Figura 1.1 mostra que não está definida na estrutura do XML (vide "Código XML") algum código que implemente qualquer definição sublinhada (vide "Definição Textual"). Uma das definições sublinhadas, por exemplo, consiste em realizar a avaliação das condições associadas a um desvio do tipo XOR (Desvio exclusivo). Após uma condição ser avaliada como verdadeira nenhuma outra será avaliada.

Além disso, outra definição da BPMN trata de definições sobre quais elementos podem ser conectados via fluxo de sequência (ex.: um *desvio* não pode ser conectado a um *evento de início*). Contudo, o XML dos elementos notacionais não apresentam alguma implementação que possibilite esse controle.

Analisando as *tags* da estrutura do XML, presentes na Figura 1.1, com base nas informações presentes na especificação da BPMN (OMG, 2013), têm-se:

[1] Maiores informações disponíveis em: http://www.w3.org/XML/Schema.html

Figura 1.1: Definição de elemento notacional pela especificação da BPMN.

Definição Textual	Each *token* arriving at any incoming Sequence Flows activates the gateway and is routed to exactly one of the outgoing Sequence Flows. In order to determine the outgoing Sequence Flows that receives the *token*, the conditions are evaluated in order. The first condition that evaluates to true determines the Sequence Flow the *token* is sent to. No more conditions are henceforth evaluated. If and only if none of the conditions evaluates to true, the *token* is passed on the default Sequence Flow. In case all conditions evaluate to false and a default flow has not been specified, an exception is thrown.
Representação Gráfica	 Ou

Código XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <exclusiveGateway p1:any_Attr="anySimpleType" default="ID1" gatewayDirection="Unspecified"
3  name="name1" id="ID1" xmlns:p1="otherNS" xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
4    <documentation id="ID2" textFormat="text/plain">text</documentation>
5    <extensionElements />
6    <auditing p1:any_Attr="anySimpleType" id="ID5" />
7    <monitoring p1:any_Attr="anySimpleType" id="ID6" />
8    <categoryValueRef>qname1</categoryValueRef>
9    <incoming>qname1</incoming>
10   <outgoing>qname3</outgoing>
11 </exclusiveGateway>
```

Fonte: Adaptado de OMG (2013).

- Linha 4 - `documentation`: Descrição textual do elemento;
- Linha 5 - `extensionElements`: Espaço para extensões do elemento notacional. Na prática, cada ferramenta (BPMS) a utiliza para personalizar o XML.
- Linha 6 - `auditing`: Permite a definição de atributos referentes a auditoria. A criação de e semântica destes atributos fica a cargo do usuário que desenvolve a ferramenta.
- Linha 7 - `monitoring`: Permite a definição de atributos referentes a monitoramento. A criação de semântica dos atributos fica a cargo do usuário que desenvolve a ferramenta.
- Linha 8 - `categoryValueRef`: Referência a uma categoria. Estas são utilizadas para categorizar os elementos de um processo de negócio.
- Linha 9 - `incoming`: Identifica o *fluxo de sequência* de entrada.
- Linha 10 - `outgoing`: Identifica o *fluxo de sequência* de saída.

Os atributos analisados mostram que o atual XML foca em questões estruturais de um modelo de processo, conforme as *tags* `documentation`, `auditing`, `monitoring`. Não há *tags* que tratam de algum tipo de referência a semântica do elemento, como por

exemplo a definição: se e somente se, nenhuma condição for avaliada como verdadeira, o *token* é passado para o fluxo de sequência *default*. Isso abre precedente para que cada fabricante implemente da sua forma os elementos notacionais, o que pode levar a nem sempre implementar as regras definidas pela BPMN. Outros exemplos em que a definição não está sendo aplicada no XML:

- Não inserir *fluxo de sequência* após o *evento de fim*.
- Uma *atividade de loop* deve continuar sua execução, enquanto a condição de *loop* for avaliada como verdadeira.
- Em um *desvio paralelo (AND)* de junção, a execução deve aguardar por todos os *fluxos de sequência* de entrada, antes de lançar o *fluxo de sequência* de saída.

O presente trabalho visa o desenvolvimento de meta-algoritmos com a lógica comportamental do elemento notacional proposta na especificação da BPMN.

Por meta-algoritmo, neste trabalho, entende-se como o algoritmo que dispõe da lógica de implementação do elemento notacional, cuja implementação é realizada de forma textual, sem o uso de linguagem de programação especifica.

Todo o trabalho realizado foi baseado na especificação da BPMN versão 2.0.2 de 2013 (OMG, 2013). Sendo que toda a vez que for feita a referência à definição textual dos elementos notacionais, está sendo feita referência a este documento.

Para possibilitar a geração dos meta-algoritmos, foi realizada uma análise das definições textuais, considerando as regras definidas em (KARAKOSTAS, 1990), que analisa a estrutura da sentença para gerar o código.

Os meta-algorítimos obtidos foram verificados utilizando as técnicas de Teste de Software: a Verificação Funcional, utilizando Tabela de Decisão e Grafos de Fluxo de Controle, e teste de aceitação, utilizando uma estrutura de questionário para verificar a aceitação do público-alvo.

Juntamente com a codificação, os meta-algoritmos serão uma fonte a mais de informação aos usuários que desejam desenvolver a sua própria ferramenta de modelagem de processos de negócio aderente à BPMN. Além disso, o trabalho proposto é um primeiro passo visando o desenvolvimento de um *parser* para verificação de modelos de processo de negócio.

1.2 Objetivos e Hipóteses

O objetivo geral desta dissertação é propor uma abordagem de incremento da codificação da BPMN. Para isso, propõe-se como hipóteses:

- É possível desenvolver uma lógica para implementação, na forma de meta-algoritmo, que seja aderente as definições textuais dos elementos notacionais da BPMN, definidas na especificação 2.0.2 de 2013 [2], e
- Tal codificação facilitará o entendimento do usuário em relação à essa definição.

Para atingir as hipóteses definidas, são descritos os seguintes objetivos específicos:

- Propor meta-algoritmos que apresentem maior aderência às definições textuais dos elementos notacionais da BPMN, considerando a especificação 2.0.2.
- Definir uma forma de verificar com o público alvo da especificação da notação, a aceitação dos meta-algoritmos propostos.
- Evidenciar que com o uso dos meta-algoritmos, o público poderá ter um maior entendimento em relação as definições textuais dos elementos notacionais da BPMN.

O público-alvo de uso dos meta-algoritmos são desenvolvedores e todos aqueles que buscam desenvolver algum sistema de modelagem de processos de negócio (BPMS). Busca-se com a realização deste trabalho, contribuir com a industria e comunidade acadêmica, oferecendo uma lógica inicial de implementação dos elementos notacionais da BPMN, para aqueles que desejam desenvolver um novo BPMS, partam já com um inicio de implementação.

1.3 Organização do Texto

Esta dissertação está dividida em seis capítulos, considerando esta introdução. Os demais capítulos estão dispostos da seguinte forma:

- o capítulo 2 contextualiza o tema de estudo, apresentando os principais conceitos relacionados à BPM, além de abordar os trabalhos relacionados.
- o capítulo 3 expõe a abordagem dos meta-algoritmos, expondo a metodologia de criação, definição, e conceitos inerentes ao seu entendimento;
- o capítulo 4 apresenta a validação dos meta-algoritmos, a partir de testes de *software*

[2] Disponível em: http://www.omg.org/spec/BPMN/2.0.2/

e diretamente com o público-alvo, apresentando também uma análise crítica dos testes realizados;

- o capítulo 5 apresenta a conclusão deste trabalho, com destaque as principais contribuições, limitações e possibilidades de extensões do mesmo.

2 FUNDAMENTOS DE GERENCIAMENTO DE PROCESSOS DE NEGÓCIO E TRABALHOS RELACIONADOS

Este capítulo objetiva introduzir os conceitos utilizados na pesquisa. É apresentado o ciclo de vida, a notação BPMN e outros conceitos necessários para o entendimento do trabalho. São discutidos também os trabalhos relacionados à pesquisa desenvolvida. Por fim, é feita uma discussão do que foi apresentado.

Segundo Dumas et al. (2013), o Gerenciamento de Processos de Negócio – BPM, objetiva organizar a realização do trabalho nas organizações visando assegurar resultados consistentes e aproveitar oportunidades de melhoria. BPM se preocupa com a gestão de eventos, atividades, decisões e invocações de processos de negócio, buscando agregar valor às organizações e aos clientes.

De acordo com (La Rosa et al., 2011), a aplicação de BPM pode possibilitar a redução de custos, além de contribuir para o gerenciamento de mudanças na organização. Com base em BPM, é possível atingir mais facilmente o alinhamento estratégico entre as áreas de negócio e TI, gerenciando as soluções tecnológicas a partir de processos que agregam valor para as organizações. Deste modo, segundo Fantinato, Gimenes and Toledo (2010), BPM pode ser visto como uma vantagem competitiva para as organizações.

Um das essências de BPM são os modelos de processo de negócio, que podem capturar como o trabalho é realizado e como os objetivos são alcançados em uma organização (EID-SABBAGH et al., 2012).

Por fim, BPM é considerada uma extensão de sistemas e abordagens clássicas de gestão de processos, incluindo *workflows*. Contudo, BPM tem um alcance mais amplo, desde a automação e a análise de processos até a gestão de processos e o trabalho organizacional (AALST, 2011).

2.1 Ciclo de Vida de BPM

O ciclo de vida de BPM envolve um conjunto de fases organizadas que visam o gerenciamento de processos de negócio. Essas fases englobam a identificação dos processos de negócio, incluindo análise, redesenho, modelagem, monitoramento e controle, propostas por Dumas et al. (2013) e formando um ciclo. A Figura 2.1 mostra as fases deste ciclo:

- **Identificação de processo**: Nesta fase, um problema de negócio é proposto, os processos relevantes ao problema são identificados, delimitados e relacionados entre si. O resultado desta identificação é uma arquitetura de processo nova ou atualizada, que fornece uma visão geral dos processos em uma organização e suas relações.
- **Descoberta de processo** (também denominada como modelagem *as-is*): O corrente estado para cada processo relevante é documentado, na forma de um ou vários modelos de processos *as-is*. Um modelo de processo *as-is* retrata o estado corrente do processo de negócio na organização.
- **Análise de processo**: Questões associadas ao processo são identificadas, documentadas e sempre que possível quantificadas, utilizando medidas de performance. Como saída têm-se um conjunto de questões estruturadas, que são organizadas conforme o seu impacto, e às vezes em função do esforço requerido para realizar determinada ação. Essas questões são utilizadas para guiar a melhoria dos processos de negócio.
- **Redesenho de processo** (também denominada como melhoria de processo): Objetiva identificar mudanças no processo que possam auxiliar na solução de problemas levantados na fase de análise e permite a organização conhecer os objetivos de performance relacionados ao processo. Para isso, várias opções de mudanças são analisadas e comparadas considerando medidas de performance. Isso implica que as fases de redesenho e análise estão relacionadas: novas mudanças são propostas e são analisadas usando as técnicas de análise. Eventualmente, as opções mais promissoras são combinadas, levando a um processo de redesenho. Como saída desta fase, têm-se tipicamente um modelo *to-be*, utilizado como base para a próxima fase. Um modelo de processo *to-be* retrata a situação futura do processo de negócio, aplicando as mudanças propostas.
- **Implementação de processo**: As alterações necessárias para transformar um processo *as-is* para *to-be* são preparadas e realizadas. Abrange dois aspectos: gestão da mudança organizacional e automação de processos. Gestão da mudança organizacional refere ao conjunto de atividades requeridas para a mudança na forma de trabalho de todos os participantes envolvidos no processo. Automação de processo abrange o uso de tecnologia da informação como suporte ao processo *to-be*, como por exemplo, um BPMS.
- **Monitoramento e controle de processo**: Uma vez redesenhado, e executando o processo, dados relevantes são coletados e analisados para determinar se o processo

está sendo executado de acordo com as medidas de performance. Como dados relevantes têm-se gargalos, erros recorrentes, ou desvios de comportamento, que são identificados e medidas corretivas são aplicadas. Pontos de melhoria podem ser identificados, implicando que o ciclo seja repetido de forma contínua.

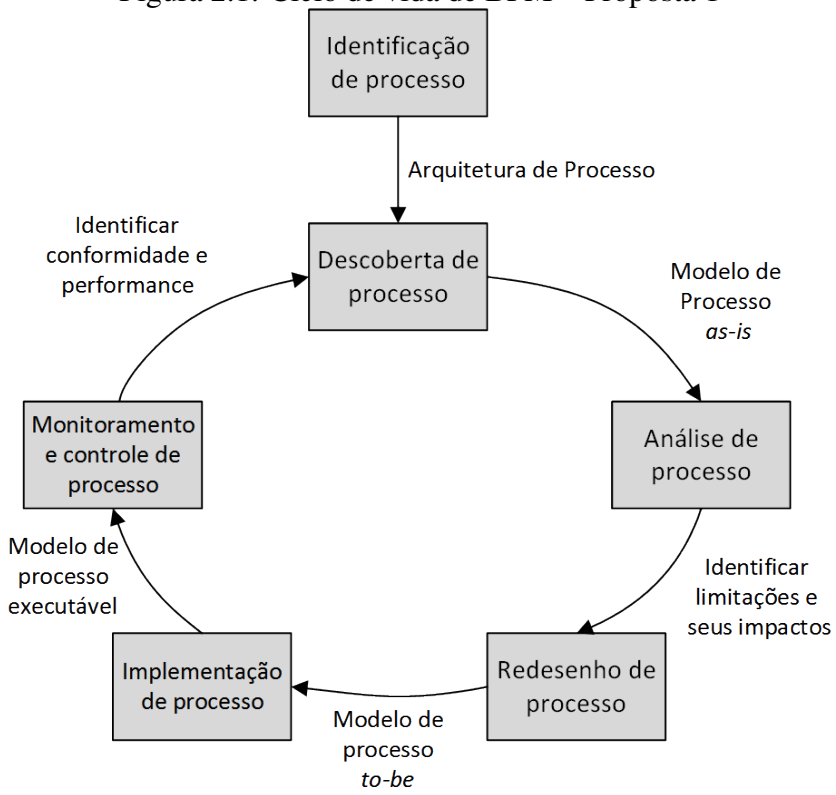
Há outras propostas visando definir e representar o ciclo de vida de BPM com algumas variações, apesar que são bastante similares. Em Weske (2012), por exemplo, é definida uma fase de projeto e análise, na qual os processos de uma organização são levantados, modelados e analisados. Na fase de configuração o modelo de processo de negócio é implementado, utilizando um sistema de apoio, podendo incluir diferentes plataformas e ferramentas. Na fase de execução, o processo é executado, podendo ocorrer a sua manutenção em execução. Por fim, na fase de avaliação, os dados gerados durante a execução são analisados para verificar se os objetivos organizacionais estão sendo atingidos de maneira satisfatória. Eventuais ineficiências nos processos podem ser identificados, possibilitando o seu aprimoramento.

A Figura 2.2 apresenta as fases da proposta de Weske (2012). Em suma, o ciclo de vida é similar ao proposto por Dumas et al. (2013), diferindo na forma em que as fases são apresentadas. Além destes, em Weber, Sadiq and Reichert (2009) e Aalst (2013) são propostos outros ciclos de vidas, mas com o mesmo propósito, como apresentar as fases de descoberta, implementação e execução e monitoramento de forma similar às descritas anteriormente.

O ciclo de BPM auxilia no entendimento do papel da tecnologia. Especialmente a tecnologia da informação (TI), é uma das chaves para melhoria do processo de negócio. Ao focar na sua automação, é necessário conhecê-lo melhor e trabalhar em pontos de melhoria, para que sua execução ocorra de acordo com as necessidades da organização (DUMAS et al., 2013).

Neste contexto, além de existir uma notação para possibilitar a modelagem do processo de negócio, é necessário que esta possibilite expressar os objetivos da organização. Neste aspecto a BPMN cumpre o seu papel. Além disso, O BPMS que irá implementar a notação precisa seguir as regras definidas na especificação. Assim, neste ponto se aplica o problema de pesquisa abordado neste trabalho, conforme abordado na seção 1.1, na página 12.

Figura 2.1: Ciclo de vida de BPM – Proposta 1



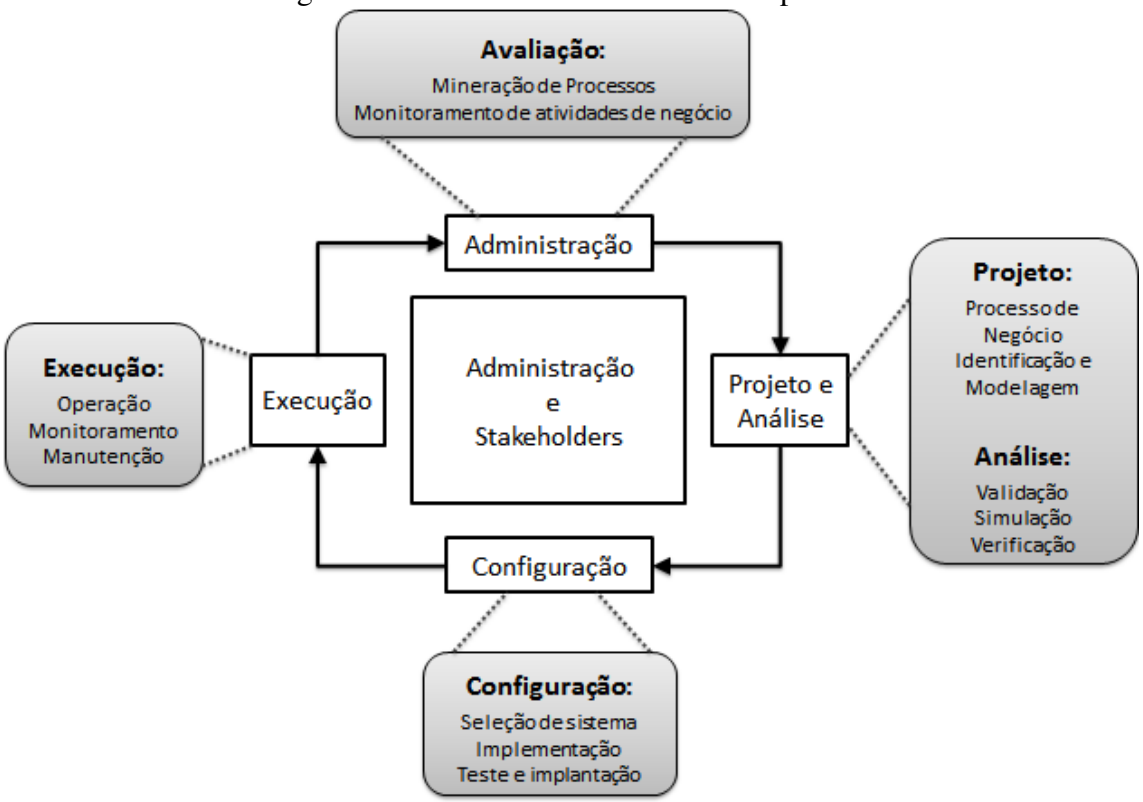
Fonte: Adaptado de Dumas et al. (2013).

