UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

RODOLFO ANTONIAZZI

Abordagem para Analisar Problemas em Modelagem de Processos

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Profa. Dr. Lucineia Heloisa Thom
Co-orientador: Msc. Vinícius Stein Dani

Porto Alegre
2019

“Eu posso não ter ido para onde eu pretendia ir,
mas eu acho que acabei terminando onde eu pretendia estar.”
— Douglas Adams

AGRADECIMENTOS

Agradeço a minha família pelo apoio recebido durante toda a caminhada, desde o primeiro momento. Toda a vontade investida em mim finalmente promove resultados e tem um final.

Agradeço a colegas que colaboraram em todos os momentos do curso, apoiando nos momentos difíceis e comemorando nos bons momentos. Não será esquecido como as dificuldades podem unir colegas.

Também agradeço os professores que colaboraram para que esse momento chegasse, construindo conhecimento e apresentando sua sabedoria.

Por fim agradeço a minha orientadora pela sua paciência e solicitude, mostrando que a tarefa de um professor é muito mais do que a transfusão de conteúdo, é um literal orientação.

RESUMO

A partir da análise de problemas comuns relacionados a modelagem de processos de negócios foram consideradas formas de detectar problemas recorrentes que podem ser avaliados através da análise do arquivo BPMN. Além disso foi construída uma avaliação do estado da arte e a relevância do projeto em meio ao cenário de ferramentas de modelagem de processo atual. Os problemas abordados são referentes a indicação de pesquisas quanto a problemas relacionados a criação e manutenção de processos de negócios que podem dificultar o entendimento e a execução correta dos mesmos. A validação foi realizada comparando as ferramentas atuais de modelagem de processos quanto a capacidade das mesmas de detectar e avisar ao usuário quanto aos problemas em questão. A partir do estudo foram detectados os problemas que não são corrigidos pelos softwares de modelagem de processos de negócio atuais e foi criado um analisador que é capaz de detectar e avisar ao usuário quanto aos problemas tendo como entrada um arquivo BPMN.

Palavras-chave: Analisador de Modelos de Processo. Problemas em Modelagem de Processos. Business Process Management. Business Process Model and Notation.

Process Model Analyzer

ABSTRACT

Beginning with the review of common problems related to modeling business process, ways of detecting recurring problems in BPMN files were created. After selecting the common problems a study in the state of art to understand the relevancy of this work inside the BPMN modeler tools market. The selected problems were made using researches related to common problems in creating and maintaining business process, problems that could harden the correct understanding and execution of the processes. These problems were validated comparing the current process modeler tools in its capacity of detecting and alerting the problems to the users. It was possible to detect which problems were not usually solved by the current process modeler tools available in the market and it was created an analyzer witch could detect and warn the user about each one of these problems, using as an input an BPMN file.

Keywords: Process Model Problems, Business Process Management, Business Process Model and Notation.

LISTA DE ABREVIATURAS E SIGLAS

BPM	Business Process Management
BPMN	Business Process Model and Notation
OMG	Object Management Group
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
XOR	Operador lógico de disjunção exclusiva
AND	Operador lógico de conjunção
OR	Operador lógico de disjunção
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Um *processo de negócio* (também denominado *processo* nesse documento) é uma cadeia de *eventos*, *atividades* e *decisões* que possibilita a uma empresa organizar de forma mais eficiente sua produção e serviços. O Gerenciamento de Processos de Negócio (*Business Process Management - BPM*) é um processo complexo principalmente pelo aspecto multidisciplinar que o caracteriza e necessita de constante manutenção(DUMAS et al., 2013). Considerando está complexidade é notável a dificuldade de criar e manter processos corretos e compreensíveis utilizando as ferramentas de modelagem de processos atuais.

A Notação e Modelo de Processos de Negócio (*Business Process Model and Notation - BPMN*) é uma notação voltada a padronização dos modelos de *processos*, utilizada para facilitar o seu entendimento, e atualmente está em sua segunda versão. A notação é especificada pelo *Object Management Group* (https://www. Omg. Org/) e é homologada pelo ISO/IEC 19510:2013 conforme referencia(GROUP, 2013).

A necessidade de *processos* bem estruturados é vital em pequenas e médias empresas, as quais não possuem grande quantidade de colaboradores e dessa forma os mesmos devem realizar várias tarefas distintas (BAZHENOVA; TARATUKHIN; BECKER, 2012). Apesar disso, pequenas e médias empresas não possuem setores dedicados para a criação, alteração e manutenção de *processos* e por esse motivo tendem a ter dificuldade com as ferramentas de modelagem de *processos*, tal como o BPMN(BAZHENOVA; TARATUKHIN; BECKER, 2012).

A criação dos *processos* pode ser prejudicada pela falta de conhecimento dos funcionários em razão do desconhecimento sobre as boas práticas de modelagem(MENDLING; REIJERS; AALST, 2008). Por sua vez, a menor quantidade de funcionários também gera dificuldades na manutenção dos *processos* uma vez que não existem funcionários dedicados a realizar essa tarefa periodicamente (BAZHENOVA; TARATUKHIN; BECKER, 2012).

A substituição de funcionários pode ser outro problema para a compreensão de *processos* nas pequenas e médias empresas, pois *atividades* podem ser executadas por apenas um colaborador. Dessa forma, é ainda mais importante para organizações menores que os seus *processos* sejam criados com *sintaxe* e *semântica* correta, pois novos colaboradores devem conseguir realizar os *processos* sem interrupções desnecessárias.

Uma das abordagens atuais para esse problema é o treino dos funcionários envol-

vidos com os *processos* de negócio. Essa abordagem não é completa pois os *processos* tem propriedades inerentes que os tornam complexos e irão prejudicar o entendimento mesmo quando as regras sejam compreendidas(AVILA, 2018). Outra abordagem possível seria a simplificação do *processo* para facilitar o seu entendimento. Apesar disso essa simplificação pode ser prejudicial ao resultado do negócio(KROGSTIE, 2012).

Para minimizar esse problema de uma forma distinta, considerando a necessidade de uma modelagem BPMN correta e de conhecermos problemas recorrentes do BPMN(ROZMAN; POLANčIč; HORVAT, 2008), é possível criar um analisador sintático BPMN que avalie os *processos* indicando problemas e avisos (*warnings*) no *processo* avaliado. Esse analisador deve receber como entrada um arquivo BPMN e indicar em sua saída pontos que compõe problemas sintáticos, semânticos ou pragmáticos do processo(ROZMAN; POLANčIč; HORVAT, 2008).

A criação do analisador de forma independente de *softwares* modeladores de projeto, utilizando como entrada somente um arquivo BPMN, também indica uma possível futura adaptação do analisador para mais de um *software* modelador a partir de *plug-ins* que podem ser programados para os mesmos. Além disso, uma análise da capacidade de detectar e corrigir problemas dos *softwares* modeladores foi realizada para indicar o estado atual dos *softwares* atuais comparados ao analisador desenvolvido.

Nesse caso, para cada problema analisado foi feita uma análise de qual forma os *softwares* atuais detectam e se os mesmo são capazes de avisar os problemas cometidos. Essa foi utilizada para demonstrar a necessidade de um analisador mais completo e estruturado que possa detectar os problemas considerados.

O resultado final deste trabalho é um analisador de modelos de *processo* capaz de detectar uma série de problemas em *processos* BPMN. Além disso também será apresentada uma comparação do analisador com o estado da arte atual nas ferramentas de modelagem de *processos*. Esse resultado foi criado de forma a possibilitar possíveis ampliações do projeto, como por exemplo a criação de *plug-ins* para sua utilização nos *softwares* modeladores de processos ou a possibilidade de corrigir outros problemas conhecidos através de uma expansão nas capacidades do projeto.

O próximo capítulo traz os fundamentos necessários para o entendimento de um *processo* BPMN e de alguns problemas comuns propostos no capítulo 3. O capítulo 3 identifica os principais problemas e informa possibilidades de resolução dos mesmos, indicando estruturas e estruturas de dados a serem utilizados para a resolução. O capítulo 4 apresenta o estado da arte comparando os principais *softwares* modeladores com o anali-

sador sintático formulado nesse trabalho. O capítulo 5 apresenta as estruturas de dados e o funcionamento do analisador sintático. O capítulo 6 discute os resultados obtidos e traz as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

O BPMN é uma notação gráfica que permite a representação e modelagem de *processos de negócio*(DUMAS et al., 2013). Essa notação permite melhor acompanhamento dos *processos*, evoluindo de forma a simplificar o entendimento e aumentar a compreensibilidade das ações a serem tomadas pelos funcionários das organizações. Resumidamente, o *BPM* é considerada a arte e ciência de observar com o trabalho é realizado em uma corporação para garantir resultados consistentes(DUMAS et al., 2013).

Para o correto entendimento dos problemas sintáticos, semânticos e pragmáticos é necessário que descrevamos detalhadamente as regras do BPMN(GROUP, 2013), além das melhores práticas que também são importantes para a compreensão correta do processo(MENDLING; REIJERS; AALST, 2008).

A partir da caracterização de um *processo* como uma cadeia de *eventos*, *atividades* e *decisões*, podemos também descrever cada um dos nodos citados separadamente. Em (DUMAS et al., 2013) *Eventos* são definidos como atuações normalmente externas e atemporais, que podem alterar o curso do processo. Exemplos são o recebimento de um e-mail ou de uma ligação e a aceitação ou rejeição de uma proposta ou ordem de compra. No BPMN eles são representados por um círculo e podem ser responsáveis pelo início (com borda mais fina) ou final (com borda mais grossa) de um processo. O círculo pode conter um envelope no seu centro representando o recebimento ou o envio de uma mensagem. As melhores práticas indicam a necessidade de um título adequado, preferencialmente possuindo um verbo, descrevendo cada um dos *eventos* do processo(MENDLING; REIJERS; AALST, 2008).

Nodos de *atividade* descrevem trabalhos que podem ser realizados por humanos ou *softwares* ou até uma combinação deles. Ao contrário dos *eventos* as *atividades* podem ter uma duração (definida ou não). *Atividades* que tem uma duração curta e podem ser feitas de uma só vez são chamadas também de tarefas. As *atividades* são representadas no BPMN como retângulos sem pontas e preferencialmente devem ser acompanhadas de um título descrevendo a atividade.