2.2 Notação Para Modelagem de Processos de Negócio

A representação de modelos de processo de negócio tanto *"as-is"* como *"to-be"* ocorre através de uma notação formal que permita a representação de atividades, desvios e eventos. Para isso, têm-se a BPMN, a qual foi desenvolvida sob a coordenação da *Object Management Group* (OMG, 2013). O modelo de processo de negócio obtido a partir desta notação é chamado de *Diagrama de Processo* (DP). A BPMN considera um conjunto básico de elementos notacionais, conforme Figura 2.3:

- Objetos de fluxo: são os principais elementos utilizados para definir o comportamento de um processo de negócio. Há três objetos de fluxo: Eventos, Atividades e Desvios.
- Artefatos: são usados para prover informação adicional sobre o processo. A BPMN oferece dois artefatos padrões, mas fabricantes de ferramentas são livres para adicionar quantos Artefatos forem necessários. O conjunto atual de artefatos oferecidos inclui: Grupo e Anotação.
- Dados: representa os dados consumidos e/ou gerados pelo processo. É subdivido

Figura 2.2: Ciclo de vida de BPM – Proposta 2



Fonte: Adaptado de Weske (2012).

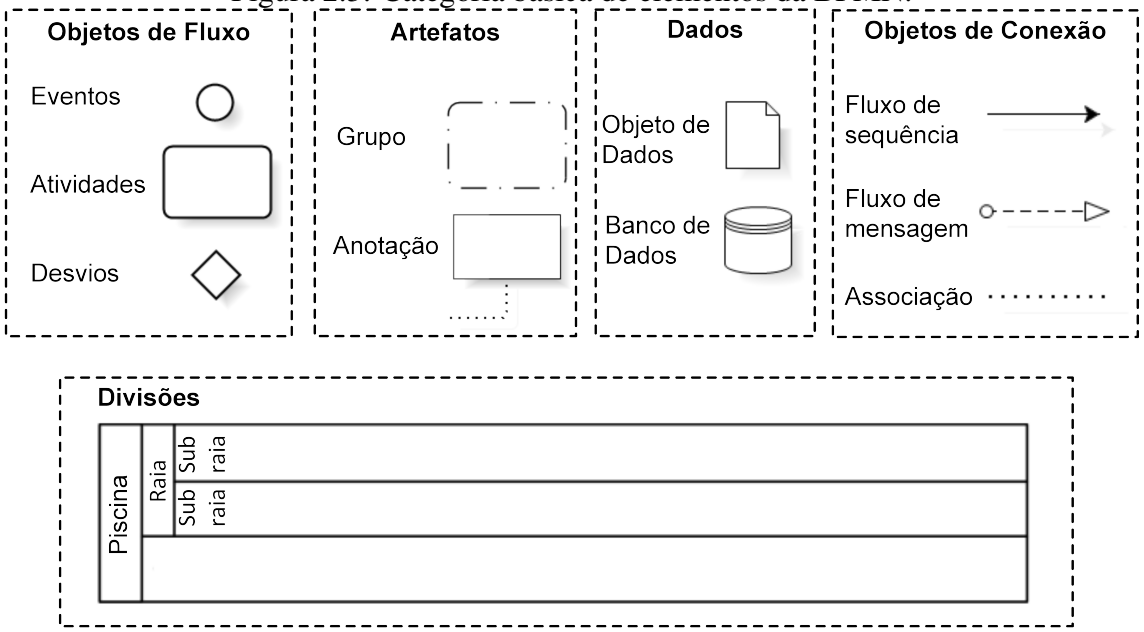
em: Objeto de dados e Banco de dados.

- Objetos de conexão: possui como função conectar dois objetos de fluxo, ou artefatos, ou dados. Os conectores que fazem parte desse grupo são: Fluxo de Sequência, Fluxo de Mensagem e Associação.
- Divisões: São elementos usados para agrupar os demais elementos notacionais. Fazem parte desse grupo: Piscina e Raia e sub-raia.

Como um exemplo de processo de negócio modelado em BPMN, é apresentado um processo de negócio para aprovação de um inquérito ministerial, adaptado de Dumas et al. (2013) (ver Figura 2.4):

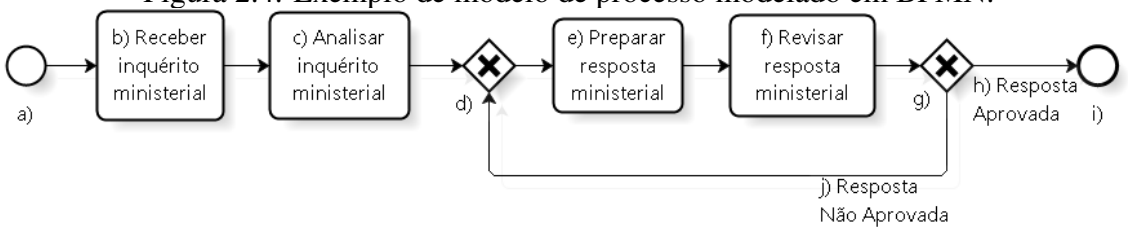
No escritório do Ministro da Fazenda, assim que um inquérito ministerial é recebido *(b)*, primeiramente, este é registrado no sistema. Em seguida, o pedido é analisado *(c)*, de modo que uma resposta ministerial possa ser preparada *(e)*. A finalização de uma resposta inclui a preparação da resposta por parte do chefe de gabinete e a revisão da resposta pelo registrador *(f)*. Se o registrador não aprovar a resposta *(j)*, esta deve novamente ser revista pelo chefe de gabinete. O processo termina assim que a resposta for aprovada *(h)*.

Figura 2.3: Categoria básica de elementos da BPMN.



Fonte: Adaptado de Weske (2012) e OMG (2013).

Figura 2.4: Exemplo de modelo de processo modelado em BPMN.



Fonte: Adaptado de Dumas et al. (2013).

Esse processo de negócio ilustra o grupo básico dos elementos notacionais da BPMN, utilizado pra modelagem de processos: Os Objetos de Fluxo. Conforme a OMG (2013), esses elementos podem ser definidos da seguinte forma:

- **Atividade:** é uma unidade de trabalho a ser executada no contexto de um processo de negócio, que pode ser atômica ou um subprocesso. Ela representa um ponto no fluxo de processo no qual o trabalho é realizado. Uma *atividade atômica* é uma tarefa, na qual representa uma ação que pode ser executada por uma pessoa ou sistema. No exemplo de processo da Figura 2.4, os elementos *b, c, e, f* são consideradas atividades, pois representam essa características. Uma subprocesso é quando uma atividade pode ser dividida em outras menores. No contexto da Figura 2.4 não há algum subprocesso definido. Porém, considerando hipoteticamente que durante a execução da atividade *c*, seja necessário a executar um conjunto de atividades, então seria possível afirmar que *c* se trata de um subprocesso.

- **Evento:** define algo que "acontece" no decorrer de um processo de negócio. Eventos afetam o fluxo e, normalmente, têm uma causa ou um impacto, além de requererem ou permitirem alguma reação. O termo "evento" é amplo o suficiente para abranger diversos aspectos de processo (ex.: o início de uma atividade, o fim de uma atividade, a mudança de estado em um documento, uma mensagem que chega). Na Figura 2.4, os elementos *a* e *i* representam eventos, do tipo início e fim, respectivamente. Eventos são divididos em três tipos:
  - *Evento de início:* indica o início de um processo;
  - *Evento de fim:* indica o fim de um processo;
  - *Evento intermediário:* indica algo que acontece em algum ponto do processo entre seu início e seu fim.
- **Desvio:** é usado para definir possíveis caminhos a serem tomados pelo fluxo de um processo, que pode ser de forma sequencial, paralela, ou por meio de uma seleção exclusiva. Na Figura 2.4, os elementos *d* e *g* representam os desvios do processo.

A partir da representação de um processo processo de negócio em um BPMS através de uma notação, neste caso a BPMN, é possível efetuar a *automação* de um processo de negócio.

Uma automação refere-se a intenção de automatizar qualquer parte concebível de trabalho processual que está contido dentro de um processo de negócio, podendo ser desde operações simples que fazem parte de uma única atividade, até a coordenação de todo processo automatizado (DUMAS et al., 2013; KARAGIANNIS, 1995).

O sistema de informação utilizado na etapa de automação de processos é o BPMS. Seu propósito consiste em coordenar um processo de negócio automatizado, de tal forma que todo o trabalho seja feito no momento adequado com os recursos corretos (DUMAS et al., 2013; MENDLING; REIJERS; AALST, 2010; REIJERS, 2006). Neste contexto, um processo de negócio automatizado, também conhecido como *workflow*, é referenciado como um processo automatizado que é, no todo ou em parte, controlado por um sistema de software. Tal sistema passa a informação a partir de um participante para outro, de acordo com dependências lógicas e temporais, conforme definido no modelo de processo (DUMAS et al., 2013; POELMANS; REIJERS; RECKER, 2013).

O mapeamento de um processo em BPMN para um processo automatizado requer uma quantidade de detalhes de execução que não é necessário em processos com o foco apenas na comunicação entre *stakeholders* ou análise (FABRA et al., 2012). Normal-

mente, modelos orientados ao negócio não são necessariamente precisos e podem conter ambiguidades. Por outro lado, os modelos de processo executáveis devem possuir especificações mais precisas, para que seja possível sua interpretação por um BPMS.

A relação do BPMS com este trabalho está na forma como a notação é utilizada na ferramenta. Para que um determinado modelo de processo de negócio, modelado em BPMN, possa ser executado em um BPMS, é necessário que a ferramenta implemente corretamente o comportamento de cada elemento notacional.

2.3 Trabalhos Relacionados

Ao longo da dissertação diversos trabalhos que contribuam para esse trabalho foram citados, porém essa seção visa o estudo de trabalhos específicos do que vem sendo trabalhados em áreas semelhante a estudada. Os estudos iniciais referentes a trabalhos relacionais iniciou-se com uma revisão sistemática referente a modelagem e execução de processos de negócio (SANTOS, 2014).

Uma revisão sistemática da literatura (ou simplesmente revisão sistemática) é um meio de identificar, avaliar e interpretar os trabalhos de pesquisa disponíveis e relevantes para uma questão de pesquisa específica, uma área temática, ou um fenômeno de interesse (KITCHENHAM; CHARTERS, 2007).

A metodologia seguida para a elaboração da revisão sistemática segue as fases definidas em Kitchenham and Charters (2007), na qual é definido um protocolo para execução, que consiste em um roteiro de como deve ser conduzida a avaliação dos trabalhos a serem estudados. A partir deste protocolo, obtêm-se um conjunto de trabalhos resultantes da revisão sistemática.

O trabalho de (STROPPI; CHIOTTI; VILLARREAL, 2011), tem como foco o desenvolvimento de um método para criação de extensões na notação BPMN. Essa extensão ocorre através da modificação do XML, adicionando novos recursos. O trabalho cita também que apesar da notação oferecer um mecanismo de extensão que permite desenvolver atributos e elementos adicionais ao conjunto inicial oferecido, existe falta de guias que considerem a extensão da notação, além da BPMN não oferecer de maneira satisfatória notação gráfica que possibilite representar extensões (por exemplo, elementos notacionais para representar aspectos de um domínio específico). Este trabalho foca no incremento da codificação através da extensão da codificação XML, adicionado recursos e dados no XML. Outras abordagens como em (TRAN et al., 2008) e (GROSSKOPF, 2007) também

abordam trabalhos de extensão da codificação.

Em (MENDLING; REIJERS; AALST, 2010) foi realizado um estudo referente a estruturação dos modelos e a probabilidade de erros de entendimento. Como resultado, foi proposto um conjunto de sete diretrizes sobre modelagem de processos de negócio. As diretrizes são:

- **G1:** Utilize o mínimo possível de elementos no modelo;
- **G2:** Minimize os caminhos de roteamento por elemento;
- **G3:** Utilize apenas um evento de início e um de fim;
- **G4:** Modele o mais estruturado possível;
- **G5:** Evite elementos de roteamento OR;
- **G6:** Utilize o estilo verbo-objeto para os rótulos de atividades;
- **G7:** Decomponha modelos com mais de 50 elementos;

As diretrizes discutidas por Mendling, Reijers and Aalst (2010) possibilitam um controle maior na modelagem e execução dos processos de negócio. Contudo, tais diretrizes são aplicas a nível de modelagem. Mesmo assim, as diretrizes podem servir de inspiração para serem aplicadas à codificação, como por exemplo usar o verbo-objeto para os rótulos de atividades, minimizar os caminhos de roteamento, dentre outros.

Em (DUMAS; ROSA; MENDLING, 2012) faz um estudo da complexidade dos modelos de processo de negócio. O trabalho realiza a comparação de um grupo de modelos de processo de negócio sem nenhum controle estrutural com outro grupo semanticamente equivalente, que está estruturado. É analisado se há diferenças de entendimento do usuário em relação aos dois grupos.

A abordagem de (SILVA et al., 2010) foca na verificação de *workflows* científicos, baseado na álgebra de processos e ferramentas de verificação de modelos. Esse trabalho é focado no uso de uma técnica externa para fazer a verificação de um modelo de processo, ao invés de focar em alguma abordagem na codificação.

Já em (KHERBOUCHE; AHMAD; BASSON, 2013) apresenta uma abordagem de detecção de erros estruturais em modelos de processo de negócio. Além disso, propõe auxiliar os usuários modeladores a evitar problemas como *deadlock* ou *livelock*. Para possibilitar essa detecção de erro, é utilizado um autômato não determinístico para representar o comportamento de um modelo de processo de negócio, na qual os nodos representam as atividades e os eventos, e as arestas representam as transições. São definidas fórmulas de transições, e partir destas é possível verificar se um modelo de processo de negócio

está livre de *deadlock* ou *livelock*, por exemplo. Neste caso, também é utilizado um meio externo para fazer a verificação de um modelo de processo de negócio, o autômato não determinístico, ao invés de uma abordagem utilizando os recurso da própria BPMN.

A partir das trabalhos apresentados, a Tabela 2.1 apresenta uma comparação dos trabalhos relacionados, utilizando como critério de comparação as soluções utilizadas pelo respectivo trabalho.

Tabela 2.1: Comparativo dos Trabalhos Relacionados

Autor	*EX*	*RM*	*TE*	*EC*
(STROPPI; CHIOTTI; VILLARREAL, 2011)	X			
(MENDLING; REIJERS; AALST, 2010)		X		
(SILVA et al., 2010)			X	
(TRAN et al., 2008)	X			
(KHERBOUCHE; AHMAD; BASSON, 2013)			X	
(DUMAS; ROSA; MENDLING, 2012)				X
(GROSSKOPF, 2007)	X			

Fonte: Os Autores

Os trabalhos foram classificados como:

- **EX** (*Extensão de XML*): Trabalhos que focam estender a codificação XML, adicionando novos recursos.
- **RM** (*Regras de Modelagem*): Trabalhos que focam em definição de regras para modelagem de processos de negócio.
- **TE** (*Técnicas Externas*): Trabalhos que focam em uso de alguma técnica externa para verificação de modelos, como por exemplo, álgebra de processos.
- **EC** (*Estudo de Compreensão*): Trabalhos que focam no estudo de compreensão estrutural de modelos de processos de negócio.

Dos trabalhos analisados, O foco está em modificar o XML, criando extensões (*EX*), ou definir regras para guiar a modelagem (*RM*), ou ainda utilizar técnicas externas (*TE*) ou estudar a compreensão de modelos de processos de negócio (*EC*). Contudo, não é abordado o foco desta dissertação, a limitação de expressividade do XML em relação as definições textuais.

2.4 Considerações finais

A BPMN oferece uma vasta quantidade de elementos notacionais, que possui um poder de expressão capaz de modelar diferentes processos de negócio. O ciclo de vida possui um conjunto de atividades que permite manter os processos de sempre atualizados com a realidade da organização.

Para que seja necessário para que o gerenciamento de processos de negócio ocorra de acordo com as necessidades da organização, é necessário existir algum meio para que os processos sejam modelados e executados. Neste contexto, o BPMS realiza esse papel no gerenciamento. Contudo necessário que a notação utilizada esteja implementada de acordo com a respectiva especificação, para que as regras comportamentais definidas para cada elemento notacional, sejam de fato seguidas.

No contexto dos trabalhos relacionados, existem abordagens que tentam tratar de problemas estruturais utilizando meios externos, como álgebra de processos em Silva et al. (2010) ou autômato não determinístico em Kherbouche, Ahmad and Basson (2013). Essas abordagens cumprem com o seu requisito, que é fazer a verificação de modelos de processo de negócio, porém para o usuário que executa essas tarefas, utilizar uma abordagem extra pode tornar o trabalho de verificação mais complexo. Sendo assim, os meta-algoritmos seriam um caminho para tentar minimizar o uso de meios externos para ter um modelo de processo de negócio completo.

Já nas abordagens de (STROPPI; CHIOTTI; VILLARREAL, 2011), (TRAN et al., 2008) e (GROSSKOPF, 2007) focam na extensão da codificação da BPMN, o XML, modificando-o para adicionar os novos recursos. Tais modificações não focam na questão da aderência com a definição textual dos elementos notacionais.

Com base nas abordagens apresentadas, este trabalho provê um caminho, através dos meta-algoritmos, para que a implementação seja aderente à definição dos elementos notacionais, sem utilizar alguma abordagem extra, ou ter que adicionar funcionalidades que inicialmente não estão definidas na especificação da BPMN.

3 DESENVOLVENDO META-ALGORITMOS PARA A NOTAÇÃO E MODELO DE PROCESSOS DE NEGÓCIO

Este capítulo descreve as etapas seguidas neste trabalho para o desenvolvimento dos meta-algoritmos a partir da definição dos elementos notacionais da BPMN. Conforme colocado na seção 1.1, a atual codificação oferecida pela especificação não está completamente aderente às regras definidas para os elementos notacionais. A partir disso, é apresentado no presente trabalho, uma abordagem para se realizar o incremento da codificação, que compreende quatro etapas importantes:

1. Definição do grupo de elementos notacionais da BPMN a serem estudados.
2. Utilização de uma forma de obtenção de fragmento de código, dada a definição de um elemento notacional.
3. Geração dos meta-algoritmos, a partir dos fragmentos obtidos.
4. Verificação e validação dos meta-algoritmos gerados.

A definição do grupo de elementos a serem estudados se faz necessária, pois a partir da especificação destes elementos são desenvolvidos os meta-algoritmos.

A partir deste grupo, são estudadas formas possíveis para que, com base na definição textual de cada elemento notacional, seja possível identificar o que deve ser implementado no meta-algoritmo.

A partir do momento que se sabe como identificar o que deve ser extraído da definição textual, os meta-algoritmos são desenvolvidos, criando uma estrutura lógica tendo como base essa definição.

Por fim, é realizada a verificação e validação, que permita identificar se os meta-algoritmos realmente estão aderentes à definição textual, além de identificar a aceitação do usuário. A validação está descrita no Capítulo 4.

3.1 Elementos Notacionais Selecionados

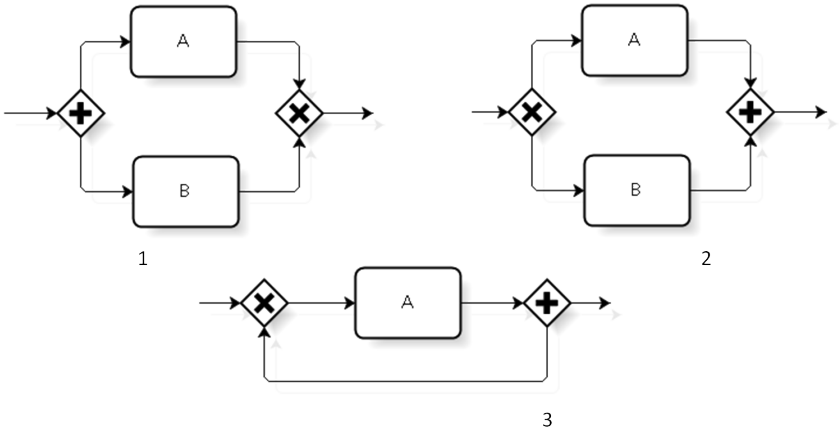
A primeira etapa do incremento foi definir o conjunto de elementos notacionais para os quais, meta-algoritmos seriam desenvolvidos. Como critério de seleção, escolheu-se os elementos que mais apresentam estudos na literatura sobre problemas estruturais. Problemas estruturais são problemas que podem ocorrer durante a execução de um processo de negócio.

Segundo Dumas et al. (2013), alguns dos problemas relatados são:

- *Deadlock*: quando não é possível continuar a execução de um processo de negócio, devido à um travamento.
- *Livelock*: acontece quando os elementos de um processo de negócio estão dispostos de forma a construir um *loop* infinito e o processo não atinge qualquer evento de fim.
- *Falta de sincronismo*: quando fluxos paralelos executam um desvio que divide o fluxo, mas não convergem para outro desvio que uni o fluxo.

Normalmente esses problemas ocorrem com desvios, cujo elementos selecionados são: *Desvio Exclusivo* (XOR), *Desvio Inclusivo* (OR) e *desvio paralelo* (AND). A Figura 3.1 traz exemplos de fragmentos de processos que incluem os problemas destacados.

Figura 3.1: Exemplos de problemas de modelagem.



Fonte: dos Autores.

Os exemplos apresentados mostram três trechos de modelagem com problemas. Em 1, têm-se um problema de *falta de sincronismo*, pois a modelagem apresenta um AND-*split* (losango com uma cruz central) e um XOR-*join* (losango com um X central). Na prática, ao executar o AND-*split*, a execução do processo é dividia, criando dois *tokens*, e *A* e *B* são executadas em paralelo; porém, ao primeiro que chegar no XOR-*join*, o fluxo continuará e não esperará o outro *token* chegar.

Em 2, têm-se o problema de *deadlock*. A modelagem apresenta um XOR-split e um AND-join. Na execução, partirá do XOR apenas um *token*, mas como há dois fluxos de sequência na entrada do AND, então este esperará a chegada dos dois *tokens*, causando o travamento da execução.

Já em 3, toda a vez que a execução chegar no AND-*join*, serão criados dois *tokens*, fazendo com que se tenha um *loop* infinito na execução do processo.

Apesar do critério de escolha dos elementos foi relacionado à problemas de modelagem, esta dissertação foca apenas em desenvolver meta-algoritmos mais expressivos em relação à definição textual, sem resolver em um primeiro momento os problemas descritos nesta seção.

Escolheu-se também os elementos notacionais de cada grupo básico de modelagem, necessários para modelar um processo de negócio. Com isso, o seguintes elementos foram escolhidos:

- Desvio Exclusivo (XOR).
- Desvio Inclusivo (OR).
- Desvio Paralelo (AND).
- Fluxo de Mensagem.
- Fluxo de Sequência.
- Tarefa de Serviço.
- Tarefa de Envio.
- Tarefa de Recebimento.
- Evento de Fim.

3.2 Desenvolvimento dos Meta-Algoritmos

A segunda etapa do incremento descreve o meio utilizado para identificar o que deve ser implementado das definições textuais dos elementos notacionais. Para isso, foram estudados formas de extração de código-fonte a partir de regras de negócio.

Para possibilitar o desenvolvimento dos meta-algoritmos, foi utilizado uma técnica para obter a lógica comportamental dos meta-algoritmos, sem interversão humana. O objetivo dessa geração era identificar a partir da definição textual de um elemento notacional, trechos que apresentam alguma lógica comportamental, como desvios. A partir dessa lógica obtida, ela é organizada em forma de algoritmo, formando os meta-algoritmos.

A técnica utlizada foi a *Regra de Negócio*. Por regra de negócio, entende-se como uma declaração que define algum comportamento do sistema de informação (KARAKOSTAS, 1990).

Utilizando como exemplo um trecho da definição textual do elemento notacional

XOR: "*If and only if none of the conditions evaluates to true, the token is passed on the default Sequence Flow*", é uma regra de negócio, pois traz uma definição do que deve ser feito com o *token*, quando nenhuma condição for avaliada como verdadeira. Segundo Karakostas (1990), durante a codificação de um sistema de informação, os desenvolvedores utilizam estas declarações para escrever o código fonte deste sistema. A Figura 3.2 apresenta uma possível implementação para o trecho de definição textual do XOR.

Figura 3.2: Trecho de código fonte que implementa uma regra de negócio.

```
1
2  if (condicoesVerdadeiras == 0)
3  {
4      DefaultSequenceFlow = token;
5  }
6
```

Fonte: dos Autores.

Neste exemplo, `condicoesVerdadeiras == 0` verifica se nenhuma das condições foi avaliada como verdadeira. Já `DefaultSequenceFlow = token` implementa a ação de passar o *token* para o fluxo de sequência padrão.

Este trabalho utilizou esse conceito de regras de negócio para a obtenção dos fragmentos para construção dos meta-algoritmos. Só que neste contexto optou-se desenvolver a implementação em uma lógica textual para ser independente de linguagem de programação.

Para a geração das regras foi analisado a definição textual de cada elemento notacional. Após, foram identificadas as regras de negócio e gerados os fragmentos dos meta-algoritmos. A geração foi realizada manualmente, pois a identificação das regras de negócio é uma tarefa complexa e documentos como especificações são destinados a seres humanos, normalmente descritos em linguagem natural, como português ou inglês (NASCIMENTO, 2014).

O código obtido foi organizado na forma algorítmica, obtendo os *meta-algoritmos*. Como primeiro passo dessa organização, foram definidos símbolos para representar a ideia de itens que aparecem repetidamente ao longo dos meta-algoritmos, de modo a tornar a lógica dos meta-algoritmos mais concisa.

Foi tomada como base para a criação dos símbolos a abreviatura como por exemplo: condição (c), exceção (e). Também foram considerados símbolos especiais que representam uma ação ou estado, como a atribuição($\leftarrow$) e o vazio ($\emptyset$). A Tabela 3.1 ilustra os símbolos utilizados nos meta-algoritmos:

A partir das regras obtidas e símbolos definidos, têm-se os meta-algoritmos defi-

Tabela 3.1: Símbolos utilizados nos meta-algoritmos

Item	*Símbolo*
token	tk
entrada	i
saída	o
processo	P
fluxo de sequência	sf
fluxo de sequência padrão	sf_{padrao}
fluxo de mensagem	fm
condição	c
verdadeira	T
conjunto de condições	C
atribuição	$\leftarrow$
vazio	$\emptyset$
exceção	e
igual	$=$
diferente	$\neq$
maior	$>$
pertence	$\in$

Fonte: Os Autores

nidos, conforme mostrado a seguir:

- Meta-algoritmo 1: *Desvio Exclusivo (XOR).*

Definição: "Each token arriving at any incoming Sequence Flows activates the gateway and is routed to exactly one of the outgoing Sequence Flows. In order to determine the outgoing Sequence Flows that receives the token, the conditions are evaluated in order. The first condition that evaluates to true determines the Sequence Flow the token is sent to. No more conditions are henceforth evaluated. If and only if none of the conditions evaluates to true, the token is passed on the default Sequence Flow.

In case all conditions evaluate to false and a default flow has not been specified, an exception is thrown."

Meta-algoritmo:

- Meta-algoritmo 2: *Desvio Inclusivo (OR).*

Definição: "Upon execution, a token is consumed from each incoming Sequence Flow that has a token. A token will be produced on some of the outgoing Sequence Flows. In order to determine the outgoing Sequence Flows that receive a token, all conditions on the outgoing Sequence Flows are evaluated. The evaluation does not have to respect a certain order. For every condition which evaluates to true, a token MUST be passed on the respective Sequence Flow. If and only if none of the conditions evaluates to true, the token is passed on the default Sequence Flow.

Figura 3.3: Meta-algoritmo elemento notacional XOR.

1. *tk* chega ao desvio, $i \leftarrow tk$.
2. Para cada $c \in C$:
    a. Se $c = T$, então
        i. $sf \leftarrow tk$;
        ii. não testar mais c;
3. se todo $c \in C$ for F então
    a. se $sf_{padrão} \neq \emptyset$, então
        i. $sf_{padrão} \leftarrow tk$;
    b. senão *lançar e;*

Fonte: dos Autores.

In case all conditions evaluate to false and a default flow has not been specified, the Inclusive Gateway throws an exception."

Meta-algoritmo:

Figura 3.4: Meta-algoritmo elemento notacional OR.

1. Token chega à entrada do gateway $i \leftarrow tk$.
2. Para cada c:
    a. Se $c = T$, então $sf \leftarrow tk$.
3. Se $c = \emptyset$ e nenhuma condição verdadeira, então $sfpadrao \leftarrow tk$.
4. Se $c = \emptyset$, nenhuma condição verdadeira *e* $sfpadrao = \emptyset$, *então lançar e;*

Fonte: dos Autores.

- Meta-algoritmo 3: *Desvio Paralelo (AND).*

Definição: "The Parallel Gateway is activated if there is at least one token on each incoming Sequence Flow. The Parallel Gateway consumes exactly one token from each incoming Sequence Flow and produces exactly one token at each outgoing Sequence Flow. If there are excess tokens at an incoming Sequence Flow, these tokens remain at this Sequence Flow after execution of the Gateway."

Meta-algoritmo:

Figura 3.5: Meta-algoritmo elemento notacional AND.

1. *tk* chega à entrada do desvio.
2. *tk* é encaminhado para sf_{saida}.
3. Se quantidade de $tk > sf_{saida}$
    a. *tk* restantes permanecem em $sf_{entrada}$.

Fonte: dos Autores.

- Meta-algoritmo 4: *Fluxo de Mensagem.*

Definição: "A Message Flow is used to show the flow of Messages between two Participants that are prepared to send and receive them. A Message Flow MUST connect

two separate Pools. They connect either to the Pool boundary or to Flow Objects within the Pool boundary. They MUST NOT connect two objects within the same Pool. A Message Flow is a line with an open circle line start and an open arrowhead line end that MUST be drawn with a dashed single line."

Meta-algoritmo:

Figura 3.6: Meta-algoritmo fluxo de mensagem.

1. *fm* é conectado ao elemento se:
   a. $p_a \neq p_b$ ou $fo_a \neq fo_b$;

Fonte: dos Autores.

- Meta-algoritmo 5: *Fluxo de Sequência.*

Definição: "A Sequence Flow is used to show the order of Flow Elements in a Process or a Choreography. Each Sequence Flow has only one source and only one target. The source and target MUST be from the set of the following Flow Elements: Events (Start, Intermediate, and End), Activities (Task and Sub-Process; for Processes), Choreography Activities (Choreography Task and Sub-Choreography; for Choreographies), and Gateways. A Sequence Flow can optionally define a condition Expression, indicating that the token will be passed down the Sequence Flow only if the Expression evaluates to true. This Expression is typically used when the source of the Sequence Flow is a Gateway or an Activity."

Meta-algoritmo:

Figura 3.7: Meta-algoritmo fluxo de sequência.

1. se *origem* = *gateway* ou *origem* = *atividade*
   a. se $c \neq \emptyset$
      i. *testar c*;

Fonte: dos Autores.

- Meta-algoritmo 6: *Tarefa de serviço.*

Definição: "Upon activation, the data in the inMessage of the Operation is assigned from the data in the Data Input of the Service Task the Operation is invoked. On completion of the service, the data in the Data Output of the Service Task is assigned from the data in the outMessage of the Operation, and the Service Task completes. If the invoked service returns a fault, that fault is treated as interrupting error, and the Activity fails."

Meta-algoritmo:

- Meta-algoritmo 7: *Tarefa de envio.*

Figura 3.8: Meta-algoritmo Tarefa de Serviço.

1. *dataInput* ← *inMessage;*
2. invocar *operation*;
3. dataOutput ← *operation .outMessage;*
4. se serviço falhar
    a. criar interrupção de erro;

Fonte: dos Autores.

Definição: "Upon activation, the data in the associated Message is assigned from the data in the Data Input of the Send Task. The Message is sent and the Send Task completes."

Meta-algoritmo:

Figura 3.9: Meta-algoritmo Tarefa de Envio.

1. dados ← *message;*
2. enviar dados;
3. concluir tarefa.

Fonte: dos Autores.

- Meta-algoritmo 8: *Tarefa de recebimento.*

Definição: "Upon activation, the Receive Task begins waiting for the associated Message. When the Message arrives, the data in the Data Output of the Receive Task is assigned from the data in the Message, and Receive Task completes. For key-based correlation, only a single receive for a given CorrelationKey can be active, and thus the Message matches at most one Process instance. For predicate-based correlation, the Message can be passed to multiple Receive Tasks. If the Receive Task's instantiate attribute is set to true, the Receive Task itself can start a new Process instance."

Meta-algoritmo:

Figura 3.10: Meta-algoritmo Tarefa de Recebimento.

1. aguardar *m*;
2. receber *m*;
3. se *instantiate* = *T* então
    a. instanciar *P*;
4. senão *finalizar tarefa*;

Fonte: dos Autores.

- Meta-algoritmo 9: *Evento de fim.*

Definição: For a terminate End Event, the Process is abnormally terminated—no other ongoing Process instances are affected.

For all other End Events, the behavior associated with the Event type is performed, e.g., the associated Message is sent for a Message End Event, the associated signal is sent for a Signal End Event, and so on. The Process instance is then completed, if and only if the following two conditions hold:

- All start nodes of the Process have been visited. More precisely, all Start Events have been triggered, and for all starting Event-Based Gateways, one of the associated Events has been triggered.

- There is no token remaining within the Process instance.

Meta-algoritmo:

Figura 3.11: Meta-algoritmo Evento de Fim.

1. finalizar instância de *P* se:
   a. todos os nodos iniciais de P foram visitados e
   b. *tk em sub-processos* = Ø.

Fonte: dos Autores.

3.3 Considerações Finais

Neste capítulo foram descritos os meta-algoritmos desenvolvidos neste trabalho. Dada a definição de um elemento notacional, foi obtido o respectivo meta-algoritmo.

Escolheu-se os elementos notacionais de cada grupo básico de modelagem da BPMN. Além disso, considerou-se aqueles a qual a literatura foca a atenção, devido a problemas de entendimento por parte dos usuários e estruturais.

O desenvolvimento dos meta-algoritmos explorou a possibilidade de se desenvolver uma lógica que abrange as regras comportamentais descritas na definição dos elementos notacionais, na especificação da BPMN, mantendo conformidade com o que está definido na definição textual dos elementos notacionais.