As *decisões* de um *processo* BPMN são caracterizadas por um nodo de *Gateway*, no formato de um diamante. Esse nodo pode possuir várias saídas ou várias entradas, dividindo ou unindo o processo. Os *gateways* são divididos em três diferentes tipos que serão discutidos ainda nesse capítulo. Eles representam uma alteração na execução do *processo* que pode levar a diferentes *atividades*. Os *gateways* de divisão possuem apenas

uma entrada e várias saídas, enquanto os *gateways* de união possuem várias entradas e apenas uma saída. A *atividade* anterior ao *gateway* é responsável por definir a pergunta que irá guiar a decisão. Cada uma das saídas de um *gateway* de divisão deve possuir um título que representa a decisão tomada.

Todos esses nodos são interligados por fluxos, representados por fechas com a cabeça preenchida. Os fluxos mostram para os leitores qual a ordem de execução do processo, iniciando no *evento* de início e terminando no *evento* de fim. De forma unitária, os fluxos determinam os caminhos pelos quais os *tokens* devem seguir durante o processo.

Por sua vez, os *tokens* são um importante recurso de análise do BPMN pois representam uma instância do processo. Esse recurso é utilizado para avaliar a continuidade dos *processos*, observando se o mesmo pode ir do início até o final e quais os caminhos são possíveis para isso. Essa análise se torna ainda mais importante quando procuramos *deadlocks* já que *tokens* que nunca finalizam o *processo* podem nos ajudar a encontrar os problemas.

A notação de *tokens* no BPMN são pequenos círculos coloridos preenchidos e móveis marcados em uma das partes do processo. É importante salientar que os *tokens* não fazem parte da criação do processo, e sim dos testes e da execução, porém são importantes para a fundamentação teórica dos tipos de *gateway*. Poderíamos interpretar os mesmos como variáveis de estado que guardam qual parte do *processo* está sendo realizada e por isso os nodos de *gateway* alteram o fluxo dos tokens.

O primeiro tipo de *gateway* é o tipo *XOR*, utilizado para *decisões* exclusivas. Esse *gateway* é o mais simples pois quando um *token* (instância do processo) chega em um *gateway XOR* divisor, ele sai por uma das opções de saída conforme a tomada de decisão. Da mesma forma, quando um *token* é recebido por alguma das entradas de um *gateway XOR* unificador ele é enviado diretamente para a saída. Esse *gateway* é representado pela letra X grafada dentro do diamante e deve ter suas condições de saída mutuamente exclusivas para que somente uma seja verdadeira por token.

O segundo tipo de *gateway* é o tipo *AND*, utilizado para *atividades* paralelas. Quando um *token* encontra um *gateway AND* divisor ele é dividido em mais *tokens* da mesma instância (representados pela mesma cor) de forma que um *token* saia por cada saída do *gateway*. Dessa forma as *atividades* entre um *gateway AND* divisor e outro *gateway AND* unificador são realizadas paralelamente. É importante ressaltar que um *gateway AND* unificador só irá enviar um *token* pela sua saída quando receber um *token* de cada uma das suas entradas. Alguns *processos* resultam em *deadlock* a partir do *gateway*

AND unificador. O gateway por um problema de desenho do *processo* pode nunca receber um *token* de um dos lados e consequentemente o processo nunca irá continuar(DUMAS et al., 2013). Esse tipo de *gateway* é representado no BPMN por um símbolo + grafado dentro do diamante.

O último tipo de *gateway* desse trabalho é o *gateway* OR. Um *gateway OR* divisor ao receber um *token* irá enviar um *token* da mesma instância (representado pela mesma cor) para cada saída correta, podendo dividir ou não o token. Da mesma forma um *gateway OR* unificador irá esperar por um *token* por cada caminho ativo até ele. Dessa forma a utilização de um *gateway OR* no lugar de um *gateway AND* pode evitar *deadlock*. Apesar disso, a utilização desse *gateway* é complexa, principalmente no sentido de *software* e hardware e por isso deve ser evitada(DUMAS et al., 2013). A notação para esse tipo de *gateway* é um diamante com um círculo grafado na sua parte interna.

Por fim existem dois elementos notacionais que não são nodos e não são necessários para o funcionamento da BPMN (sintaticamente) porém são extremamente recomendados. A primeira notação é representada por uma folha com o canto dobrado e representa um documento que é resultado do processo. O documento é conectado ao BPMN através de uma flecha pontilhada e o sentido da flecha representa se o mesmo é utilizado na *atividade* conectada ou é um produto dessa atividade.

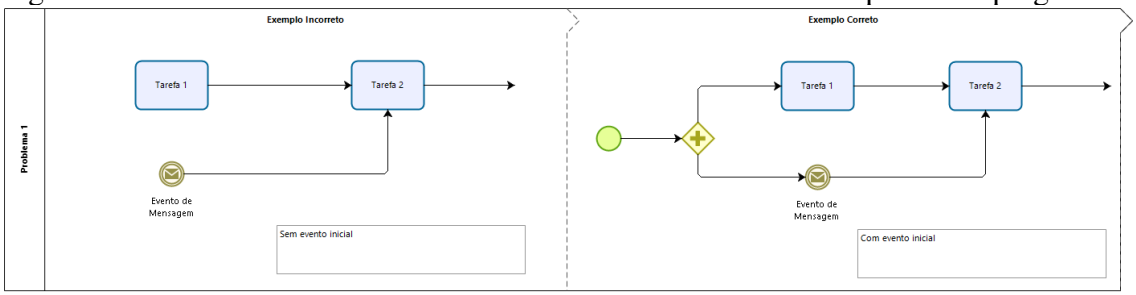
O segundo elemento notacional é utilizado para arquivamento de dados e é necessário para dados persistentes que necessitam durar um tempo maior do que o processo. Normalmente esse tipo de dado é armazenado por *software* em um banco de dados da empresa. Também é conectado a BPMN através da mesma flecha pontilhada usada nos documentos, porém são representados por um cilindro três linhas na borda superior.

A partir dos conhecimentos iniciais de BPMN é possível dividir os problemas de modelagem estudados e dispostos a seguir em três tipos, sendo eles sintáticos (problemas na utilização das regras do BPMN), semânticos (problemas no entendimento da funcionalidade dos nodos BPMN) ou pragmáticos (geram problemas graves de compreensão do *processo* mesmo não sendo problemas sintáticos)(ROZMAN; POLANčIč; HORVAT, 2008). Essa divisão é relevante para a seleção e avaliação dos problemas mais importantes a serem corrigidos.

3 ABORDAGEM PARA CORREÇÃO DE PROBLEMAS EM MODELAGEM DE PROCESSOS

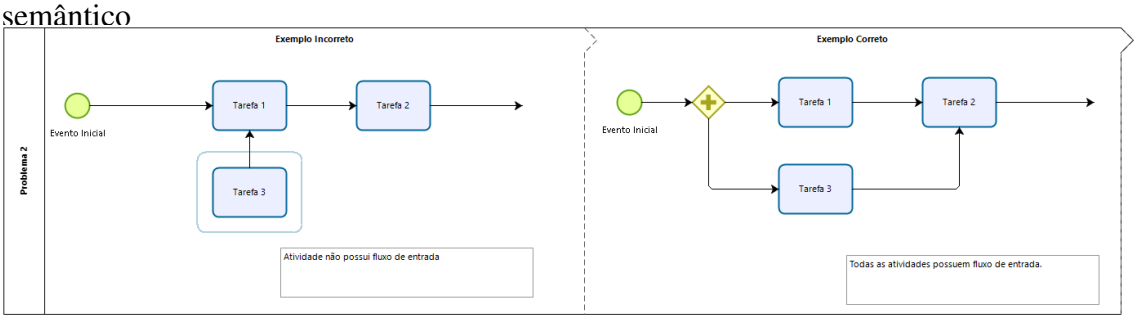
A partir da fundamentação teórica descrita no capítulo 2 e do artigo (ROZMAN; POLANčIč; HORVAT, 2008) foram definidos alguns dos problemas de modelagem mais comuns ocorridos no BPMN. Esses problemas foram organizados considerando a sua relevância e o tempo necessário para resolver o mesmo. O trabalho inicia com o problema mais trivial (porém ainda bastante relevante) e finaliza com o problema mais complexo. A seguinte ordem de problemas de modelagem foi estabelecida.

Figura 3.1: Problema 1 - O *Processo* não contém um *evento* inicial - problema pragmático



Fonte: O Autor baseado em (ROZMAN; POLANčIč; HORVAT, 2008)

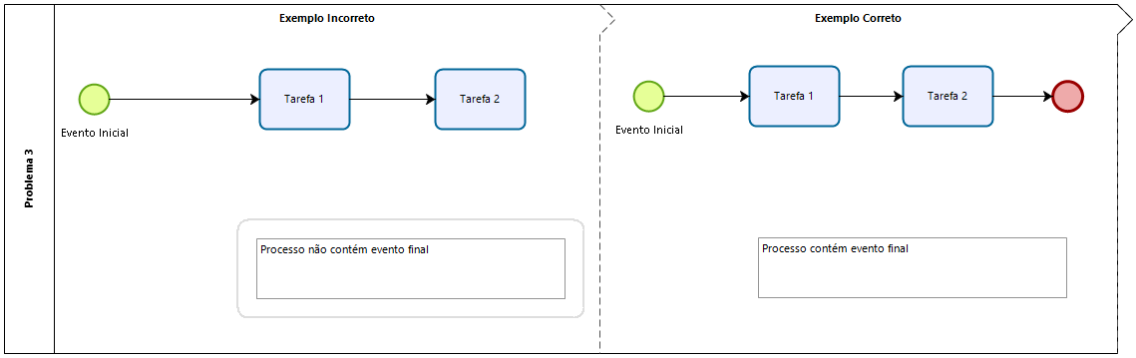
Figura 3.2: Problema 2 - *Atividade* ou *evento* não possui um fluxo de entrada - problema semântico



Fonte: O Autor baseado em (ROZMAN; POLANčIč; HORVAT, 2008)