Como principais limitações dessa etapa, pode-se afirmar que nem todas as definições textuais possuem um formato que facilitou o desenvolvimento dos meta-algoritmos. Por exemplo, na definição do meta-algoritmo 4 (fluxo de mensagem), são definidas regras do elemento em questão, porém não é dito qual procedimento seguir, caso tais regras não forem seguidas (ex.: É definido que dois elementos devem estar conectados em diferentes piscinas, porém não é dito o que deve ser feito se essa regra não ser seguida). Isso pode levar a diferentes interpretações por parte do leitor, no momento da implementação desse elemento.

Outros pontos identificados é que, considerando a definição dos elementos da especificação, não há algum tratamento definido tentando evitar problemas como *deadlock* ou *livelock*. Esse é outro ponto permite diversas implementações por parte das fabricantes de BPMS.

Uma possibilidade para abranger esse problema descrito, seria o desenvolvimento de meta-algoritmos alternativos que, além da lógica desenvolvida na definição textual dos elementos notacionais, consideraria também regras adicionais definidas na literatura, como as sete diretrizes de (MENDLING; REIJERS; AALST, 2010), ou padrão verbo-objeto de (GASSEN et al., 2014).

A partir do conjunto de meta-algoritmos desenvolvidos, a próxima etapa contempla a validação e verificação dos mesmos, descritos no capítulo a seguir.

4 VALIDAÇÃO E VERIFICAÇÃO DOS META-ALGORITMOS

Com o desenvolvimento dos meta-algoritmos, é necessário criar caminhos para evidenciar que o trabalho desenvolvido realmente está aderente às definições dos elementos notacionais da BPMN, além de que são aceitáveis em relação aos usuários. Para isso, o presente trabalho faz uso dos *testes de software* para atingir esses objetivos.

O teste de *software* é um elemento de um aspecto mais amplo, que frequentemente é referido como verificação e validação. A *verificação* trata do conjunto de atividades que garante que o *software* implementa corretamente os requisitos levantados. Já a *validação* trata de um conjunto de atividades que garante que o *software* construído corresponde ao desejo do cliente (PRESSMAN, 2006).

Neste trabalho, a verificação foi realizada através de *Tabelas de Decisão* e *Grafos de Fluxo de Controle* (CFG), considerando como requisitos as definições dos elementos notacionais. A validação foi implementada através de *questionário*, verificando a usabilidade dos usuários e capturando a respectiva aceitação.

4.1 Verificação Funcional

A primeira verificação foi realizada através da utilização de *tabelas de decisão* e *grafos de fluxo de controle* para verificar se os meta-algoritmos desenvolvidos estão aderentes as definições dos elementos notacionais. Este tipo de verificação é considerado um *teste funcional*.

Testes funcionais são aplicados a qualquer descrição do comportamento de um sistema desenvolvido, desde uma descrição informal até uma especificação oficial e em qualquer nível de granularidade. Uma especificação funcional é a fonte de informação mais importante para a elaboração de testes. A derivação de casos de testes a partir de uma especificação é chamada teste funcional (PEZZÈ; YOUNG, 2008; PATTON, 2006).

No contexto deste trabalho, a especificação dos elementos notacionais da BPMN pode ser considerada uma especificação funcional, visto que a partir desta é definido o comportamento esperado para a implementação do elemento notacional.

Os seguintes termos são necessários para entender a verificação aplicada neste trabalho, de acordo com Pezzè and Young (2008):

- **Caso de teste:** Consiste de um conjunto de entradas, condições de execução e um

critério de sucesso/falha.

- **Suíte de teste:** É um conjunto de casos de testes. Tipicamente, um método de teste funcional diz respeito à criação de uma suíte de teste.
- **Teste ou execução de teste:** Este termo é usado para denotar a atividade de executar casos de testes e avaliar seus resultados.

4.1.1 Tabelas de Decisão e Grafos de Fluxo de Controle

A primeira etapa para realização da verificação dos meta-algoritmos é a definição de uma estrutura de teste que permita representar a definição textual dos elementos notacionais da BPMN. Para tanto, foi utilizada a abordagem de *tabelas de decisão*.

Essa abordagem foi escolhida por conta das especificações que frequentemente são representadas como estruturas de decisão, tais como conjunto de decisões sobre valores de entrada e as ações ou resultados correspondentes. Um modelo da estrutura de decisão pode ser utilizada para selecionar casos de testes que podem revelar discrepâncias entre as decisões tomadas no código e a estrutura de decisão desejada (PEZZÈ; YOUNG, 2008).

Essa estrutura de decisão pode ser disposta na forma de tabela, na qual as linhas representam as condições básicas e as colunas representam as combinações básicas.

Considerando uma especificação fictícia como exemplo de construção de tabela de decisão, que determina o valor que uma variável (denominada y) deve assumir, com base na variável x:

"*De acordo com o valor de x, calcule o valor de y:*

*- **Se** x corresponder a 1, **então** y deverá receber o dobro de x, **senão deve ser testado** se x corresponde a 2. **Caso for verdadeiro, então** y deverá receber 2 na potência x. **Caso for falso, então verificar** se x corresponde a 3. Nesta situação, y receberá 2 divido por x.*"

O primeiro passo para construção da tabela de decisão é listar as condições presentes na especificação. São consideradas condições as sentenças que tratam alguma entrada, processamento, saída ou critério de decisão (PEZZÈ; YOUNG, 2008). No texto de exemplo, foram definidas como condições os trechos que tratam de algum processamento (ex.: estrutura *se*, *então*). Na especificação de exemplo, as sentenças destacadas se encaixam neste critério:

- Valor de x corresponde a 1? - Valores possíveis: *True* (T) ou *False* (F).

- Valor de x corresponde a 2? - Valores possíveis: T ou F.
- Valor de x corresponde a 3? - Valores possíveis: T ou F.

No exemplo, considerando a especificação, é identificado se x pode corresponder a três valores diferentes (1, 2 e 3). Essas regras identificadas configuram as condições a serem utilizadas na tabela de decisão.

O segundo passo é calcular o espaço de combinações dos valores possíveis. Como os valores possíveis são booleanos (T ou F), então o espaço de combinações é dado por $2^{\text{número de condições}}$. Na situação de exemplo foram identificadas 3 condições, então têm-se $2^3 = 8$ combinações possíveis.

No terceiro passo, é criada a tabela de decisão com a combinação de valores e as ações para as respectivas combinações. As ações são identificadas com base nas combinações das condições. A Figura 4.1 apresenta a tabela de decisão inicial.

Figura 4.1: Tabela de Decisão inicial

		Combinações							
Condições	*x* corresponde a 1	F	F	F	F	T	T	T	T
	x corresponde a 2	F	F	T	T	F	F	T	T
	x corresponde a 3	F	T	F	T	F	T	F	T
Ações	*y* recebe dobro de *x*					X	X	X	X
	y recebe potência de *x*			X	X				
	y recebe 2 dividido por *x*		X						
				1		2		3	

Fonte: dos Autores.

Na área das condições, é apresentada as combinações possíveis para as três condições levantadas. Cada coluna apresenta uma combinação das condições, sendo que *T* representa *True* e *F False*. Área das ações dispõe do conjunto de ações levantadas, sendo que a marcação "X" corresponde a ação para aquela determinada combinação de condições.

Considerando a segunda coluna das combinações, a leitura de uma combinação da tabela pode ser feita da seguinte forma: "Quando x corresponder a 1 for falso, x corresponder a 2 for falso e x corresponder a 3 for verdadeiro, então a ação y dividido por 2 pode ocorrer".

As colunas destacadas na Figura 4.1 (vide 1, 2 e 3) indicam colunas da tabela que podem ser minimizadas. Por exemplo, em 1 é possível que as colunas possuem valor idêntico, com exceção de um valor (indicado com a elipse). Isso indica que independentemente do valor assumido nesta condição, esta não afeta a ação a ser realizada. Com isso as colunas são unidas e onde se tinha os valores dispares, é adicionado o *don't care* ("-"). O mesmo procedimento é realizado em 2 e 3. Ao fim desta primeira etapa, é revisado as colunas, e se necessário o procedimento é repetido até não haver mais combinações entre as colunas. A minimização é necessária para eliminação de colunas redundantes.

Ao final da minimização, obtêm-se a Tabela de decisão final, conforme apresenta a Figura 4.2.

Figura 4.2: Tabela de Decisão minimizada

		Combinações			
Condições	x corresponde a 1	F	F	F	T
	x corresponde a 2	F	F	T	-
	x corresponde a 3	F	T	-	-
Ações	y recebe dobro de x				X
	y recebe potência de x			X	
	y recebe 2 dividido por x		X		

Fonte: dos Autores.

A partir de uma tabela de decisão final, têm-se uma suíte de teste que pode ser aplicada para testar uma implementação que seja representada na forma de um CFG.

O CFG é um grafo dirigido em que os nodos representam regiões do código fonte e os arcos representam a possibilidade da execução do programa prosseguir diretamente do fim de uma região para o começo de outra, por execução sequencial, ou por desvio (PEZZÈ; YOUNG, 2008). As regiões de códigos representadas pode ser comandos individuais ou operações individuais do programa. A Figura 4.3 ilustra um exemplo de CFG.

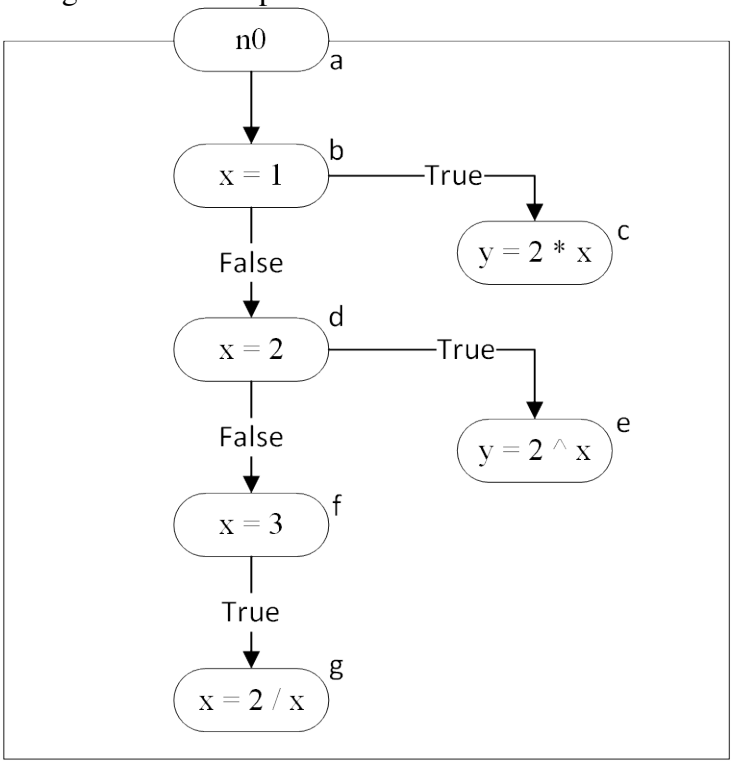
Realizando a implementação da especificação fictícia, que calcula o valor de y, têm-se:

```
if( x = 1 )
    y = 2 * x;
else
```

```
    if( x = 2 )
        y = 2 ^ x;
else
        if( x = 3 )
            y = 2 / x;
```

A partir deste é possível gerar o respectivo grafo de fluxo de controle, conforme Figura 4.3.

Figura 4.3: Exemplo de Grafo de Fluxo de Controle



Fonte: dos Autores.

Como exemplo, é considerada a Tabela de Decisão da Figura 4.2 representando uma especificação, o código dessa especificação, e o CFG da Figura 4.3 representando a implementação. Então, é necessário percorrer o CFG com cada um dos casos de testes (cada coluna da área *Combinações* da Figura 4.2). Cada caso de teste que percorre o grafo e chegar a um nodo final, é dito que aquele percusso está verificado. Caso algum caso de teste não alcance algum nodo final, então há alguma inconsistência (alguma inconformidade na implementação). Ao final do conjunto de testes, se todos os casos chegaram a um nodo final do grafo, então é possível afirmar, com base neste teste, que o código implementado *está aderente* à especificação.

Com base no conjunto de casos de testes da tabela de decisão da Figura 4.2, são realizados os seguintes percussos no grafo:

- Caso de teste t_1: {b=F, d=F, f=F} $\longrightarrow$ [Nodos percorridos p_1: a, b, d, f];
- Caso de teste t_2: {b=F, d=F, f=T} $\longrightarrow$ [Nodos percorridos p_2: a, b, d, f, g];
- Caso de teste t_3: {b=F, d=T, f=-} $\longrightarrow$ [Nodos percorridos p_3: a, b, d, e];
- Caso de teste t_4: {b=T, d=-, f=-} $\longrightarrow$ [Nodos percorridos p_4: a, b, c];

Cada caso de teste simula o valor que pode ser assumido em determinado nodo (para aqueles que possuem alguma condição definida). Por exemplo, em t_1 é apresentado um teste: *b*=*F*. Isso indica que a condição presente no nodo *b* ($x = 1$) será testada com o valor *falso* (*F*). A partir da combinação do valor assumido neste nodo com os demais, são identificados os nodos percorridos, mostrados em p_1. Nos casos de testes, cujos nodos apresentam o valor "-" (*don't care*), assume-se que independentemente do valor que vá assumir, este não afetará os nodos percorridos.

No exemplo descrito, todos os casos de testes alcançaram os nodos finais do CFG (*c, e, f, g*), então é possível afirmar, com base neste teste, que a implementação está representando sua respectiva especificação.

Neste trabalho as tabelas de decisão representam as definições textuais dos elementos notacionais, conforme especificado pela BPMN, e o CFG representam os meta-algoritmos. A partir destes, busca-se verificar se os meta-algoritmos estão aderentes as definições textuais, sendo este um dos objetivos específicos deste trabalho (vide seção 1.2).

4.1.2 Verificação Funcional dos Meta-algoritmos

A elaboração das tabelas de decisão e Grafos de Fluxo de Controle para a verificação deste trabalho seguiu os passos descritos na seção 4.1.1.

Os trechos em destaques no texto correspondem a condições que foram identificadas na especificação dos elementos notacionais. Assim como na elaboração dos meta-algoritmos, as tabelas de decisão utilizaram a especificação oficial da BPMN (OMG, 2013).

- Meta-algoritmo 1: *Desvio Exclusivo (XOR).*

Definição: "Each token arriving at any incoming Sequence Flows activates the gateway and is routed to exactly one of the outgoing Sequence Flows. In order to determine the outgoing Sequence Flows that receives the token, the conditions are evaluated in order. **The first condition that evaluates to true** determines the Sequence Flow the token is sent to. No more conditions are henceforth evaluated.

If and only if none of the conditions evaluates to true, the token is passed on the default Sequence Flow.

In case all conditions evaluate to false and a **default flow has not been specified**, an exception is thrown."

Meta-algoritmo:

Figura 4.4: Meta-algoritmo elemento notacional XOR.

1. *tk* chega ao desvio, $i \leftarrow tk$.
2. Para cada $c \in C$:
   a. Se $c = T$, então
      i. $sf \leftarrow tk$;
      ii. não testar mais c;
3. se todo $c \in C$ for F então
   a. se $sf_{padrão} \neq \emptyset$, então
      i. $sf_{padrão} \leftarrow tk$;
   b. senão *lançar e*;

Fonte: dos Autores.

Primeiro passo consiste na identificação das condições. Os trechos destacados no texto correspondem as condições levantadas:

- *The first was evaluated to true?* - Valores possíveis: *True* (T) ou *False* (F).
- *None of the conditions evaluated to true?* - Valores possíveis: T ou F.
- *A default flow has been specified*? - Valores possíveis: T ou F.

A partir das condições identificadas, o segundo passo foi definir a tabela de decisão, conforme Tabela 4.1.

Tabela 4.1: Tabela de decisão elemento notacional XOR.

		Combinações		
Condições	*The first was evaluated to true?*	F	F	T
	None of the conditions evaluated to true?	T	T	-
	A default flow has been specified?	F	T	-
Ações	*The token is sent to sequence flow*			X
	No more conditions are henceforth evaluated			X
	The token is passed on the default Sequence Flow		X	
	An exception is thrown	X		

Fonte: Os Autores

A partir da Tabela de decisão, têm-se a suíte de testes para validar o meta-algoritmo definido para o elemento notacional XOR. Com isso, o terceiro passo foi elaborar o Grafo de Fluxo de Controle, apresentado na Figura 4.5.

A partir do conjunto de casos de testes da tabela de decisão, são realizados os seguintes percussos no CFG definido:

- Caso de teste t_1: {d=F, g=T, h=F} $\longrightarrow$ [Nodos percorridos p_1: a, b, c, d, g, h, j];
- Caso de teste t_2: {d=F, g=T, h=T} $\longrightarrow$ [Nodos percorridos p_2: a, b, c, d, g, h, i];
- Caso de teste t_3: {d=T, g=-, h=-} $\longrightarrow$ [Nodos percorridos p_3: a, b, c, d, e, f];

Para esta suíte de teste, todos os casos de testes alcançaram os nodos finais do CFG (*f, i, j*), então é possível afirmar, que o meta-algoritmo do elemento notacional XOR está aderente à respectiva definição textual.

- Meta-algoritmo 2: *Desvio Inclusivo (OR).*

Definição: "Upon execution, a token is consumed from each incoming Sequence Flow that has a token. A token will be produced on some of the outgoing Sequence Flows. In order to determine the outgoing Sequence Flows that receive a token, **all conditions on the outgoing Sequence Flows are evaluated**. The evaluation does not have to respect a certain order. **For every condition** which evaluates to true, a token MUST be passed on the respective Sequence Flow.