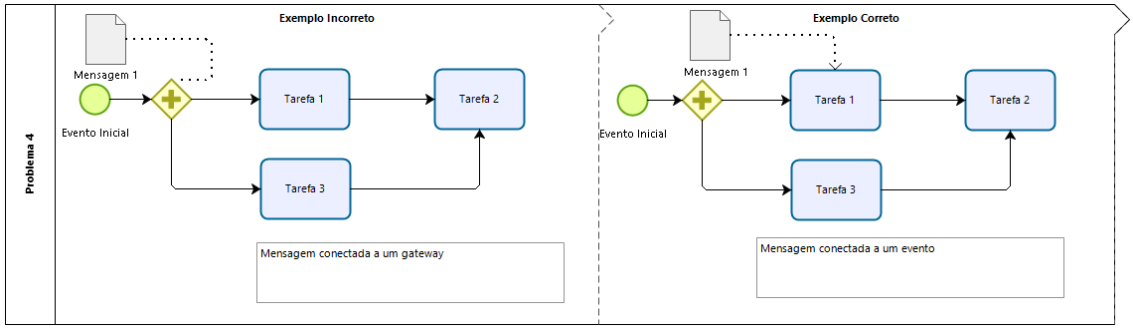
Possibilidade de *deadlock* durante *processo* - problema pragmático Os problemas citados causam grandes dificuldades na realização do processo. Um *processo* que não contém um *evento* inicial pode nunca ser iniciado, por exemplo(ROZMAN; POLANčIč; HORVAT, 2008). Para a correção desses problemas iremos utilizar como entrada um arquivo .BPMN, arquivo na linguagem de marcação XML. Esse arquivo descreve todas as *atividades*, *eventos*, gateways, fluxos e dados criados pelo modelador do *processo* com suas respectivas conexões e podem ser exportados a partir dos *softwares* de modelagem BPMN

Figura 3.3: Problema 3 - O *processo* não contém um *evento* final - problema pragmático



Fonte: O Autor baseado em (ROZMAN; POLANčIč; HORVAT, 2008)

Figura 3.4: Problema 4 - Um *gateway* recebe ou gera mensagens - problema sintático e semântico



Fonte: O Autor baseado em (ROZMAN; POLANčIč; HORVAT, 2008)

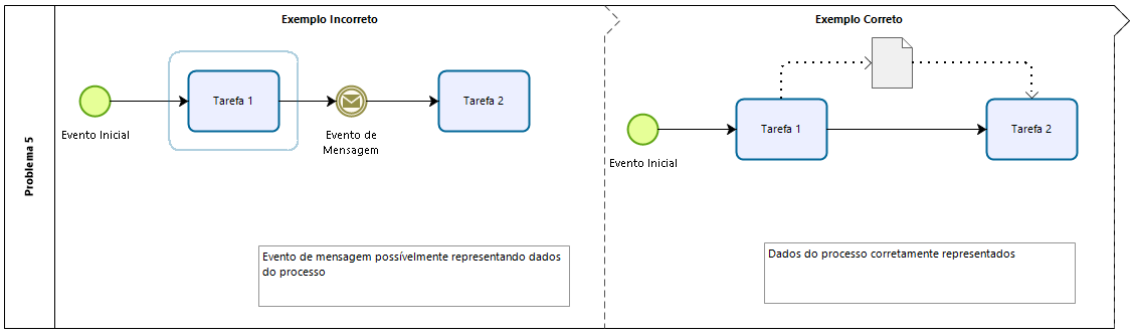
(como por exemplo os softwares Bizagi, Bonita, Camunda e Signavio).

3.1 Especificações do Analisador de Processo

A leitura do arquivo .BPMN é programada na linguagem de programação C++, pois essa linguagem é orientada a objetos, de execução rápida(PRECHELT, 2000) e já conhecida pelo orientando. O objetivo é que o analisador possa ser utilizado de forma rápida mesmo em *processos* com muitos elementos notacionais. Os elementos serão populados em objetos que facilitam a análise pois podem futuramente ser organizados nas estruturas de dados utilizadas. Os objetos serão criados conforme a seguinte descrição:

1. Process: id, name
2. StartEvent: id, name, incoming, outgoing
3. Task: id, name, incoming, outgoing
4. SequenceFlow: id, sourceRef, targetRef, name
5. ParallelGateway: id, gatewayDirection, incoming, outgoing

Figura 3.5: Problema 5 - *Eventos* de mensagem representando dados do *processo* - problema semântico



Fonte: O Autor baseado em (ROZMAN; POLANčIč; HORVAT, 2008)

6. ExclusiveGateway: id, gatewayDirection, incoming, outgoing
7. InclusiveGateway: id, gatewayDirection, incoming, outgoing
8. EndEvent: id, name, incoming, outgoing
9. Event: id, name, incoming, outgoing
10. Data: id, name, incoming, outgoing
11. MessageFlow: id, sourceRef, targetRef, name
12. IntermédiateThrowEvent: id, name, incoming, outgoing

Os objetos criados possibilitam a análise do *processo* na forma de um grafo, já que essa é a forma real do BPMN(DUMAS et al., 2013). *Atividades*, *eventos*, *gateways* e dados são considerados nodos (vértices) enquanto fluxos sequenciais e fluxos de mensagens são considerados arestas. Será possível percorrer o grafo a partir de um nodo sem dados no atributo *incoming* para descobrir qual é o primeiro nodo do grafo. No caso de encontrarmos mais de um atributo sem dados no atributo *incoming*, podemos observar o tipo do objeto e detectar se temos dois *eventos* iniciais (um problema sintático) ou uma *atividade* ou *evento* sem fluxo de entrada (segundo problema a ser resolvido). Na figura 4.2 podemos ver que a *atividade* chamada de *"Atividade 3"* não possui fluxo de entrada e por isso nunca será executada no processo(ROZMAN; POLANčIč; HORVAT, 2008).

Além do grafo são criadas listas encadeadas por tipo de objeto, que referenciam o objeto no grafo, facilitando assim encontrar alguns dos problemas sem excessivo acréscimo a memória. Uma lista encadeada de *StartEvents* vazia representa o primeiro problema que deve ser resolvido, como podemos ver na figura 3.1, onde o *processo* não apresenta *evento* de início e pode nunca ser iniciado(ROZMAN; POLANčIč; HORVAT, 2008). A mesma teoria também pode detectar o terceiro problema da lista acima, como podemos ver pela figura referente ao problema 3, onde o *processo* não possui *evento* final

e por isso uma instância pode nunca ser considerada como finalizada(ROZMAN; POLANčIč; HORVAT, 2008).

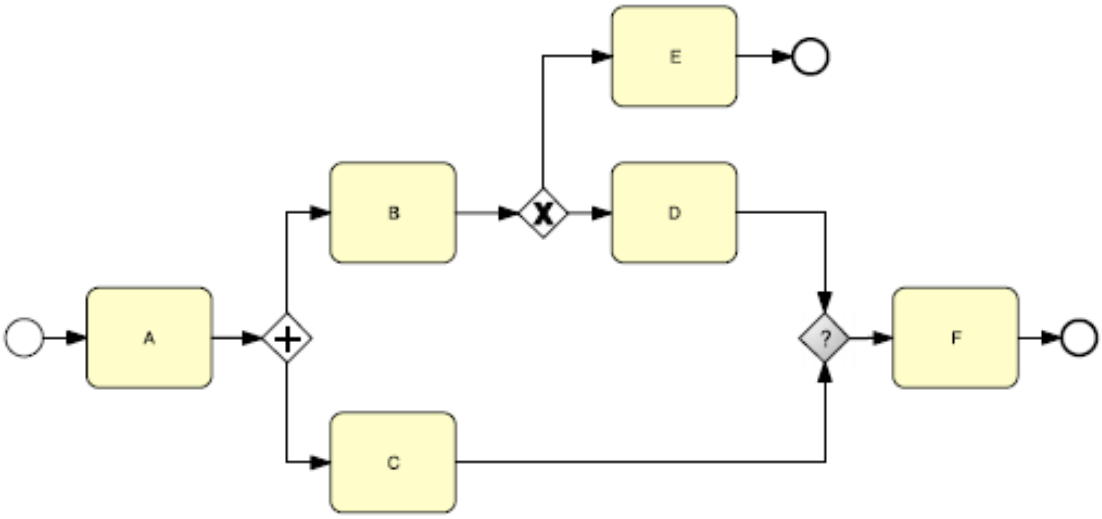
Para a identificação do quarto problema da lista é necessário detectar se a saída de um *gateway* do BPMN é realizada a partir de um data *flow*, ou seja, uma flecha pontilhada que ligaria o *gateway* a um dado. Podemos ver na figura referente ao problema 4 que as flechas representando data *flows* se conectam diretamente ao *gateway*, de forma que os dados gerados pelo *processo* possam nunca ser criados(ROZMAN; POLANčIč; HORVAT, 2008). O quinto problema pode ser detectado analisando se o *evento* de mensagem em questão não possui valores dentro do atributo *DataFlow* e possui uma entrada a partir de um *SequenceFlow* pois nesse caso o *processo* está utilizando o *Evento* de Mensagem como dado criado a partir da ferramenta de modelagem, como podemos observar na figura 3.5, e não como um *evento* que necessita o recebimento ou envio de uma mensagem para continuar. Esse problema pode gerar a compreensão de que o *processo* necessita receber um dado quando na verdade o dado deve ser criado pela atividade, fazendo com que o *processo* pare(ROZMAN; POLANčIč; HORVAT, 2008).

O sexto e último problema necessita de uma avaliação a partir de *tokens* para sua detecção. É necessária a criação de uma série de *tokens* que possam percorrer todos os caminhos do *processo*. Todos devem começar no *evento* de início e terminar no *evento* de fim. A impossibilidade de um *token* chegar ao *evento* de fim significa a presença de um *deadlock* no processo.

Na figura 3.6, caso o *gateway* com ponto de interrogação seja substituído por um *gateway* unificador para *atividades* paralelas (AND), quando um *token* for recebido pelo *gateway* da *atividade* 'C' ele será obrigado a esperar pelo *token* da *atividade* 'D', que pode nunca chegar pois o *token* pode ter acabado o *processo* a partir da *atividade* 'E'. Dessa forma uma parte de um *token* ficará trancado no *gateway AND* enquanto outra parte do mesmo *token* terá acabado o *processo* sem chance de se unir a outra metade.

Substituindo o mesmo por um *gateway* unificador *OR* o exemplo se torna completo pois no momento que uma parte do *token* vai para a *atividade* 'E' ele não é mais esperado pelo gateway, que envia o seu *token* para o próximo elemento do processo. Os *gateways OR* devem ser evitados pela complexidade que eles concedem ao *processo*. Porém algumas utilizações são necessárias como no da figura 3.6 onde sem o gateway correto o processo possuiria *deadlock*(DUMAS et al., 2013).

Figura 3.6: Problema 6 - Exemplo de *deadlock* em um *processo* - problema número 6



Fonte: O Autor baseado em (ROZMAN; POLANčIč; HORVAT, 2008)

Após a detecção de cada problema é criada uma mensagem de saída de problema do C++ para cada caso. A partir dessa mensagem é comparado como o analisador sintático criado para esse trabalho se equipara com os *softwares* de modelagem utilizados atualmente. Essa análise é importante para indicar a relevância da abordagem e verificar se está compatível com o estado da arte, podendo dessa forma ser futuramente utilizado como *plug-in* das ferramentas de modelagem.