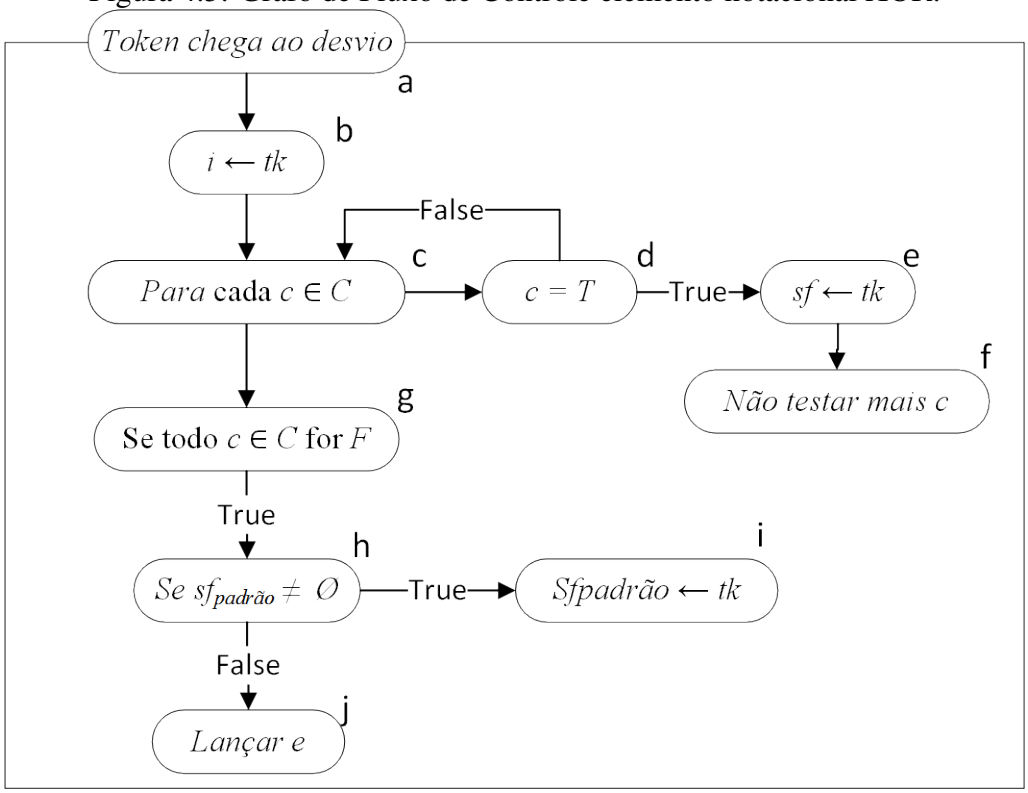
If and only if none of the conditions evaluates to true, **the token is passed on the default Sequence Flow**.

In case **all conditions evaluate to false and a default flow has not been specified**, the Inclusive Gateway throws an exception."

Meta-algoritmo:

Primeiro passo consiste na identificação das condições. Os trechos destacados no

Figura 4.5: Grafo de Fluxo de Controle elemento notacional XOR.



Fonte: dos Autores.

Figura 4.6: Meta-algoritmo elemento notacional OR.

1. Token chega à entrada do gateway $i \leftarrow tk$.
2. Para cada c:
   a. Se $c = T$, então $sf \leftarrow tk$.
3. Se $c = \emptyset$ e nenhuma condição verdadeira, então $sfpadrao \leftarrow tk$.
4. Se $c = \emptyset$, nenhuma condição verdadeira *e* $sfpadrao = \emptyset$, *então lançar e;*

Fonte: dos Autores.

texto correspondem as condições levantadas:

- *The conditions are evaluated?* - Valores possíveis: *True* (T) ou *False* (F).
- *None of the conditions evaluates to true*? - Valores possíveis: T ou F.
- *All conditions evaluate to false and default flow has not been specified*? - Valores possíveis: T ou F.

A partir das condições identificadas, o segundo passo foi definir a tabela de decisão, conforme Tabela 4.2.

A partir da Tabela de decisão, têm-se a suíte de testes para validar o meta-algoritmo definido para o elemento notacional OR. Com isso, o terceiro passo foi elaborar o Grafo de Fluxo de Controle, apresentado na Figura 4.5.

A partir do conjunto de casos de testes da tabela de decisão, são realizados os seguintes percussos no CFG definido:

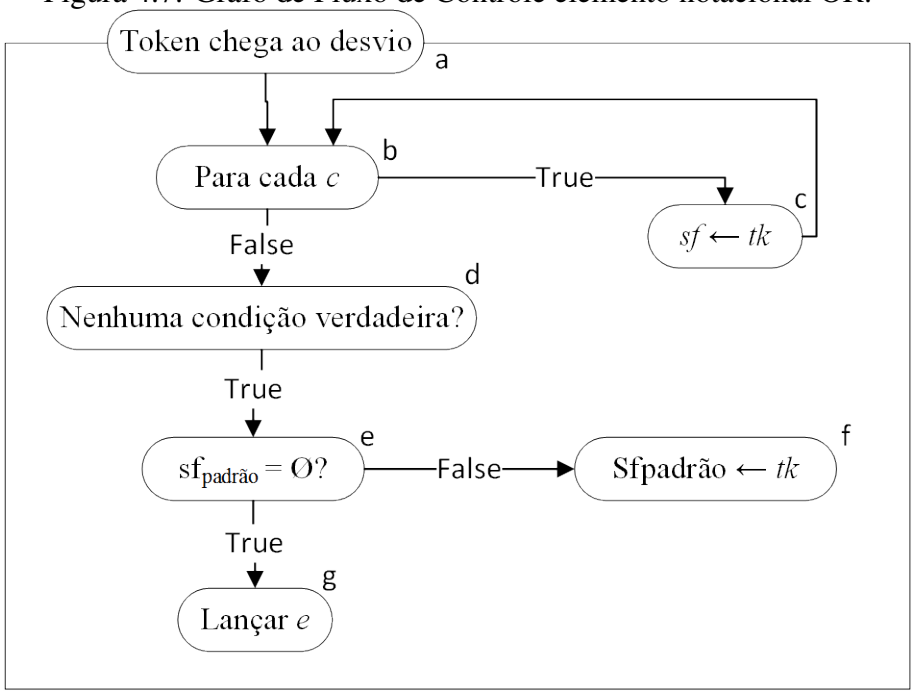
- Caso de teste t_1: {b=F, d=T, e=F} $\longrightarrow$ [Nodos percorridos p_1: a, b, d, e, f];
- Caso de teste t_2: {b=F, e=T, f=T} $\longrightarrow$ [Nodos percorridos p_2: a, b, d, e, g];

Tabela 4.2: Tabela de decisão elemento notacional OR.

		Combinações		
Condições	*The conditions are evaluated?*	F	F	T
	None of the conditions evaluates to true?	T	T	-
	All conditions evaluate to false and default flow has not been specified?	F	T	-
Ações	*The token is sent to sequence flow*			X
	The token is passed on the default Sequence Flow	X		
	An exception is thrown		X	

Fonte: Os Autores

Figura 4.7: Grafo de Fluxo de Controle elemento notacional OR.



Fonte: dos Autores.

- Caso de teste t_3: {b=T, e=-, f=-} $\longrightarrow$ [Nodos percorridos p_3: a, b, c];

Para esta suíte de teste, todos os casos de testes alcançaram os nodos finais do CFG (*f, g, c*), então é possível afirmar, que o meta-algoritmo do elemento notacional OR está aderente à respectiva definição textual.

- Meta-algoritmo 3: *Desvio Paralelo (AND).*

Definição: "The Parallel Gateway is activated if there is at least one token on each incoming Sequence Flow. The Parallel Gateway consumes exactly one token from each incoming Sequence Flow and produces exactly one token at each outgoing Sequence Flow. **If there are excess tokens** at an incoming Sequence Flow, these tokens remain at this Sequence Flow after execution of the Gateway."

Meta-algoritmo:

Primeiro passo consiste na identificação das condições. Os trechos destacados no texto correspondem as condições levantadas:

Figura 4.8: Meta-algoritmo elemento notacional AND.

1. *tk* chega à entrada do desvio.
2. *tk* é encaminhado para $sf_{saída}$.
3. Se quantidade de *tk* > $sf_{saída}$
   a. *tk* restantes permanecem em $sf_{entrada}$.

Fonte: dos Autores.

- *Are there excess tokens?* - Valores possíveis: *True* (T) ou *False* (F).

A partir da condição identificada, o segundo passo foi definir a tabela de decisão, conforme Tabela 4.3.

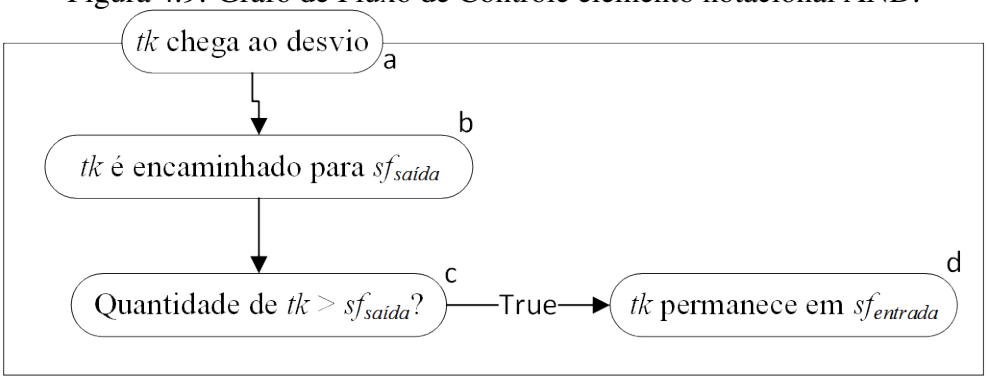
Tabela 4.3: Tabela de decisão elemento notacional AND.

		Combinações	
Condição	*Are there excess tokens?*	F	T
Ação	*The tokens remain at Sequence Flow*		X

Fonte: Os Autores

A partir da Tabela de decisão, têm-se a suíte de testes para validar o meta-algoritmo definido para o elemento notacional AND. Com isso, o terceiro passo foi elaborar o Grafo de Fluxo de Controle, apresentado na Figura 4.9.

Figura 4.9: Grafo de Fluxo de Controle elemento notacional AND.



Fonte: dos Autores.

A partir do conjunto de casos de testes da tabela de decisão, são realizados os seguintes percussos no CFG definido:

- Caso de teste t_1: {c=T} $\longrightarrow$ [Nodos percorridos p_1: a, b, c, d];
- Caso de teste t_2: {c=F} $\longrightarrow$ [Nodos percorridos p_2: a, b, c];

Para esta suíte de teste, todos os casos de testes alcançaram os nodos finais do CFG (*c, d*), então é possível afirmar, que o meta-algoritmo do elemento notacional AND está aderente à respectiva definição textual.

- Meta-algoritmo 4: *Fluxo de Mensagem.*

Definição: "A Message Flow is used to show the flow of Messages between two Participants that are prepared to send and receive them.

A Message **Flow MUST connect two separate Pools** (*p*). They connect either to the Pool boundary or to Flow Objects (*fo*) within the Pool boundary. **They MUST NOT connect two objects within the same Pool**. A Message Flow is a line with an open circle line start and an open arrowhead line end that MUST be drawn with a dashed single line."

Meta-algoritmo:

Figura 4.10: Meta-algoritmo fluxo de mensagem.

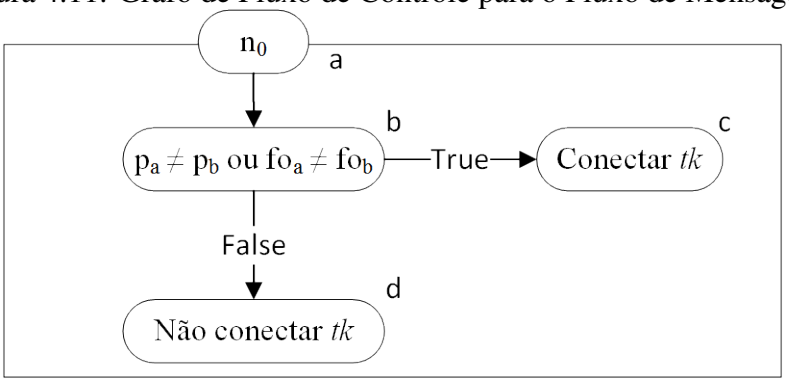
1. *fm* é conectado ao elemento se:
   a. $p_a \neq p_b$ ou $fo_a \neq fo_b$;

Fonte: dos Autores.

Nesta situação não está definida alguma condição especifica. O texto expressa alguma regras (destacadas no texto) que devem ser seguidas, contudo não define o que deve ser feito se estas não forem seguidas (ex.: se um fluxo de sequencia não estiver conectando duas *pools*). Isto é, falta as ações para poder constituir uma tabela de decisão.

Contudo, mesmo assim foi elaborado o CFG para o meta-algoritmo desenvolvido. A Figura 4.11 apresenta o grafo obtido.

Figura 4.11: Grafo de Fluxo de Controle para o Fluxo de Mensagem.



Fonte: dos Autores.

O CFG mostra que partindo de um estado inicial (nodo *a*), a execução passa por uma condição (nodo *b*), que verifica se o fluxo de mensagem está tentando conectar duas *pools* diferentes ($p_a \neq p_b$), ou dois elementos de *pools* diferentes ($fo_a \neq fo_b$). Esta condição implementa as regras destacas na definição do fluxo de mensagem.

Considerando que o meta-algoritmo implementa as regras destacadas, então pode-se dizer que este está aderente a definição textual.

- Meta-algoritmo 5: *Fluxo de Sequência.*

Definição: "A Sequence Flow is used to show the order of Flow Elements in a Process or a Choreography. Each Sequence Flow has only one source and only one target.

The source and target MUST be from the set of the following Flow Elements: Events (Start, Intermediate, and End), Activities (Task and Sub-Process; for Processes), Choreography Activities (Choreography Task and Sub-Choreography; for Choreographies), and Gateways.

A Sequence Flow can optionally **define a condition Expression**, indicating that the token will be passed down the Sequence Flow only if the Expression evaluates to true.

This Expression is typically used when the source of **the Sequence Flow is a Gateway or an Activity**."

Meta-algoritmo:

Figura 4.12: Meta-algoritmo para o Fluxo de Sequência.

1. se *origem* = *gateway* ou *origem* = *atividade*
   a. se $c \neq \emptyset$
      i. *testar c*;

Fonte: dos Autores.

Primeiro passo consiste na identificação das condições. Os trechos destacados no texto correspondem as condições levantadas:

- *Source is gateway or activity?* - Valores possíveis: *True* (T) ou *False* (F).
- *Is there a expression?* - Valores possíveis: T ou F.

A partir das condições identificadas, o segundo passo foi definir a tabela de decisão, conforme Tabela 4.4.

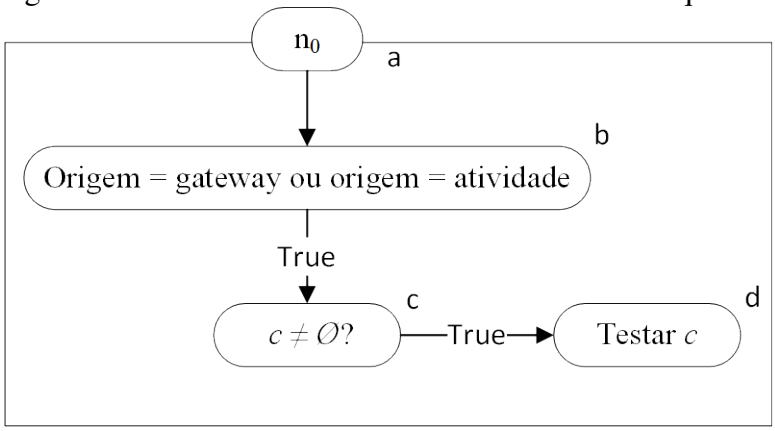
Tabela 4.4: Tabela de decisão Fluxo de Sequência.

		Combinações		
Condições	*Source is gateway or activity?*	F	T	T
	Is there a expression?	-	F	T
Ações	*Evaluate the expression*			X

Fonte: Os Autores

A partir da Tabela de decisão, têm-se a suíte de testes para validar o meta-algoritmo definido para o fluxo de sequência. Com isso, o terceiro passo foi elaborar o Grafo de Fluxo de Controle, apresentado na Figura 4.13.

Figura 4.13: Grafo de Fluxo de Controle Fluxo de Sequência.



Fonte: dos Autores.

A partir do conjunto de casos de testes da tabela de decisão, são realizados os seguintes percussos no CFG definido:

- Caso de teste t_1: {b=F, c=-} $\longrightarrow$ [Nodos percorridos p_1: a, b];
- Caso de teste t_2: {b=T, c=F} $\longrightarrow$ [Nodos percorridos p_2: a, b, c];

- Caso de teste t_3: {b=T, c=T} $\longrightarrow$ [Nodos percorridos p_3: a, b, c, d];

Para esta suíte de teste, todos os casos de testes alcançaram os nodos finais do CFG (*b, c, d*), então é possível afirmar, que o meta-algoritmo do fluxo de sequência está aderente à respectiva definição textual.

- Meta-algoritmo 6: *Tarefa de serviço.*

Definição: "Upon activation, the data in the inMessage of the Operation is assigned from the data in the Data Input of the Service Task the Operation is invoked. On completion of the service, the data in the Data Output of the Service Task is assigned from the data in the outMessage of the Operation, and the Service Task completes.

If the invoked service returns a fault, that fault is treated as interrupting error, and the Activity fails."

Meta-algoritmo:

Figura 4.14: Meta-algoritmo Tarefa de Serviço.

1. *dataInput* ← *inMessage;*
2. invocar *operation*;
3. dataOutput ← *operation .outMessage;*
4. se serviço falhar
    a. criar interrupção de erro;

Fonte: dos Autores.

Primeiro passo consiste na identificação das condições. O trecho destacado no texto corresponde a condição levantada:

- *The invoked service returned a fault?* - Valores possíveis: *True* (T) ou *False* (F).

A partir da condição identificada, o segundo passo foi definir a tabela de decisão, conforme Tabela 4.5.