4 AVALIAÇÃO DAS FERRAMENTAS DE MODELAGEM DO MERCADO

Para entender a relevância deste trabalho para o mercado atual de ferramentas de modelagem, é necessário testar as ferramentas atuais e avaliar como elas se comportam quando apresentados os problemas que foram considerados nesse trabalho.

É importante ressaltar que este trabalho não se propõe a comparar as ferramentas para avaliar a que se encontra mais adequada a resolver os problemas, de forma comparativa como já foi realizado em trabalhos anteriores (DIAS, 2018). A intenção desse capítulo é entender a habilidade dos softwares modeladores atuais em geral de trabalhar com esses problemas.

Dessa forma o que é avaliado nesse trabalho é a média do resultado entre as ferramentas de modelagem, de forma que possamos entender se a maioria das ferramentas atuais tem a capacidade de detectar os problemas citados no capítulo 3 ou se a maioria deles falha nessa funcionalidade.

A partir da avaliação das ferramentas de modelagem existentes foram selecionados 4 softwares para testes dos atributos que compõe este trabalho. Os softwares foram selecionados baseados em recomendação da orientadora e utilização prévia do orientando. As quatro ferramentas selecionadas foram, em ordem alfabética, Bizagi Modeler (BIZAGI, 2019), Bonita Studio (S.A., 2019), Camunda Modeler (GMBH, 2019) e Signavio Process Manager (SIGNAVIO, 2019).

As ferramentas foram avaliadas conforme suas características específicas, de forma que as modelagens de entrada para os mesmos casos podem ser diferentes. As ferramentas de modelagem detectam diferentes níveis de problemas (como poderá ser confirmado posteriormente) e assim mensagens não compatíveis de problema podem ser apresentadas em algumas situações, nas quais é necessário alterar a modelagem do processo. Além disso, os problemas e avisos alertados pelas ferramentas são bastante diferentes entre eles e por isso foram definidos três níveis de correção.

Um aviso completamente incorreto indica que o software não avisou que existe nenhum problema na modelagem. Um aviso parcialmente correto indica que a ferramenta avisou que existe algum problema na modelagem porém não conseguiu detectar o problema específico. Um aviso correto indica que o software indicou o problema correto da modelagem.

4.1 Avaliação do Problema 1

Iniciamos avaliando o primeiro problema, o processo não contém um evento inicial. Por ser um problema pragmático, a detecção desse problema não é obrigatória, já que podem existir modelos de processos sem eventos iniciais pelas normas do BPMN(DUMAS et al., 2013). Apesar disso, como já foi discutido no capítulo 3, é fortemente recomendado que todos os processos se iniciem por um evento de inicio (ROZMAN; POLANčIč; HORVAT, 2008).

Considerando isso, foram definidas ferramentas com avisos corretos somente os softwares que apresentam uma mensagem adequada indicando a falta de um evento de início (DANI, 2019), normalmente apresentado como *warning*, ou em português, aviso.

Algumas ferramentas apresentaram um problema no caminho padrão de um processo, ou seja, apresentaram um aviso de que o processo não começa em um evento de início e acaba em um evento de final. Porém, essa abordagem foi considerada parcialmente correta pois indica a falta do evento de início sem avisar corretamente ao usuário o que exatamente está faltando, mostrando mais de uma opção e não sendo completamente clara.

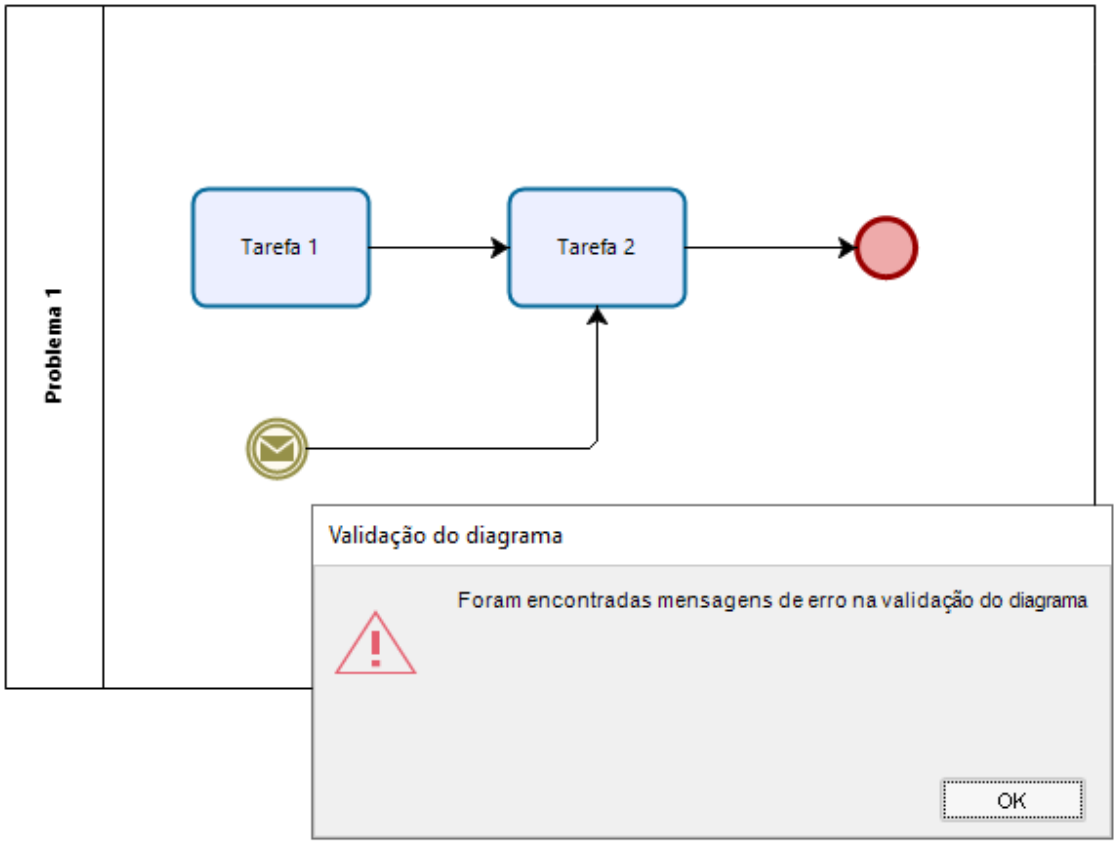
A ferramenta Bizagi apresentou um aviso incorreto no problema 1, conforme apresentado nas imagens abaixo.

Como é possível avaliar na figura 4.1, foi detectado um problema no processo avaliado, problema esse que realmente está presente. A ferramenta Bizagi ainda possui partes não traduzidas, o que se apresenta no problema da figura 4.2, apesar de grande parte do software se encontrar em português.

Quando observamos a indicação do problema na figura 4.2, podemos traduzir a mesma em tradução livre para "O evento deveria ter um fluxo de entrada.", o que é uma avaliação correta, porém não representa o problema que esperamos que seja detectado aqui. A avaliação da ferramenta é extremamente simples e não indica que um aviso de que é possível que o problema para esse processo possa ser a falta de um evento inicial.

A ferramenta nem mesmo apresenta um aviso sobre a Tarefa 1, que da mesma forma que o evento de mensagens na figura 4.1, não apresenta fluxo de entrada. Sendo assim a ferramenta avisa o problema sintático, porém não avisa o usuário da falta de evento

Figura 4.1: Problema 1 - Ferramenta Bizagi - Apresentação de problema.



Fonte: O Autor

Figura 4.2: Problema 1 - Ferramenta Bizagi - Indicação do problema

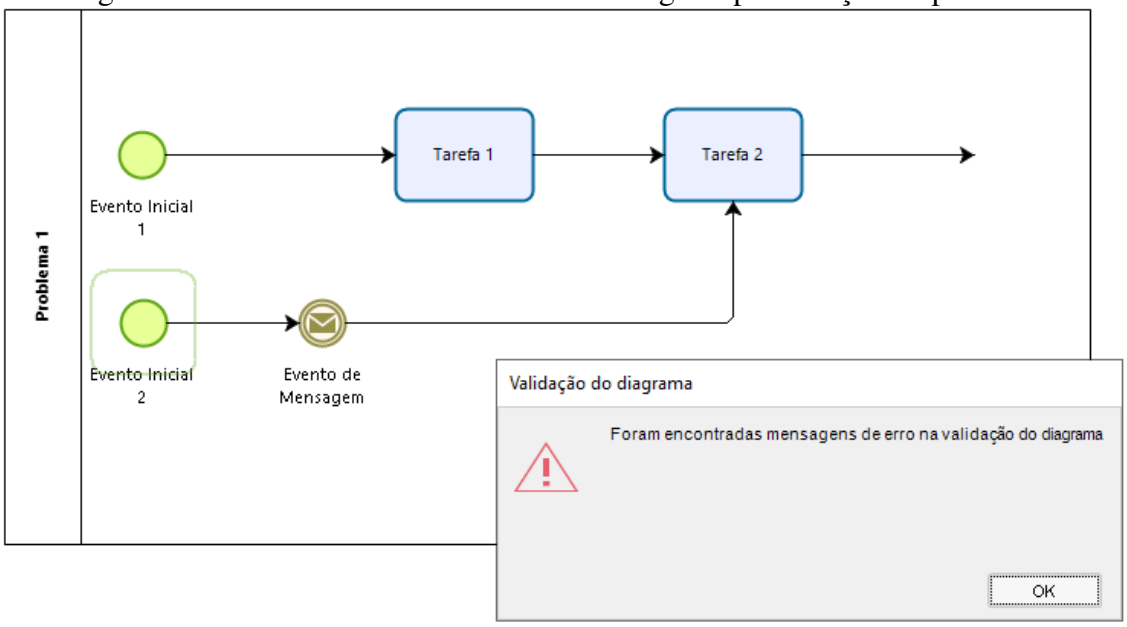
Fonte: O Autor

de inicio.

A falta de aviso da não existência de evento inicial se torna ainda mais emblemática quando apresentado um outro problema não abordado nos 6 problemas desse trabalho porém também corrigido pelo analisador que será abordado no capítulo 5, que é a existência de 2 (ou mais) eventos iniciais. Quando apresentamos esse caso para o Bizagi ele informa corretamente os problemas, conforme figura 4.3.

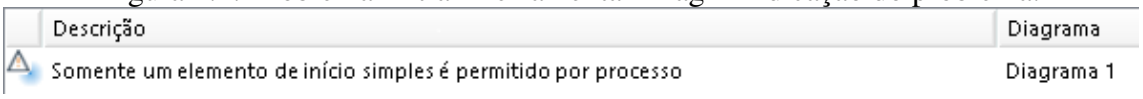
Na figura 4.3 existem 2 eventos iniciais, o que gera mensagens de problema na verificação do processo. Além disso, quando analisamos o motivo do problema na figura 4.4 nota-se observar uma avaliação correta por parte da ferramenta, o que nos mostra que a primeira avaliação também poderia ser implementada pelo mesmo. Ainda assim, como está sendo avaliado um aviso na falta de um evento de início, o Bizagi acaba estando in-

Figura 4.3: Problema Extra - Ferramenta Bizagi - Apresentação do problema.



Fonte: O Autor

Figura 4.4: Problema Extra - Ferramenta Bizagi - Indicação do problema.



Fonte: O Autor

correto.

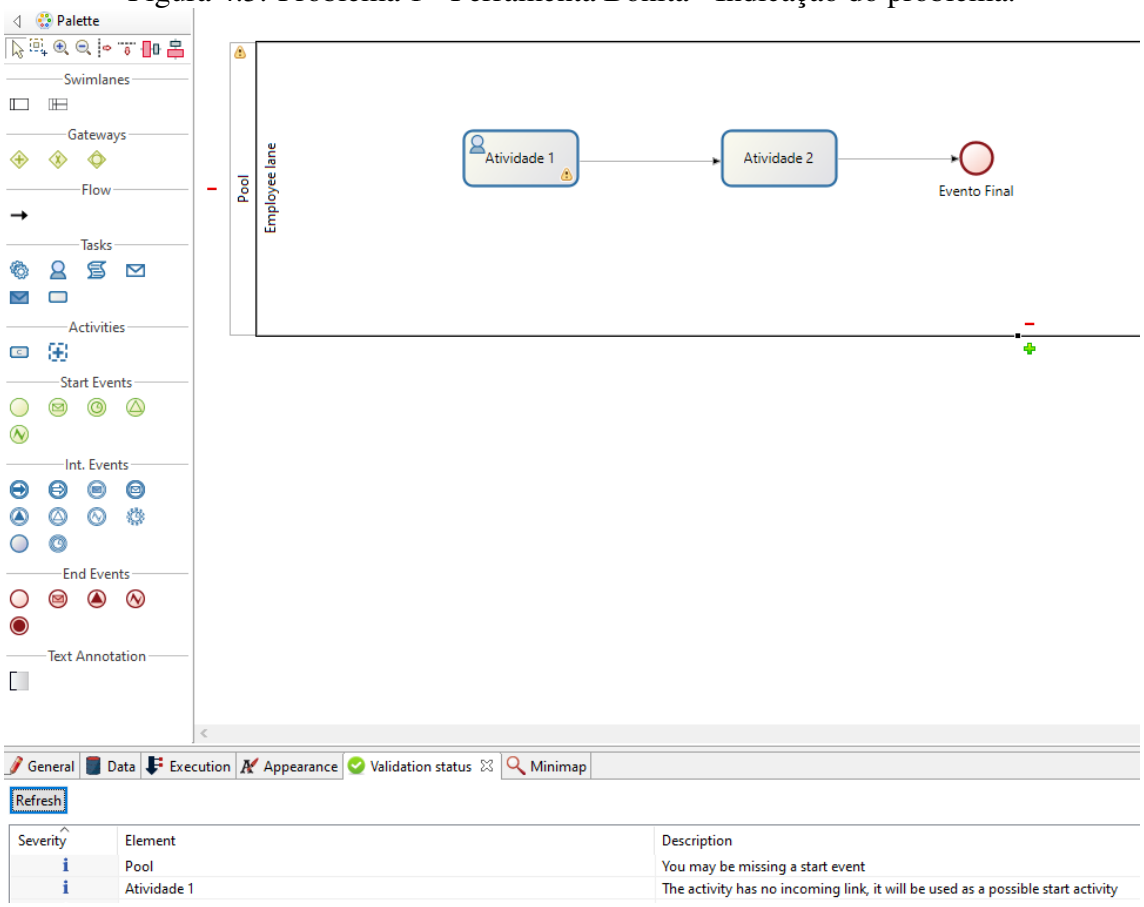
A segunda ferramenta para ser avaliada no primeiro problema é a Bonita, a qual apresentou uma avaliação correta desse problema. A avaliação pode ser vista na figura 4.5.

Na figura 4.5 podemos notar na parte inferior da figura a avaliação da ferramenta sobre o problema do processo, que pode ser traduzida (em tradução livre) como "Você pode estar esquecendo um evento de início". Essa avaliação é correta e muito útil para um usuário que realmente esqueceu um evento de início. Além disso, ela deixa claro que isso não é necessariamente um problema, mas é recomendado.

A segunda mensagem apresentada pelo software é muito interessante. A mensagem em tradução livre significa "A atividade não tem fluxo de entrada, ela será utilizada como possível atividade de início". Essa mensagem é relacionada a Atividade 1 e está correta quanto a falta de fluxo de entrada.

A ferramenta Bonita além de apresentar a indicação de falta de evento inicial também apresenta uma opção de atividade inicial baseada na falta de fluxo de entrada, o que é muito interessante para um usuário que estaria procurando o início do processo.

Figura 4.5: Problema 1 - Ferramenta Bonita - Indicação do problema.



Fonte: O Autor

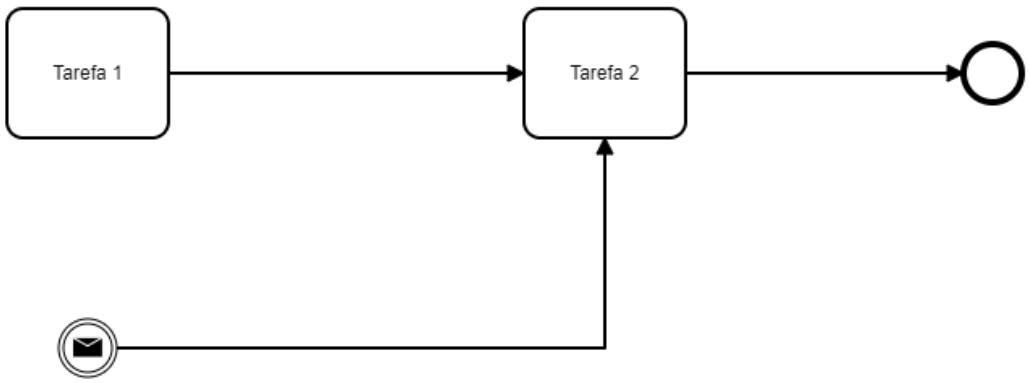
A terceira ferramenta (Camunda) não apresenta um analisador de modelos de processos e dessa forma será avaliada como incorreta. Apesar disso a ferramenta apresentou a impossibilidade de criação de alguns dos problemas, e sendo assim ainda pode ser considerada correta em um dos quesitos. No problema 1 a ferramenta possibilitou a criação do processo conforme figura 4.6.

Pela falta de indicação de problema no processo da figura 4.6 a ferramenta Camunda foi definida como incorreta nesse problema.

A última ferramenta a ser avaliada para esse problema foi a ferramenta Signavio. Essa ferramenta apresentou uma maneira diferente de avaliar o início dos processos, conforme figura 4.7.

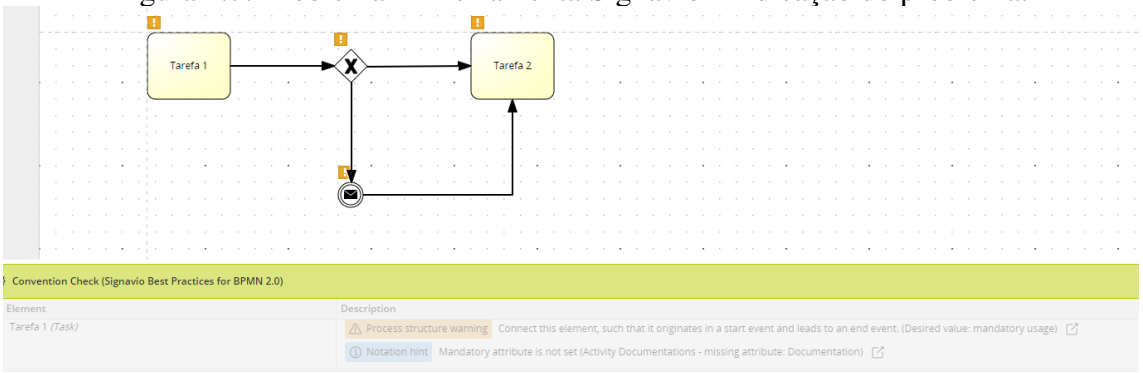
O software apresentou corretamente um aviso (*warning*) que indica que a atividade não faz parte de um caminho começando em um evento inicial e terminando em um evento final. A mensagem pode ser traduzida livremente como "Conecte esse elemento de forma que ele se origine em um evento inicial e acabe em um evento final". É uma maneira alternativa e correta de verificar as atividades do processo.

Figura 4.6: Problema 1 - Ferramenta Camunda - Sem indicação do problema.



Fonte: O Autor

Figura 4.7: Problema 1 - Ferramenta Signavio - Indicação do problema.



Fonte: O Autor

Apesar disso quando a ferramenta apresenta o aviso ela não indica o que está faltando para o usuário, tornando assim mais complexo para o entendimento do aviso. Nessa situação é mais transparente informar que está faltando um evento inicial, do que informar para cada atividade e evento que não está conectado em um caminho de um evento inicial para um final.

Considerando isso a ferramenta foi considerada parcialmente correta, já que ela indica corretamente a falta de um evento inicial, porém sem especificar diretamente ao usuário esse problema, indicando ao invés disso com uma opção alternativa.

Completando assim a avaliação do primeiro problema, podemos analisar o conjunto das ferramentas e suas respectivas avaliações para esse caso. Para isso foi montado uma comparação com três cores diferentes indicando os diferentes níveis de avaliação.

Uma ferramenta correta conforme a descrição indicada no início do capítulo 4 é indicada por um quadrado verde, enquanto uma ferramenta incorreta é indicada por um

quadrado vermelho. Por fim uma ferramenta parcialmente correta é indicada por um quadrado amarelo. Todos os níveis são apresentados na legenda para fim de uma análise puramente a partir da figura.