Tabela 4.5: Tabela de decisão para o elemento Tarefa de Serviço.

		Combinações	
Condição	*The invoked service returned a fault?*	F	T
Ação	*The fault is treated as interrupting error*		X

Fonte: Os Autores

A partir da Tabela de decisão, têm-se a suíte de testes para validar o meta-algoritmo definido para tarefa de serviço. Com isso, o terceiro passo foi elaborar o Grafo de Fluxo de Controle, apresentado na Figura 4.15.

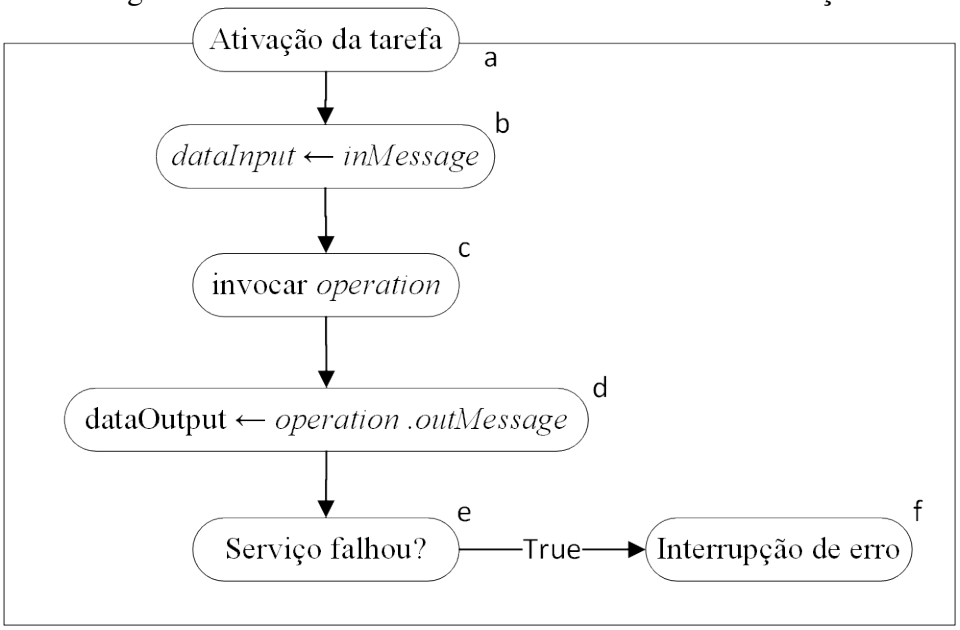
A partir do conjunto de casos de testes da tabela de decisão, são realizados os seguintes percussos no CFG definido:

- Caso de teste t_1: {e=F} $\longrightarrow$ [Nodos percorridos p_1: a, b, c, d, e];
- Caso de teste t_2: {e=T} $\longrightarrow$ [Nodos percorridos p_2: a, b, c, d, e, f];

Para esta suíte de teste, todos os casos de testes alcançaram os nodos finais do CFG (*e, f*), então é possível afirmar, que o meta-algoritmo do elemento tarefa de serviço está aderente à respectiva definição textual.

- Meta-algoritmo 7: *Tarefa de envio.*

Figura 4.15: Grafo de Fluxo de Controle Tarefa de Serviço.



Fonte: dos Autores.

Definição: "Upon activation, the data in the associated *Message* is assigned from the data in the Data Input of the Send Task. The Message is sent and the Send Task completes."

Meta-algoritmo:

Figura 4.16: Meta-algoritmo para o elemento Tarefa de Envio.

1. dados ← *message;*
2. enviar dados;
3. concluir tarefa.

Fonte: dos Autores.

Neste caso a definição da tarefa de serviço não define nenhuma condição, apenas ações sequencias: Obter os dados associados à *Message*, enviar a mensagem e finalizar a tarefa. Com isso, não é possível definir a tabela de decisão a partir desta definição.

Contudo, foi elaborado o CFG para o meta-algoritmo desenvolvido. A Figura 4.17 apresenta o grafo obtido.

O CFG apresenta as ações descritas na definição:

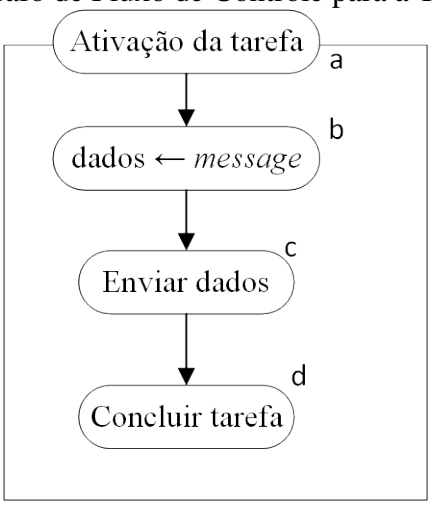
- Nodo b: $dados \leftarrow message$ - representa a associação do dado de *Message* na tarefa de envio.
- Nodo c: $enviardados$ - representa o envio da mensagem.
- Nodo d: $concluirtarefa$ - representa completar a tarefa de envio.

Com isso é possível dizer que o meta-algoritmo implementa as regras destacadas, então pode-se dizer que este está aderente a definição textual.

- Meta-algoritmo 8: *Tarefa de recebimento.*

Definição: "Upon activation, the Receive Task begins waiting for the associated Message. When the Message arrives, the data in the Data Output of the Receive Task

Figura 4.17: Grafo de Fluxo de Controle para a Tarefa de Envio.



Fonte: dos Autores.

is assigned from the data in the Message, and Receive Task completes. For key-based correlation, only a single receive for a given CorrelationKey can be active, and thus the Message matches at most one Process instance. For predicate-based correlation, the Message can be passed to multiple Receive Tasks. **If the Receive Task's** instantiate attribute is set to true, the Receive Task itself can start a new Process instance."

Meta-algoritmo:

Figura 4.18: Meta-algoritmo para o elemento Tarefa de Recebimento.

1. aguardar *m*;
2. receber *m*;
3. se *instantiate* = *T* então
   a. instanciar *P*;
4. senão *finalizar tarefa*;

Fonte: dos Autores.

Primeiro passo consiste na identificação das condições. O trecho destacado no texto corresponde a condição levantada:

- *Is the instantiate attribute set to true?* - Valores possíveis: *True* (T) ou *False* (F).

A partir da condição identificada, o segundo passo foi definir a tabela de decisão, conforme Tabela 4.6.

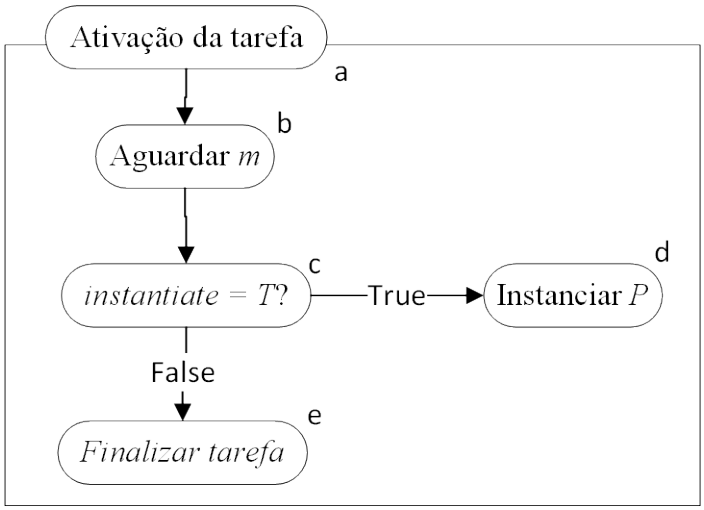
Tabela 4.6: Tabela de decisão elemento Tarefa de Recebimento.

		Combinações	
Condição	*Is the instantiate attribute set to true?*	F	T
Ações	*Start a new Process instance*		X
	Only complete the Receive Task	X	

Fonte: Os Autores

A partir da Tabela de decisão, têm-se a suíte de testes para validar o meta-algoritmo definido para a tarefa de recebimento. Com isso, o terceiro passo foi elaborar o Grafo de Fluxo de Controle, apresentado na Figura 4.19.

Figura 4.19: Grafo de Fluxo de Controle Tarefa de Recebimento.



Fonte: dos Autores.

A partir do conjunto de casos de testes da tabela de decisão, são realizados os seguintes percussos no CFG definido:

- Caso de teste t_1: {c=F} $\longrightarrow$ [Nodos percorridos p_1: a, b, c, d];
- Caso de teste t_2: {c=T} $\longrightarrow$ [Nodos percorridos p_2: a, b, c, e];

Para esta suíte de teste, todos os casos de testes alcançaram os nodos finais do CFG (*d, e*), então é possível afirmar, que o meta-algoritmo do elemento tarefa de recebimento está aderente à respectiva definição textual.

- Meta-algoritmo 9: *Evento de fim.*

Definição: For a terminate End Event, the Process is abnormally terminated — no other ongoing Process instances are affected.

For all other End Events, the behavior associated with the Event type is performed, e.g., the associated Message is sent for a Message End Event, the associated signal is sent for a Signal End Event, and so on. The Process instance is then completed, **if and only if the following two conditions hold**:

- **All start nodes of the Process have been visited.** More precisely, all Start Events have been triggered, and for all starting Event-Based Gateways, one of the associated Events has been triggered.

- **There is no token remaining** within the Process instance.

Meta-algoritmo:

Figura 4.20: Meta-algoritmo Evento de Fim.

1. finalizar instância de *P* se:
   a. todos os nodos iniciais de P foram visitados e
   b. *tk em sub-processos* = Ø.

Fonte: dos Autores.

Primeiro passo consiste na identificação das condições. Os trechos destacados no texto correspondem as condições levantadas:

- *All start nodes of the Process have been visit?* - Valores possíveis: *True* (T) ou

False (F).

- *Is there remaining tokens?* - Valores possíveis: T ou F.

A partir das condições identificadas, o segundo passo foi definir a tabela de decisão, conforme Tabela 4.7.

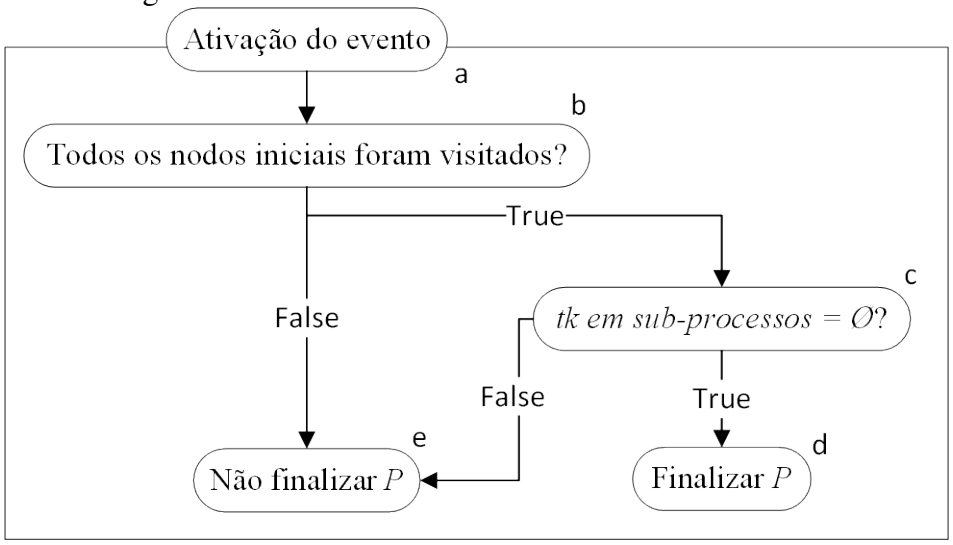
Tabela 4.7: Tabela de decisão Evento de Fim.

		Combinações		
Condições	*Source is gateway or activity?*	F	T	T
	Is there a expression?	-	F	T
Ações	*The process instance is finalized*			X
	The process instance is not finalized	X	X	

Fonte: Os Autores

A partir da Tabela de decisão, têm-se a suíte de testes para validar o meta-algoritmo definido para o evento de fim. Com isso, o terceiro passo foi elaborar o Grafo de Fluxo de Controle, apresentado na Figura 4.21.

Figura 4.21: Grafo de Fluxo de Controle Evento de Fim.



Fonte: dos Autores.

A partir do conjunto de casos de testes da tabela de decisão, são realizados os seguintes percussos no CFG definido:

- Caso de teste t_1: {b=F, c=-} $\longrightarrow$ [Nodos percorridos p_1: a, b, e];
- Caso de teste t_2: {b=T, c=F} $\longrightarrow$ [Nodos percorridos p_2: a, b, c, e];
- Caso de teste t_3: {b=T, c=T} $\longrightarrow$ [Nodos percorridos p_3: a, b, c, d];

Para esta suíte de teste, todos os casos de testes alcançaram os nodos finais do CFG (*d, e*), então é possível afirmar, que o meta-algoritmo do evento de fim está aderente à respectiva definição textual.

A partir da verificação funcional dos meta-algoritmos, partiu-se para a realização da validação da aceitação dos usuários em relação ao que foi desenvolvido.

4.2 Validação Com Usuários

A segunda verificação trata da validação dos meta-algoritmos juntamente com os usuários. O objetivo da validação é verificar a aceitabilidade, visando identificar se os meta-algoritmos facilitam o entendimento do usuários, quanto às definições dos elementos notacionais.

Essa validação ocorre por meio de testes com usuários, que fornecem entradas e conselhos sobre o testes de um sistema. As influências do ambiente de trabalho do usuário têm efeito importante sobre a confiabilidade, o desempenho, a usabilidade e a robustez de um sistema (SOMMERVILLE, 2011).

A validação baseada em participação dos usuários é particularmente importante para validar a usabilidade do *software*. Pode ser aplicado em diversos estágios de desenvolvimento e pode ser utilizado como diferentes objetivos, como por exemplo: capturar o modelo mental do usuários finais, avaliar alternativas de projeto, testar a usabilidade (PEZZÈ; YOUNG, 2008).

Existem três tipos de testes com o usuário, que segundo Sommerville (2011), são:

- *Teste alfa*, em que os usuários de *software* trabalham diretamente com a equipe que desenvolve o sistema, efetuando os testes no local de desenvolvimento.
- *Teste beta*, em que um *release* do *software* é disponibilizado aos usuários para que possam experimentar e levantar pontos de melhoria descobertos juntamente com os desenvolvedores do sistema.
- *Teste de aceitação*, em que clientes experimentam um sistema para decidir se está ou não pronto para ser aceito pelos desenvolvedores de sistemas e implantado no ambiente do cliente.

Destes três tipos, o teste de aceitação é o mais adequado de ser aplicado à este trabalho, visto que o objetivo é avaliar a usabilidade dos meta-algoritmos, visando identificar dificuldades e obstáculos, medidos através da interação usuário e meta-algoritmo, além de medir características, como o grau de dificuldade ao implementá-los.

Para possibilitar a interação, aplicou-se a validação utilizando uma estrutura de questionário, objetivando capturar informações estatísticas dos usuários, como experiência em desenvolvimento de *software*, modelagem de processos. As questões foram escolhidas de forma que fosse possível conhecer melhor os participantes, além de ser possível obter a opinião sobre o uso dos meta-algoritmos.

Foram analisados dados de 30 participantes, sendo eles: estudantes de cursos da área da computação de universidades brasileiras, desenvolvedores de *software*, entre outros.

4.2.1 Questionário Aplicado

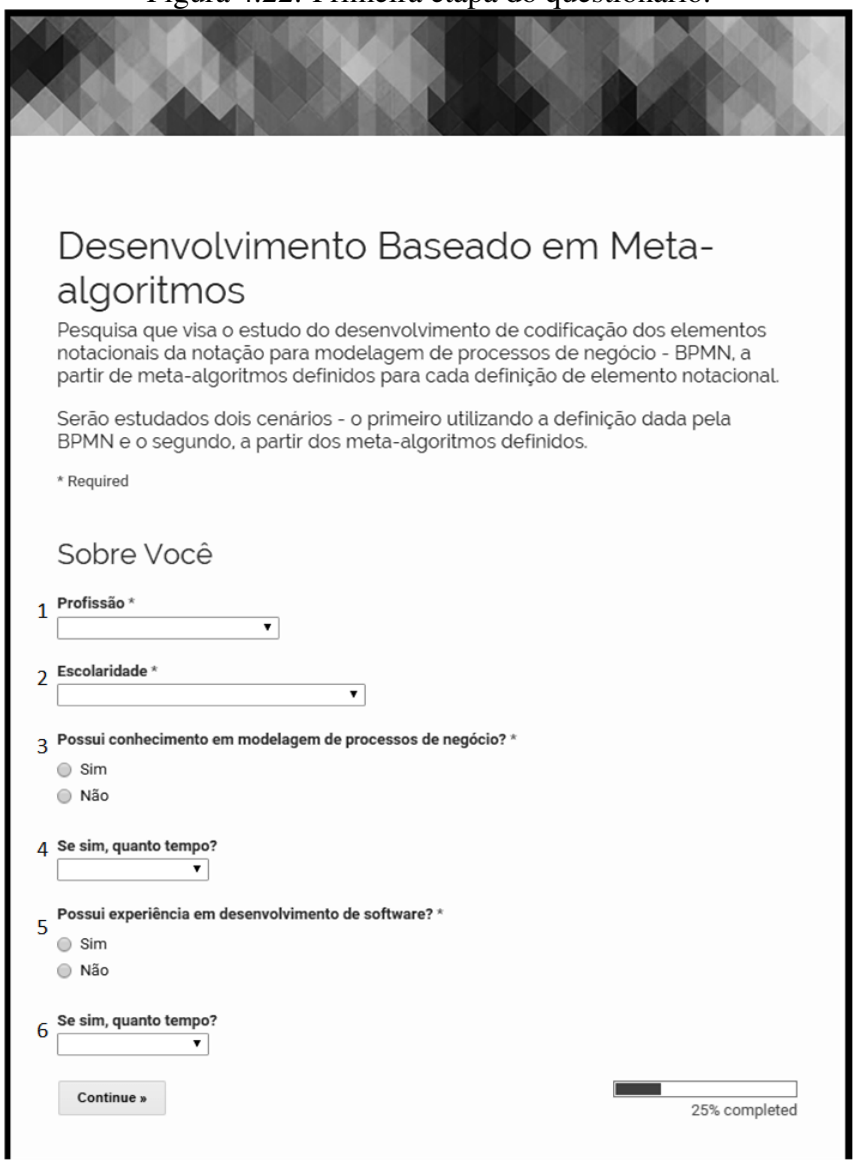
A estrutura de questionário utilizada foi a do *Google Forms*[1], que possibilita uma fácil construção de questionários. O período em que ficou disponível foi de 15 de novembro de 2015 à 15 de dezembro de 2015. Foi amplamente divulgado em redes sociais, sites, dentre outros meios. O único requisito para participação era conhecimentos básicos em programação, por parte do participante.