A partir da comparação entre ferramentas podemos atribuir um peso a cada um

Tabela 4.1: Problema 1 - Comparação Entre Ferramentas.

Problema/Ferramenta	Bizagi	Bonita	Camunda	Signavio
1				
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

dos padrões de cores e capacidade de verificação que havíamos definido no início do capítulo 4. Podemos atribuir o peso 1 para uma avaliação correta, e peso 0 para uma avaliação incorreta. Também podemos atribuir um peso 0,5 para a avaliação parcialmente correta.

Os valores atribuídos a essas avaliações são aproximações de quão correta a solução se apresenta e para padronizar esses valores não são utilizados valores específicos para cada solução, e sim um valor aproximado para uma avaliação específica. Isso ocorre pois esse trabalho não tem o intuito de comparar qual dos softwares se apresenta melhor nessas situações, e sim avaliar o estado da arte e entender se em sua maioria os softwares que temos no mercado hoje em dia solucionam corretamente cada um desses problemas.

Dessa forma a avaliação considerou que os softwares em conjunto apresentaram 1,5 pontos, considerando também que a melhor ferramenta avaliada conseguiu apresentar uma solução bastante adequada ao problema, enquanto as ferramentas menos adequadas não conseguiram detectar e apresentar o problema ao usuário.

Dessa forma, considerando que foram somados 1,5 pontos pelas ferramentas de 4 pontos possíveis, podemos avaliar que a média das ferramentas avaliadas como um todo para esse problema foi de 0,375 ou 37,5%.

Essa média indica que a realidade das ferramentas de modelagem atuais têm uma dificuldades para apresentar corretamente o problema de número 1. Mesmo uma das ferramentas apresentando resultado correto, a maioria apresentou um resultado insuficiente ou completamente incorreto. Dessa forma podemos avaliar que o problema 1 é relevante para o analisador sintático desenvolvido no capítulo 5 desse trabalho e deve ser imple-

Tabela 4.2: Problema 1 - Avaliação de Resultados.

Problema/Resultados	Maior	Menor	Total	Media
1			1,5	0,375
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

mentado.

4.2 Avaliação do Problema 2

Concluindo assim o problema 1, passamos a avaliar o segundo problema, que se apresenta quando uma *Atividade* ou *evento* não possui um fluxo de entrada e é categorizado como problema semântico. Quando presente esse problema possibilita que uma atividade ou evento não seja executada, mesmo quando o processo é corretamente concluído.

Como um problema semântico, esse problema pode alterar o sentido e a funcionalidade de processos e por isso a correção é considerada obrigatória. Conforme o capítulo 7 da descrição oficial do BPMN um fluxo de entrada é obrigatório para atividades e eventos(GROUP, 2013). Assim sendo iniciamos avaliando o problema 2 para a ferramenta Bizagi, que foi executado conforme imagens 4.10.

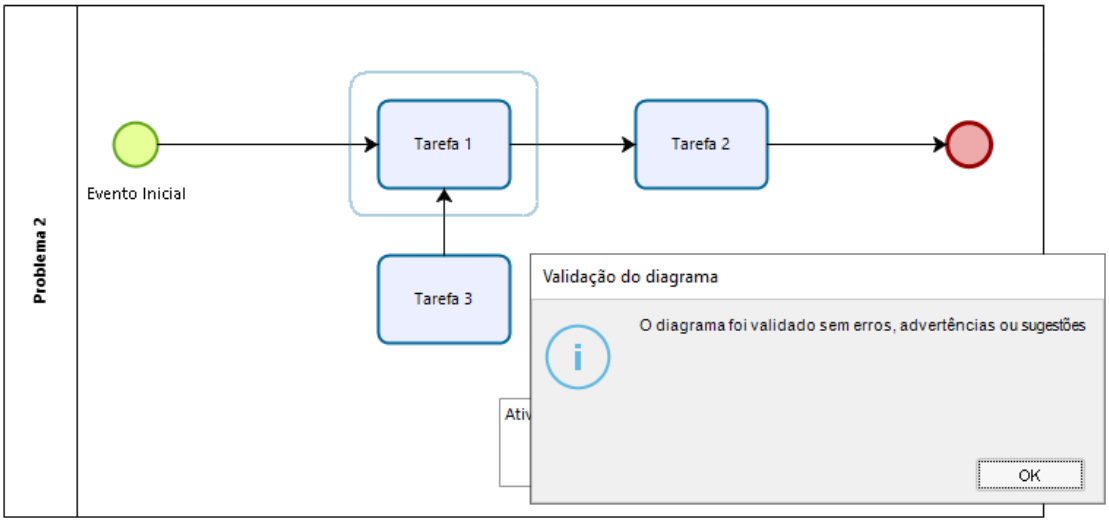
A partir da figura 4.8 podemos notar que a ferramenta indica que a avaliação foi concluída sem problemas, advertências ou sugestões. Apesar disso, ao avaliar a figura 4.8 notamos que existe uma atividade sem fluxo de entrada, em conformidade com o problema 2.

Foram realizados testes com eventos e os mesmos apresentaram a correta indicação de problema, conforme figura 4.1 e 4.2. Dessa forma a ferramenta tem uma abordagem diferente para eventos e atividades, porém considerando que nossa abordagem considera os dois casos, podemos considerar as duas situações.

Dessa forma a ferramenta foi considerada parcialmente correta, já que apresenta resultados corretos para eventos e resultados incorretos para atividades. Assim é notado que as ferramentas devem ser completas para serem avaliadas como corretas.

A próxima ferramenta a ser avaliada é a ferramenta Bonita, que foi testada con-

Figura 4.8: Problema 2 - Ferramenta Bizagi - Apresentação de problema



Fonte: O Autor

forme figura 4.9.

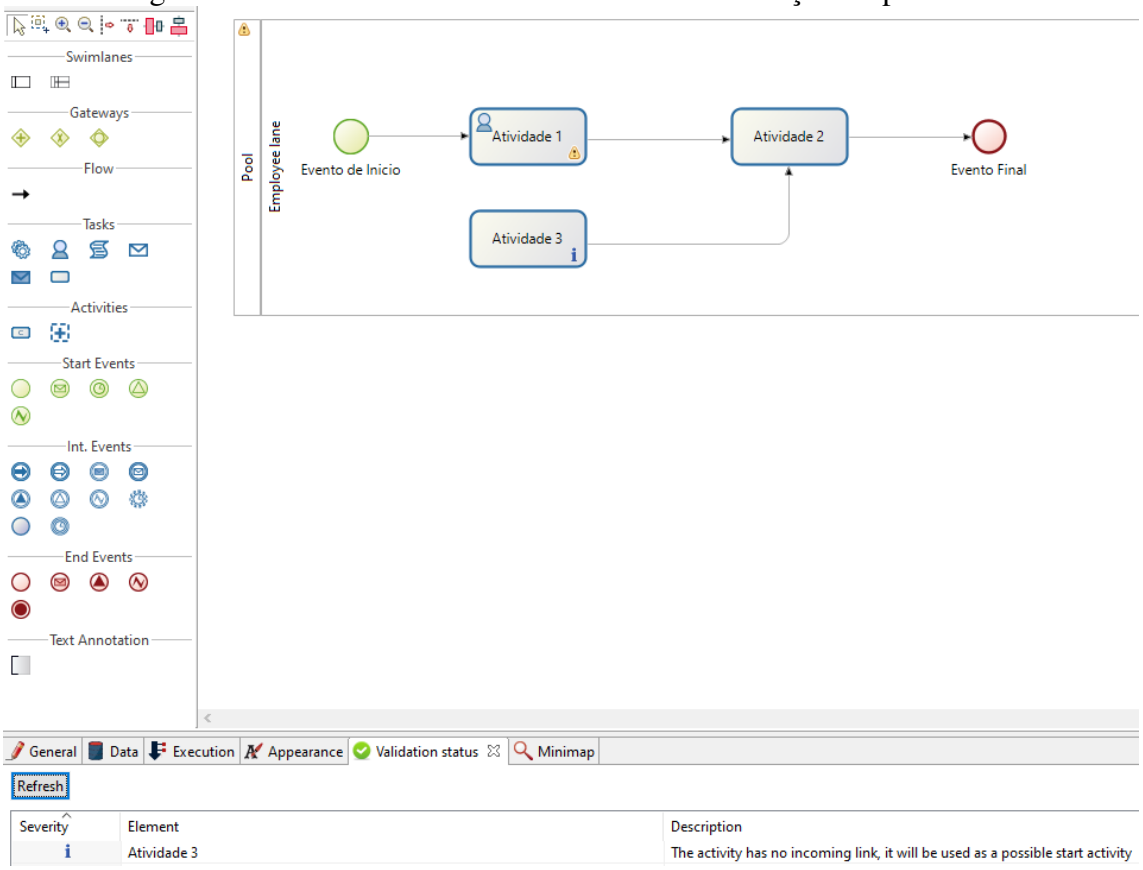
Conforme podemos observar na figura 4.9, a ferramenta apresenta uma mensagem de aviso que pode ser traduzida livremente para "A atividade não tem fluxo de entrada, ela será utilizada como possível atividade de início". Da mesma forma que no problema 1, essa mensagem está correta. Porém a mesma indica que a atividade será utilizada como possível evento de início mesmo já existindo um evento inicial.

Ainda assim a mensagem informa corretamente a falta de fluxo de entrada de uma atividade e mostra para o usuário exatamente qual a atividade a qual o problema é apresentado. Dessa forma, a indicação incorreta de que a atividade poderia ser considerada inicial pode ser descartada com a evidente correção que deverá ser feita pelo usuário.

Considerando a avaliação e o aviso correto sobre o problema indicado, a ferramenta pode ser considerada correta na resolução desse problema. Isso se deve principalmente a indicação exata da mensagem sobre a falta de fluxo de entrada, deixando assim evidente o problema para qualquer usuário.

A terceira ferramenta a ser avaliada é o software Camunda. Esse software conforme citado acima não apresenta uma ferramenta de avaliação de problemas e por isso só pode ser avaliado se a ferramenta permite ou não a criação dos modelos. Podemos ver que a ferramenta permite a criação do modelo do problema 2 na imagem 4.10. É possível avaliarmos a partir da imagem 4.10 que novamente a ferramenta permite que seja realizado a criação do problema sem nenhum aviso e nenhuma mensagem de problema. Além disso todas as conexões puderam ser realizadas o que indica que a ferramenta não está apta a avaliar esse problema.

Figura 4.9: Problema 2 - Ferramenta Bonita - Indicação do problema..