O questionário foi dividido em 4 etapas. A primeira tinha como objetivo levan-

[1] http://www.google.com/forms/about/

tar informações sobre o participante. De inicio, o formulário apresenta os objetivos da questionário ao participante, explicando também o seu formato. A Figura 4.22 ilustra o formulário desta etapa, cujos campos são os seguintes:

Figura 4.22: Primeira etapa do questionário.



Fonte: dos Autores.

1 - *Profissão:* Visava identificar a profissão do participante. Como opções disponíveis tinha-se: Desenvolvedor de software, professor, estudante e outros.

2 - *Escolaridade:* Visava identificar o nível de escolaridade do participante. Como opções disponíveis tinha-se: Ensino Médio, Superior Incompleto, Superior Completo, Pós-graduação (mestrado ou doutorado).

3 - *Possui conhecimentos em modelagem de processos de negócio:* Pergunta com o objetivo de identificar se o participante possuía algum conhecimento de modelagem de processos de negócio. Como opções disponíveis tinha-se: Sim ou Não.

4 - A resposta dessa opção era condicionada a resposta da pergunta 3. Ao responder sim, o participante informava o tempo de experiência em modelagem de processos de negócio. Opções disponíveis: Menos de 2 anos, de 2 a 4 anos, mais de 4 anos.

5 - *Possui experiência em desenvolvimento de software*. Semelhante a questão 3, visava identificar se o participante possuía alguma experiência em desenvolvimento de *software*. Como opções disponíveis tinha-se Sim ou Não.

6 - Resposta condicionada à resposta da pergunta 5. Semelhante a questão 4, ao responder sim, o participante informava o tempo de experiência em desenvolvimento de *software*. Opções disponíveis: Menos de 2 anos, de 2 a 4 anos, mais de 4 anos.

Todos os campos eram obrigatórios, exceto aqueles em que era condicionada a resposta anterior. Por se tratar de um questionário que visava capturar a aceitação do usuário com o desenvolvimento, utilizando os meta-algoritmos, objetivou-se identificar sua experiência com o desenvolvimento de *software* (questões 5 e 6) e modelagem de processos de negócio (3 e 4). Esse conhecimentos influenciam no desenvolvimento do meta-algoritmo, sendo que a experiência em desenvolvimento trata da capacidade de desenvolver a lógica de implementação, e o conhecimento em modelagem foca na capacidade de entender sobre qual elemento notacional está sendo realizada a implementação.

Além destas, buscou-se identificar nas questões 1 e 2 as origens do participante, a partir de sua escolaridade e profissão, para fins informativos.

Após o preenchimento dos dados, o participante passava para a segunda etapa, conforme apresenta a Figura 4.23. Inicialmente é introduzido o objetivo desta etapa, ao participante, que consistia em desenvolver a lógica dos elementos notacionais XOR (mostrado em 7) e OR (mostrado em 8). Estes elementos foram escolhidos pois são os mais estudados em termos de entendimento do usuário (KOSSAK; ILLIBAUER; GEIST, 2012; FIGL; RECKER; MENDLING, 2013; DEHNERT; AALST, 2004) e modelagem (MENDLING; REIJERS; AALST, 2010).

A estrutura da segunda etapa do questionário foi realizada da seguinte forma: é apresentado o elemento notacional, o seu desenho e a definição textual. A partir dessa definição, o participante implementava no campo "Implemente o respectivo código" a lógica comportamental do elemento notacional. Após era informado o grau de dificuldade da implementação, que era baseada na escala Likert (LIKERT, 1932; EDWARDS, 1983) de cinco pontos, variando de fácil até difícil. Assim como na primeira etapa, os campos de implementação e grau de dificuldade eram de obrigatório preenchimento.

A linguagem de programação para implementação da lógica era de escolha do participante. Assim que completasse a implementação, o participante passava para a próxima etapa.

A terceira etapa do questionário visava o desenvolvimento com os meta-algoritmos. A disposição dos campos no formulário é semelhante a segunda etapa, diferindo que o participante utiliza o meta-algoritmo para o desenvolvimento da lógica, ao invés da definição textual. A Figura 4.24 ilustra o formulário aplicado.

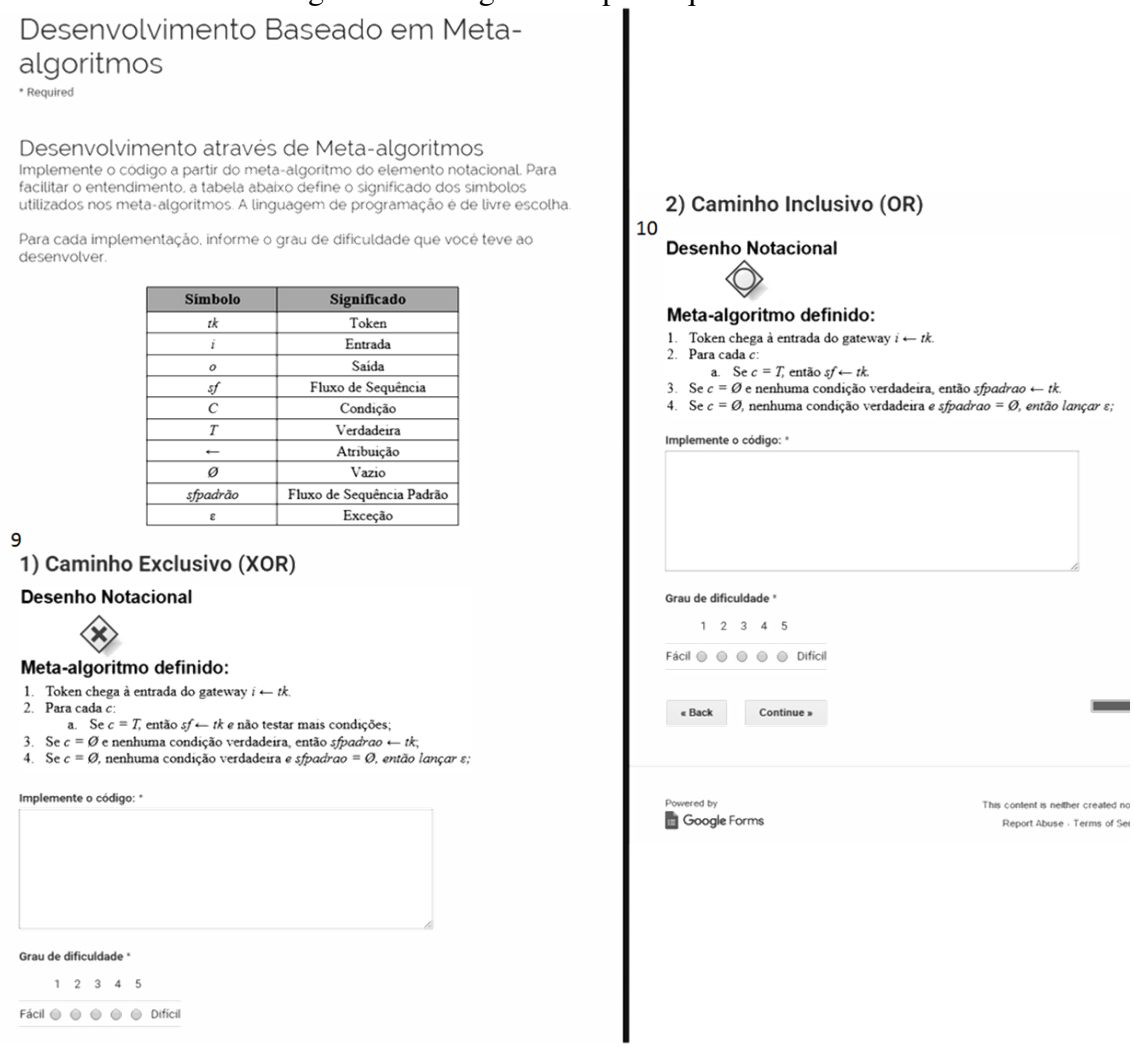
Para facilitar o entendimento sobre os meta-algoritmos, foi adicionada a tabela referente aos correspondentes dos símbolos utilizados. Os campos eram obrigatórios, e ao terminar esta etapa, o usuário passava para a quarta e última.

Por fim, a última etapa visava capturar a satisfação do usuário com o desenvolvimento com meta-algoritmos. Para isso, a Figura 4.25 apresenta os campos presentes no formulário:

11 - *Qual das opções há mais detalhes (lógica do elemento) de implementação?*: nesta questão o participante informava qual das opções fornecia mais detalhes da lógica comportamental do elemento notacional, de acordo com a respectiva opinião: Desenvolvimento através da definição textual ou Desenvolvimento através de meta-algoritmos.

12 - *Os meta-algoritmos ajudaram a identificar mais detalhes de implementação?*:

Figura 4.23: Segunda etapa do questionário.



Fonte: dos Autores.

a partir da experiência com os meta-algoritmos, o participante informava se com seu uso foi possível identificar mais detalhes de implementação do elemento notacional, do que utilizando apenas da definição textual. Como respostas possíveis, o participante escolhia "sim" ou "não".

13 - *Você concorda que o uso de meta-algoritmos auxilia na implementação dos elementos notacionais?*: visava verificar se o participante aprova o uso dos meta-algoritmos na implementação dos elementos notacionais. Como respostas possíveis, o participante escolhia "sim" ou "não".

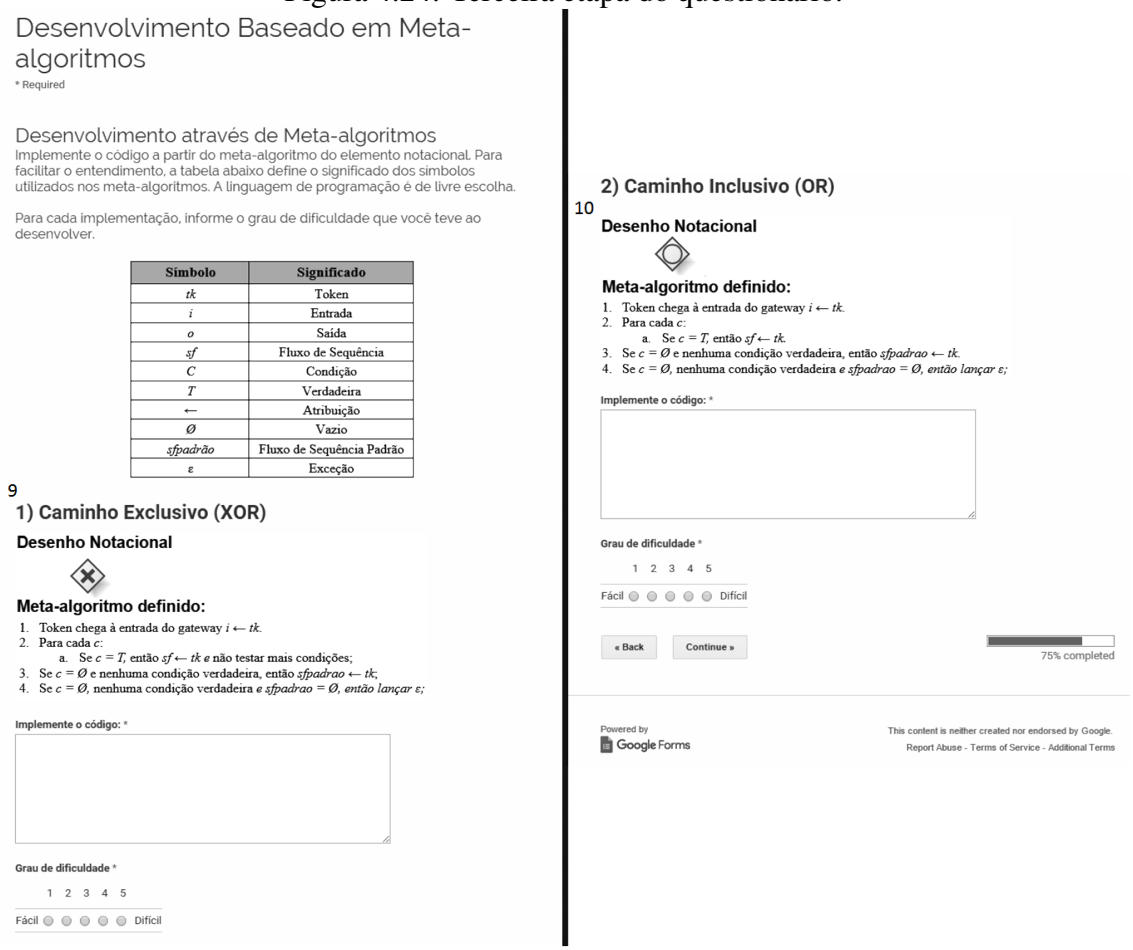
14 - *Grau geral de dificuldade de implementação dos códigos*: o participante informava o grau de dificuldade no desenvolvimento da lógica dos elementos notacionais, considerando o intervalo "1 - Fácil" à "5 - Difícil".

Ao finalizar essa etapa, o participante concluía a participação no questionário.

4.2.2 Resultados Obtidos

A partir da aplicação do questionário, foi obtida a participação dos usuários, que foram, distribuídos entre estudantes de graduação e pós-graduação de cursos da área da

Figura 4.24: Terceira etapa do questionário.



Fonte: dos Autores.

computação, desenvolvedores de *software*, professores, dentre outros. A Tabela 4.8 mostra a distribuição das profissões dos participantes envolvidos.

Tabela 4.8: Participantes por profissão.

Profissão	*Total*	*%*
Estudante	15	50,00
Professor	3	10,00
Desenvolvedor	8	26,67
Outra	4	13,33

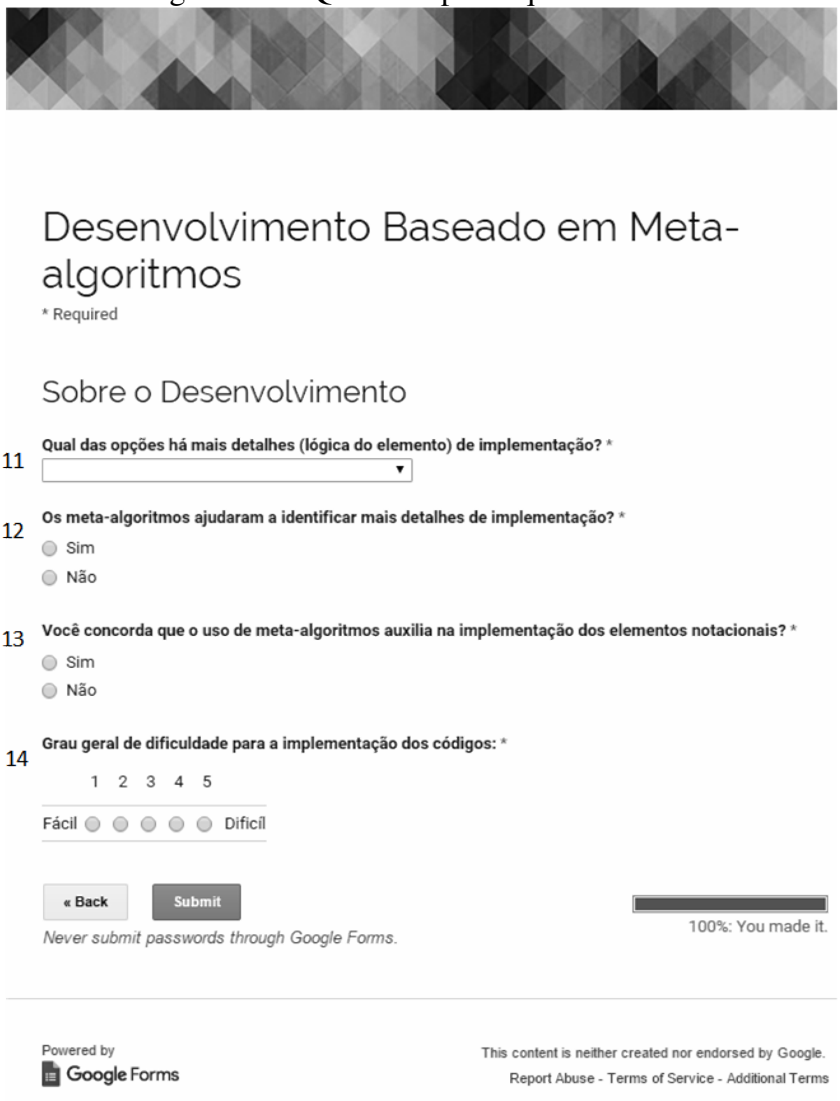
Fonte: Os Autores

Desse grupo total:

- 18 participantes possuíam o ensino superior incompleto;
- 8 participantes possuíam o ensino superior completo;
- 4 participantes possuíam pós-graduação (mestrado e/ou doutorado).

Metade dos participantes foram estudantes (de cursos como ciência da computação e engenharia de computação), seguido de desenvolvedores (26,67%) e outras profissões (4%). A Tabela 4.9 apresenta o conhecimento em desenvolvimento de *software* dos participantes.