Fonte: O Autor

Dessa forma a ferramenta Camunda pode ser considerada incorreta para o problema 2, já que a mesma não tem a capacidade de enviar avisos e problemas para os usuários e permite a criação do problema 2.

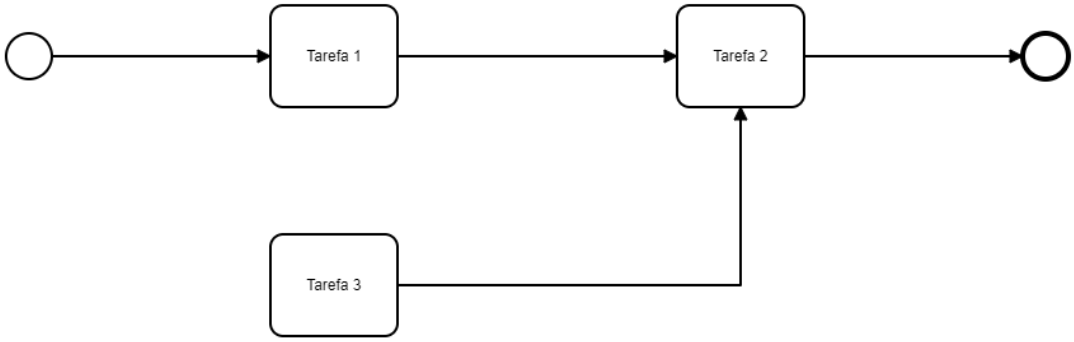
A última ferramenta a ser avaliada é a ferramenta Signavio, que foi testada conforme a figura 4.11.

A ferramenta testada apresenta a mesma mensagem de aviso do problema 1, porém dessa vez a mensagem só é apresentada para a atividade que não possui fluxo de entrada. A avaliação da mensagem pode ser traduzida livremente como "Conecte esse elemento de forma que ele se origine em um evento inicial e acabe em um evento final".

A mensagem apresentada, apesar de correta, não esclarece diretamente o problema do processo. Ela indica que existe algum motivo para a atividade não estar dentro do fluxo padrão, que começa em um evento inicial e termina em um evento final, porém não consegue definir qual é o motivo. Isso pode ser comprovado pois a mesma mensagem foi indicada no último problema por um motivo diferente, e em nenhuma das duas oportunidades foi especificado o motivo exato.

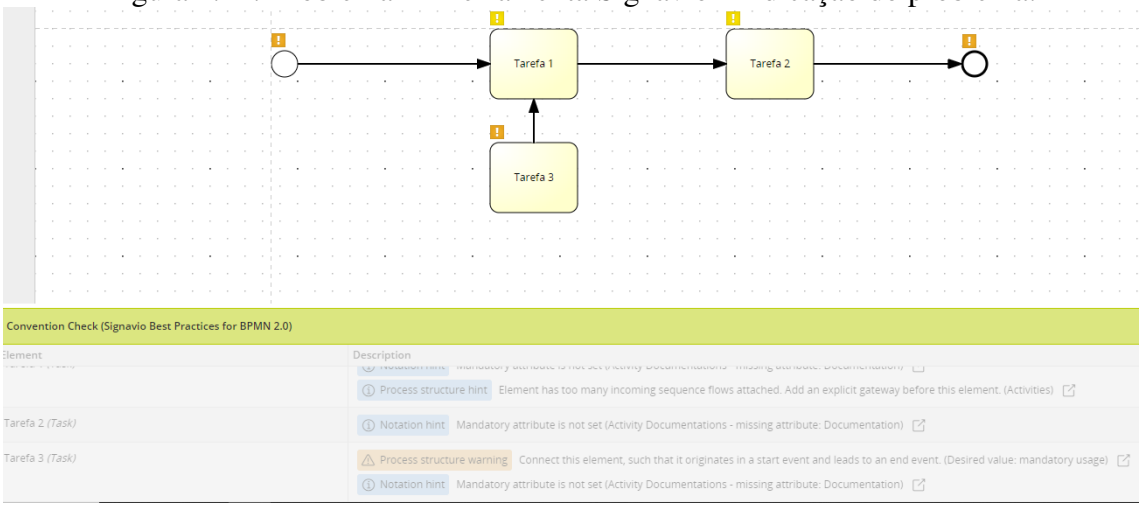
Dessa forma podemos considerar a ferramenta Signavio parcialmente correta, já

Figura 4.10: Problema 2 - Ferramenta Camunda - Sem indicação do problema.



Fonte: O Autor

Figura 4.11: Problema 2 - Ferramenta Signavio - Indicação do problema.



Fonte: O Autor

que a mesma apresenta uma mensagem correta que indica um problema na atividade certa, porém não consegue detectar exatamente qual o motivo desse problema. É necessário que o próprio usuário avalie e descubra qual o problema em questão.

Por fim podemos condensar as avaliações das ferramentas para o problema 2 na tabela 4.3. É necessário considerar a legenda conforme anteriormente estipulado. Uma avaliação completamente correta recebe cor verde, enquanto uma parcialmente correta recebe cor amarela. Quando uma avaliação é considerada incorreta ela recebe cor vermelha.

Observando a tabela 4.3 podemos reparar que todas as ferramentas que possuem avaliação de problemas tiveram um resultado ao menos parcialmente correto para esse problema. Apesar disso somente um resultado obtido foi considerado completamente correto, por ter detectado corretamente o problema e apresentado o mesmo ao usuário de

Tabela 4.3: Problema 2 - Comparação Entre Ferramentas.

Problema/Ferramenta	Bizagi	Bonita	Camunda	Signavio
2				
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

forma correta.

Podemos avaliar também que as duas ferramentas parcialmente corretas tiveram dificuldades para detectar os problemas, tendo uma detectado o problema mas não o motivo exato (Signavio) e uma detectado o problema somente em um dos casos (Bizagi). A única ferramenta que detectou corretamente foi a ferramenta Bonita.

Para melhor avaliar esses resultados utilizamos novamente uma pontuação onde um resultado correto tem um valor 1, enquanto um resultado parcialmente correto tem um valor 0,5. Um resultado incorreto tem um valor 0 já que não apresentou nem parcialmente o problema para o usuário. Assim podemos montar as estatísticas na tabela 4.4.

Conforme podemos observar na tabela 4.4, a média dos resultados apresentados

Tabela 4.4: Problema 2 - Avaliação de Resultados.

Problema/Resultados	Maior	Menor	Total	Media
2			2	0,5
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

foi de 0,5 ou 50%. Essa média indica que uma parte das ferramentas disponíveis no mercado podem corretamente indicar o problema 2. Apesar disso uma parte das ferramentas disponíveis no mercado não consegue indicar o problema 2.

Por esse motivo o problema 2 parece ser adequado para o trabalho do próximo capítulo, já que o mesmo não é corretamente avaliado por muitas ferramentas do mercado. Dessa forma um analisador sintático que tenha capacidade de detectar e indicar esse problema é relevante para o estado da arte.

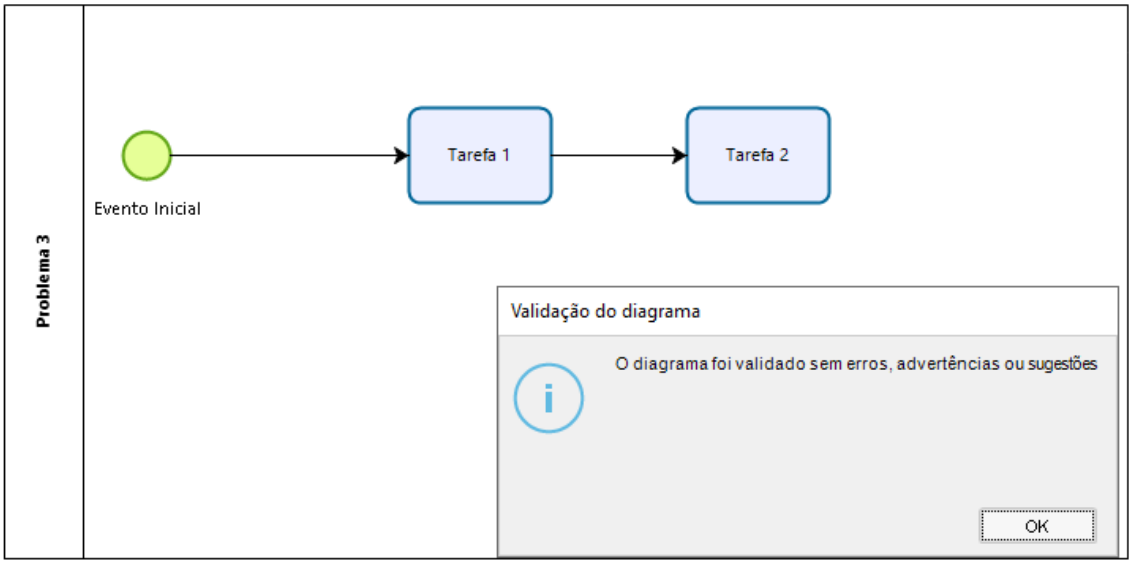
4.3 Avaliação do Problema 3

Com essa conclusão podemos passar ao problema 3, que é definido por um *processo* não conter um *evento* final. Esse problema é importante pois sem um evento final nenhuma atividade pode ser considerada concluída pela empresa. Além disso alguns eventos finais podem ser gatilhos para outros eventos e processos e sem eles os gatilhos nunca serão ativados. A definição do problema 3 é semelhante ao do problema 1 e por esse motivo iremos comparar os resultados obtidos aqui com os resultados do problema 1.

Inicialmente testamos a ferramenta Bizagi, conforme a figura 4.12.

A ferramenta avaliada apresenta a mensagem que podemos ver na figura 4.12,

Figura 4.12: Problema 3 - Ferramenta Bizagi - Sem apresentação de problema.



Fonte: O Autor

sendo ela "O diagrama foi validado sem problemas, advertências ou sugestões". Essa mensagem indicada pela ferramenta de modelagem é diferente da indicada no problema 1 pela mesma ferramenta, onde foram avaliados problemas.

Além disso, não foi informado nenhum aviso de problema, o que também contraria o problema 1, onde foi avisado que uma atividade não possuía fluxo de entrada. Nesse caso, como todas as atividades possuem fluxo de entrada nenhum aviso foi enviado ao usuário, mesmo com a falta de um evento final.