Figura 4.25: Quarta etapa do questionário.



Fonte: dos Autores.

Considerando os 24 participantes que informaram possuir conhecimentos em desenvolvimento, foi analisado o tempo de experiência, informado na Tabela 4.10.

Considerando o período de experiência dos participantes, pode-se analisar que o questionário abrange indiretamente o público-alvo deste trabalho, visto que apesar da maioria ser estudante (15 participantes), 22 participantes possuem dois ou mais anos de experiência em desenvolvimento (10 participantes com experiência entre 2 e 4 anos e outros 12 com mais de 4 anos). A experiência deve se levada em consideração, visto que é necessária para o desenvolvimento da implementação.

Outras informação levantada dos participantes foi o conhecimento em modelagem de processos de negócio. A Tabela 4.11 ilustra os resultados.

Conforme apresentado a maioria dos participantes possuem conhecimentos em modelagem de processos de negócio, sendo destes 22 participantes, 15 possuem menos de dois anos de experiência, dois entre 2 e 4 anos e outros 5 apresentam mais de 4 anos de experiência. Neste caso a experiência do participante em modelagem de processos de negócio pode fazer com que o mesmo já tenha um conhecimento prévio do comportamento do elemento notacional, no momento de desenvolver a sua implementação.

Tabela 4.9: Conhecimento em desenvolvimento dos participantes.

Experiência	*Total*	*%*
Possui	24	80,00
Não possui	6	20,00

Fonte: Os Autores

Tabela 4.10: Experiência em desenvolvimento dos participantes.

Tempo	*Total*	*%*
Menos de 2 anos	2	8,33
Entre 2 e 4 anos	10	41,67
Mais de 4 anos	12	50,00

Fonte: Os Autores

Sobre a percepção dos participantes em relação aos meta-algoritmos, a Tabela 4.12 apresenta em qual abordagem há mais detalhes do que deve ser implementado de um elemento notacional, segundo a opinião dos participantes.

Para 19 participantes, a abordagem dos meta-algoritmos apresenta mais detalhes do que deve ser implementado, em relação a definição textual dos elementos notacionais. Isso mostra que esta abordagem pode facilitar a tarefa do usuário de identificar o que deve ser implementado, se comparado com apenas a utilização da definição textual.

Ao final foi verificado com os participantes a aceitação dos meta-algoritmos. Foi perguntado se concordavam com o uso dos meta-algoritmos como auxilio na implementação dos elementos notacionais. A Tabela 4.13 apresenta os resultados obtidos.

Para 22 participantes (73,33% do total) aprovam o uso dos meta-algoritmos. Isto mostra que o trabalho desenvolvido obteve uma boa aceitação, a partir deste público analisado.

A partir da aceitação dos usuários em relação ao uso dos meta-algoritmos, pôde-se alcançar um dos objetivos específicos deste trabalho, que era “evidenciar que com o uso dos meta-algoritmos, o público poderá ter um maior entendimento em relação as definições textuais dos elementos notacionais da BPMN”.

4.3 Discussão dos resultados

Este capítulo teve como foco aplicar técnicas que pudessem verificar e validar o trabalho desenvolvido. Buscou-se através destes, verificar se os objetivos específicos deste trabalho foram alcançados. Para isso, foi abordado uma verificação para demonstrar que os meta-algoritmos realmente estão aderentes à definição textual da especificação da BPMN e também uma validação da aceitação do usuário.

Na validação funcional dos meta-algoritmos, foi realizado um trabalho utilizando as tabelas de decisão e os grafos de fluxo de controle para mostrar que os meta-algoritmos estão aderentes às definições textuais dos elementos notacionais da BPMN. Como pode ser verificado na seção 4.1, de fato os meta-algoritmos estão aderentes, considerando a abordagem de testes aplicada neste trabalho.

Como limitações desta verificação, nem todos os meta-algoritmos puderam aplicar as tabelas de decisão, para aqueles que não tinham alguma estrutura condicional na definição textual. Apesar disso, buscou-se alguma alternativa para realizar essa validação verificação. Para isso, foi verificado se as regras que estavam na definição, de fato foram

Tabela 4.11: Conhecimento em modelagem de processos de negócio.

Informação	*Total*	*%*
Possui	22	73,33
Não possui	8	26,67

Fonte: Os Autores

Tabela 4.12: Abordagem que apresenta mais detalhes de implementação.

Abordagem	*Total*	*%*
Definição textual	11	36,67
Meta-algoritmos	19	63,33

Fonte: Os Autores

Tabela 4.13: Aceitação do uso dos meta-algoritmos.

Resposta	*Total*	*%*
Sim	22	73,33
Não	8	26,67

Fonte: Os Autores

aplicadas no meta-algoritmo.

Essa validação foi aplicada para validar um dos objetivos específicos deste trabalho que visava definir meta-algoritmos que fossem aderentes à definição textual da BPMN, considerando a especificação 2.0.2 de 2013.

Outra validação aplicada foi a aceitação do usuário. Para isso, foi feito uso de estrutura de questionário para proporcionar aos participantes interação com o trabalho desenvolvido, visando obter retorno quanto a satisfação com os mesmos.

A estrutura de questionário utilizada permitiu que os participantes pudessem interagir com os meta-algoritmos, utilizando como base para a implementação dos elementos notacionais, bem como capturar a satisfação com o uso.

A limitação do questionário aplicado se refere a participação baixa de desenvolvedores (26,67%), contudo se for levar em consideração o período de experiência em desenvolvimento de *software* de dois ou mais anos, têm-se um total de 91,67% dos participantes. Com isso, tinha-se um público que mesmo não sendo desenvolvedor profissional, dominava os conhecimentos de desenvolvimento, requisito para implementação dos meta-algoritmos.

Dos resultados obtidos através do questionário, destaca-se que os meta-algoritmos tiveram aceitação com o público, sendo que 63,33% acharam mais representativo que a definição textual e 73,33% aprovam o seu uso.

5 CONCLUSÃO

Neste capítulo, as considerações finais do trabalho são apresentadas, com ênfase nas contribuições nos resultados obtidos por este trabalho. Por fim, são expostas as perspectivas de trabalhos futuros.

Este trabalho forneceu contribuições sobre o incremento da codificação da BPMN. Foi levantado o problema motivador deste trabalho, a falta de aderência entre o XML e a definição do elemento notacional, apresentando um exemplo motivacional na seção 1.1. Neste contexto, uma abordagem que visa desenvolver uma lógica que seja mais aderente à definição textual dos elementos notacionais, presente na especificação 2.0.2 da BPMN, foi proposta. Tal abordagem foi chamada de *meta-algoritmo*. Para verificar os meta-algoritmos desenvolvidos, foi realizada a verificação funcional, através das Tabelas de Decisão e Grafos de Fluxo de Controle e a validação dos usuários, com a aplicação de um questionário, visando capturar a aceitação.

Desta forma, os seguintes objetivos específicos foram alcançados:

- *Propor meta-algoritmos que apresentem maior aderência às definições textuais dos elementos notacionais da BPMN, comparado com a codificação XML, considerando a especificação 2.0.2:* o desenvolvimento dos meta-algoritmos foi o caminho utilizado para desenvolver uma lógica com maior aderência às definições textuais. A verificação funcional, através da tabela de decisão que representou a definição textual, e o grafo de fluxo de controle que representou o meta-algoritmo, permitiu demonstrar que é possível desenvolver uma lógica mais aderente à regra textual, do que a codificação oferecida pela especificação, o XML.
- *Definir uma forma de verificar com o público alvo da especificação da notação, a aceitação dos meta-algoritmos propostos:* a forma definida foi a aplicação de um questionário, visando obter a satisfação dos usuários, totalizando 30 participantes.
- *Evidenciar que com o uso dos meta-algoritmos, o público poderá ter um maior entendimento em relação as definições textuais dos elementos notacionais da BPMN:* através do questionário aplicado, foi evidenciado que mais da maioria dos participantes afirmaram que os meta-algoritmos são mais representativos que a definição textual, totalizando 63,33%, além dos 73,33% dos participantes que aprovam seu uso.

Como principais contribuições deste trabalho, pode-se citar a definição de uma técnica que permita a identificação das regras inseridas, permitindo a extração do código para a definição dos meta-algoritmos. Seguindo os passos de (KARAKOSTAS, 1990), foi possível definir um meio de poder identificar as regras presentes na definição textual que poderiam ser aplicadas no meta-algoritmo, como por exemplo, a cada vez que a definição conter um “if”, trecho referido será considerado uma lógica no meta-algoritmo.

As limitações identificadas durante a realização deste trabalho tratam do desenvolvimento dos meta-algoritmos, na qual nem todas as regras puderam ser implementadas nos meta-algoritmos. Por exemplo, em algumas situações, a definição apresenta uma regra na implementação, porém não está definido o que fazer caso esta regra não ser seguida. Por conta disso, na etapa da verificação funcional, nem todas as regras puderam fazer uso das tabelas de decisão. Na validação com os usuários, poderia haver a participação de mais desenvolvedores de *software*, para verificar melhor a aceitação do público-alvo, apesar que do total de participantes, a maioria possui mais de 2 anos de experiência (totalizando 91,67%), o que poderia compensar a falta dos participantes de-

senvolvedores, visto que o requisito para o desenvolvimento dos meta-algoritmos é apenas o conhecimento em desenvolvimento de *software*.

As vantagens deste trabalho são, que a partir do momento em que se tem meta-algoritmos aderentes à definição textual dos elementos notacionais, é possível que os usuários que pretendem desenvolver um novo BPMS, possam ter um ponto de partida para o desenvolvimento da implementação dos elementos notacionais. Por já dispor da definição dos elementos notacionais apresentada na forma forma de algoritmo, essa poderia ser "traduzida" para a linguagem de programação a ser utilizada para o desenvolvimento de um BPMS (traduzir os meta-algoritmos para Java ou C#, por exemplo).

A partir do trabalho desenvolvido foram identificados novos horizontes de trabalho que, apesar de não ser aplicados nesta dissertação, ficam registrados com futuros trabalhos:

- O desenvolvimento de *variantes* dos meta-algoritmos, que contenham além das regras definidas na definição dos elementos notacionais, outras regras obtidas na literatura, como em (MENDLING; REIJERS; AALST, 2010). Com isso teria-se um conjunto de codificações cada qual com uma característica especifica (ex.: No meta-algoritmo do XOR, ter uma variante que trate do *deadlock*, por exemplo).
- Desenvolver meios de extrair de forma automática os trechos da codificação. Usar técnicas de linguagem natural ou semelhantes para que seja possível a partir de uma especificação, obter o respectivo código. Com isso elimina-se um possível viés de interpretação no desenvolvimento dos meta-algoritmos. Os primeiros passos foram dados no trabalho de Bombassaro (2015).

REFERÊNCIAS

AALST, W. M. P. v. d. Business Process Management: A Comprehensive Survey. **ISRN Software Engineering**, v. 2013, p. 1–37, 2013. ISSN 2090-7680.

AALST, W. M. P. v. d. et al. Workflow Patterns. p. 5–51, 2003.

AALST, W. V. D. **Process mining: discovery, conformance and enhancement of business processes**. 1. ed. [S.l.]: Springer Science & Business Media, 2011. ISBN 9783642193453.

BOMBASSARO, L. C. **Extração Assistida de Código XML a Partir da Descrição Textual de Elementos Notacionais da BPMN**. 40 p. Monografia (Graduação) — Universidade Federal do Rio Grande do Sul, 2015.

DEHNERT, J.; AALST, W. M. P. van der. Bridging the Gap between Business Models and Workflow Specifications. **Journal of Cooperative Information Systems**, p. 1–39, 2004.

DUMAS, M. et al. **Fundamentals of Business Process Management**. First. Berlin, Germany: Springer-Verlag, 2013.

DUMAS, M.; ROSA, M. L.; MENDLING, J. Understanding business process models: the costs and benefits of structuredness. **Advanced Information . . .**, p. 31–46, 2012.

EDWARDS, A. L. **Techniques of attitude scale construction**. [S.l.]: Ardent Media, 1983.

EID-SABBAGH, R.-H. et al. A Platform for Research on Process Model Collections. In: INTERNATIONAL WORKSHOP ON BUSINESS PROCESS MODEL AND NOTATION. **Proceedings...** Vienna, Austria: Springer, 2012. v. 125, n. 4, p. 8–22.

FABRA, J. et al. Automatic execution of business process models: Exploiting the benefits of Model-driven Engineering approaches. **Journal of Systems and Software**, Elsevier Inc., v. 85, n. 3, p. 607–625, 2012. ISSN 01641212.

FANTINATO, M.; GIMENES, I. M. S.; TOLEDO, M. B. F. Product line in the business process management domain. In: **Applied Software Product Line Engineering**. [S.l.]: Auerbach Publications, 2010. chp. 20, p. 497–530.

FIGL, K.; RECKER, J.; MENDLING, J. A study on the effects of routing symbol design on process model comprehension. **Decision Support Systems**, Elsevier B.V., v. 54, n. 2, p. 1104–1118, 2013.

GASSEN, J. B. et al. Business Process Modeling: Vocabulary Problem and Requirements Specification. In: INTERNATIONAL CONFERENCE ON THE DESIGN OF COMMUNICATION, Colorado Springs, USA. **Proceedings...** New York, USA: ACM, 2014. (SIGDOC '14), p. 1–10.

GROSSKOPF, A. An extended resource information layer for bpmn. **Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam**, 2007.

KARAGIANNIS, D. BPMS: business process management systems. **CM SIGOIS Bulletin**, v. 16, n. 1, p. 10–13, 1995.

KARAKOSTAS, V. Modelling and maintenance software systems at the teleological level. **Journal of Software Maintenance**, John Wiley & Sons, Inc., New York, NY, USA, v. 2, n. 1, p. 47–60, mar. 1990.

KHERBOUCHE, O. M.; AHMAD, A.; BASSON, H. Using model checking to control the structural errors in bpmn models. In: INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE. **Proceedings...** [S.l.]: IEEE, 2013. p. 1–12.

KITCHENHAM, B.; CHARTERS, S. **Procedures for Performing Systematic Reviews**. Department of Computer Science, Keele University, UK, 2007.

KOSSAK, F.; ILLIBAUER, C.; GEIST, V. Event-based gateways: Open questions and inconsistencies. In: INTERNATIONAL WORKSHOP, BPMN 2012, 2012, Vienna, Austria. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 53–67.

La Rosa, M. et al. Managing Process Model Complexity via Concrete Syntax Modifications. **IEEE Transactions on Industrial Informatics**, v. 7, n. 2, p. 255–265, 2011.

LIKERT, R. A technique for the measurement of attitudes. **Archives of psychology**, 1932.

MENDLING, J.; REIJERS, H. A.; AALST, W. M. P. v. d. Seven process modeling guidelines (7PMG). **Information and Software Technology**, v. 52, n. 2, p. 127–136, 2010.

NASCIMENTO, G. S. d. **Um Método para Descoberta Semi-Automática de Processos de Negócio Codificados em Sistemas Legados**. 109 p. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, 2014.

OMG. **Business Process Modeling Notation (BPMN), V. 2.0.2**. [S.l.], 2013.

PATTON, R. **Software testing**. 2. ed. Indianopolis: Sams Pub., 2006. ISBN 0672327988.

PEZZÈ, M.; YOUNG, M. **Teste e Análise de Software: Processos, Princípios e Técnicas**. Porto Alegre: Bookman, 2008. ISBN 9788577802623.

POELMANS, S.; REIJERS, H. A.; RECKER, J. Investigating the success of operational business process management systems. **Information Technology and Management**, v. 14, n. 4, p. 295–314, 2013.

PRESSMAN, R. **Engenharia de Software**. 6. ed. São Paulo: McGraw Hill Brasil, 2006. ISBN 8586804576.

PRIEGO-ROCHE, L. M. et al. Business process design from virtual organization intentional models. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING. **Proceedings...** Berlin, Heidelberg: Springer-Verlag, 2012. (CAiSE'12), p. 549–564.

REIJERS, H. A. Implementing BPM systems: the role of process orientation. **Business Process Management Journal**, v. 12, n. 4, p. 389–409, 2006.

SANTOS, C. F. H. **Uma Revisão Sistemática sobre Modelagem e Execução de Processos de Negócio**. 30 p. Monografia (Trabalho Individual) — Universidade Federal do Rio Grande do Sul, 2014.

SANTOS, C. F. H.; THOM, L. H.; FANTINATO, M. Investigating completeness of coding in business process model and notation. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS. **Proceedings...** Barcelona, Espanha: SCITEPRESS, 2015. v. 3, p. 328–333.

SILVA, E. et al. Especificação formal e verificação de workflows científicos. **IV e-Science**, Belo Horizonte, Minas Gerais, Brazil, v. 2, n. 1, p. 23–25, 2010.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 9788579361081.

STROPPI, L. J. R.; CHIOTTI, O.; VILLARREAL, P. D. Extending bpmn 2.0: Method and tool support. In: BUSINESS PROCESS MODEL AND NOTATION: THIRD INTERNATIONAL WORKSHOP. **Proceedings...** Berlin, Heidelberg: Springer, 2011. v. 95, p. 59–73.

THOM, L. H.; REICHERT, M.; IOCHPE, C. On the support of workflow activity patterns in process modeling tools: Purpose and requirements. **Workshop on Business Process Management**, 2009.

TRAN, H. et al. Modeling human aspects of business processes–a view-based, model-driven approach. In: SPRINGER. **Model Driven Architecture–Foundations and Applications**. [S.l.], 2008. p. 246–261.

WEBER, B.; SADIQ, S.; REICHERT, M. Beyond rigidity – dynamic process lifecycle support. **Computer Science - Research and Development**, v. 23, n. 2, p. 47–65, abr. 2009. ISSN 1865-2034.

WESKE, M. **Business Process Management**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN 978-3-642-28615-5.

WFMC. **Workflow Management Coalition Terminology & Glossary**. [S.l.], 1999.