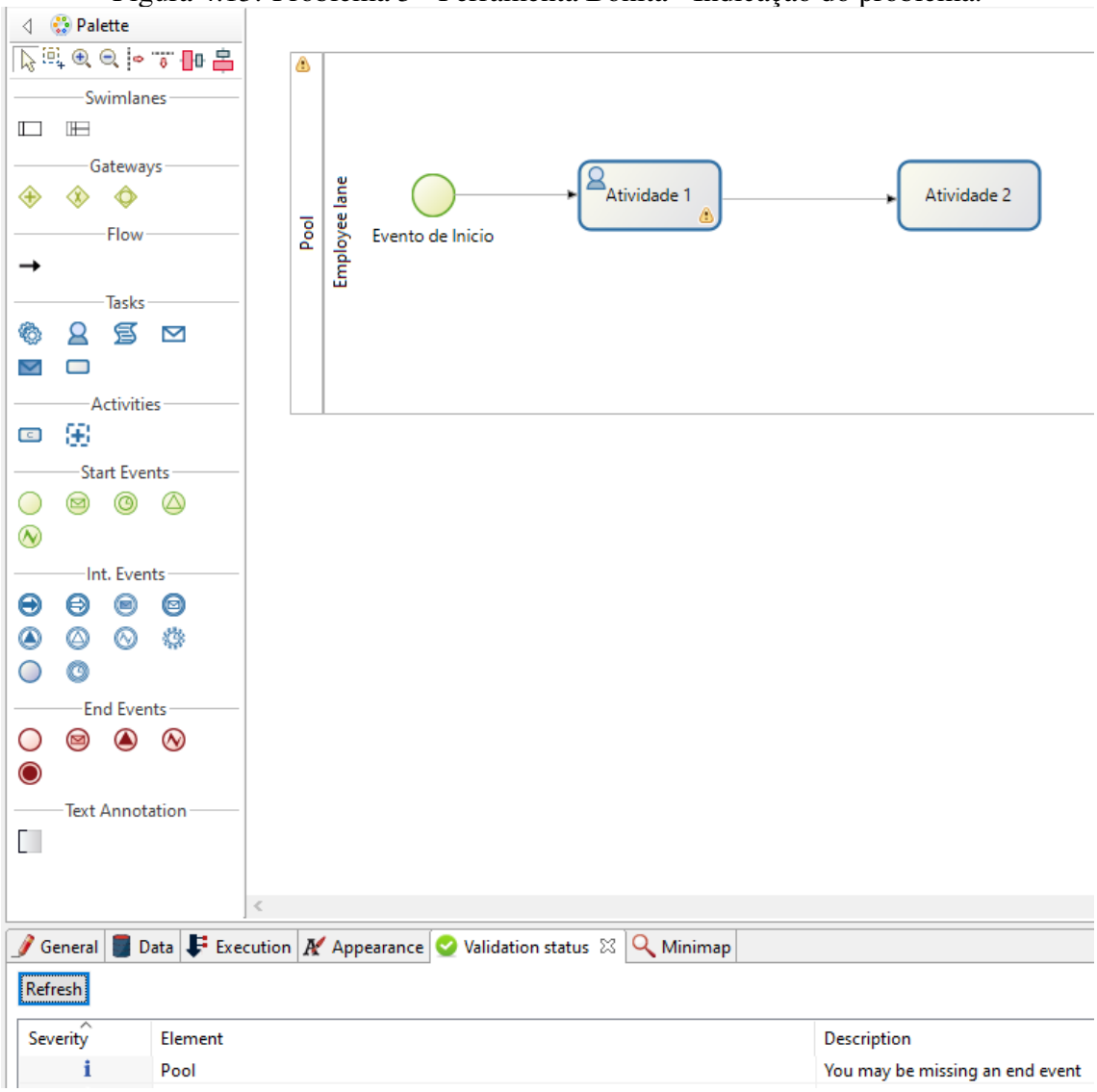
A avaliação Bizagi também mostra por que não podemos considerar a avaliação do problema 1 parcialmente correta. A ferramenta não detectou a falta de evento inicial e sim a falta de entrada em um evento.

Considerando que não foi detectado nenhum problema a ferramenta deve ser considerada incorreta para o problema 3. Além disso absolutamente nada foi informado ao usuário, o que piora a situação já que o usuário pode não perceber o seu problema.

A ferramenta Bonita foi testada para a ferramenta 2 de forma a gerar a figura 4.13.

Na figura 4.13 a ferramenta Bonita indica uma mensagem de aviso que pode ser

Figura 4.13: Problema 3 - Ferramenta Bonita - Indicação do problema.



Fonte: O Autor

traduzida livremente como "Você pode estar esquecendo um evento final". Novamente a mensagem da ferramenta é exatamente o que era esperado como avaliação, já que a mesma apresenta não só que existe um problema, mas também o problema exato.

Quando comparamos essa mensagem com a mensagem do problema 1 notamos que a ferramenta consegue diferenciar eventos iniciais de eventos finais e indicar qual é o evento que está faltando, o que cria uma advertência exata para o usuário.

Assim considerando que a ferramenta indica o problema 3 corretamente para o usuário a ferramenta é considerada Correta para o problema 3. A ferramenta também foi considerada correta no problema 1.

A próxima ferramenta a ser testada foi a ferramenta Camunda, teste esse que pode ser observado na figura 4.14.

Novamente a ferramenta Camunda não indica nenhum problema e permite que

Figura 4.14: Problema 3 - Ferramenta Camunda - Sem indicação do problema.

Fonte: O Autor

o usuário crie um processo em um formato inadequado, como já observado por outros trabalhos (DANI, 2019). É importante observar que a ferramenta não possui suporte para avaliar problemas e por isso não indica problemas ao usuário.

Ainda assim, a ferramenta não permite que algumas ações possam ser efetuadas e por isso ainda pode ser avaliada, principalmente considerando que é uma ferramenta utilizada no mercado. A ferramenta Camunda foi avaliada como incorreta para o problema 3.

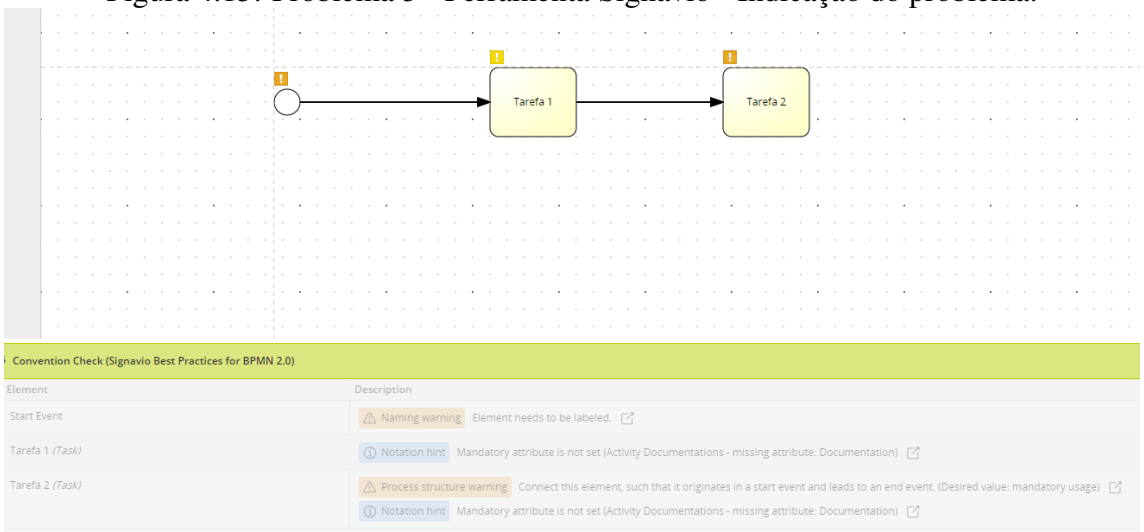
Por fim, testamos a ferramenta Signavio para o problema 3 conforme figura 4.15.

A figura 4.15 mostra que a ferramenta Signavio apresentou uma mensagem de aviso que pode ser traduzida como "Conecte esse elemento de forma que ele se origine em um evento inicial e acabe em um evento final". Essa mensagem se apresenta correta já que as atividades não participam de um fluxo que vai até o evento final.

Porém a mensagem não informa que na verdade não existe um evento final, e por essa razão não é possível que exista um fluxo correto, conforme solicitado pela ferramenta. A mensagem apresentada é exatamente igual a mensagem do problema 1, e tem a mesma dificuldade de detectar a origem do problema.

Mesmo considerando o aviso gerado pela ferramenta correto é importante destacar que ele não indica o exato motivo do problema e por isso não pode ser considerado completamente correto. A mensagem é insuficiente para que o usuário possa entender seu problema sem que tenha que reavaliar todo o processo.

Figura 4.15: Problema 3 - Ferramenta Signavio - Indicação do problema.



Fonte: O Autor

Dessa forma a ferramenta Signavio recebeu uma avaliação parcialmente correta no problema 3. A ferramenta recebeu exatamente a mesma avaliação no problema 1, sem nenhuma modificação nem mesmo na mensagem de aviso.

Assim concluímos a avaliação do problema 3 que pode ser resumida na tabela 4.5.

Na tabela 4.5 podemos observar que das quatro avaliações somente uma foi consi-

Tabela 4.5: Problema 3 - Comparação Entre Ferramentas.

Problema/Ferramenta	Bizagi	Bonita	Camunda	Signavio
3				
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

derada completamente correta, enquanto duas foram consideradas completamente incorretas. Entre elas existiu ainda uma avaliação parcialmente correta.

As ferramentas tiveram bastante dificuldade para perceber a falta de um evento final, com duas delas (Bizagi e Camunda) nem mesmo citando um problema, enquanto uma conseguiu detectar um problema mas não conseguiu especificar qual.

Novamente o destaque do problema 3 foi a ferramenta Bonita, que especificou exatamente o problema e alertou o usuário de seu problema. A avaliação do problema 3 foi no final das contas a mesma do problema 1, mostrando que as ferramentas que possuiam dificuldades para detectar a falta de eventos iniciais também tem dificuldades para detectar a falta de eventos finais.

São apresentadas na tabela 4.6 as estatísticas do problema 3.

Como podemos observar na tabela 4.6, mesmo tendo um resultado máximo po-

Tabela 4.6: Problema 3 - Avaliação de Resultados.

Problema/Resultados	Maior	Menor	Total	Media
3			1,5	0,375
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

sitivo, a média de acerto das ferramentas ainda foi abaixo do esperado. É importante ressaltar que uma avaliação correta implica em 1 ponto na avaliação acima, enquanto uma avaliação parcialmente correta implica em 0.5 pontos. Uma avaliação incorreta não atribui nenhum ponto para a ferramenta nesse problema.

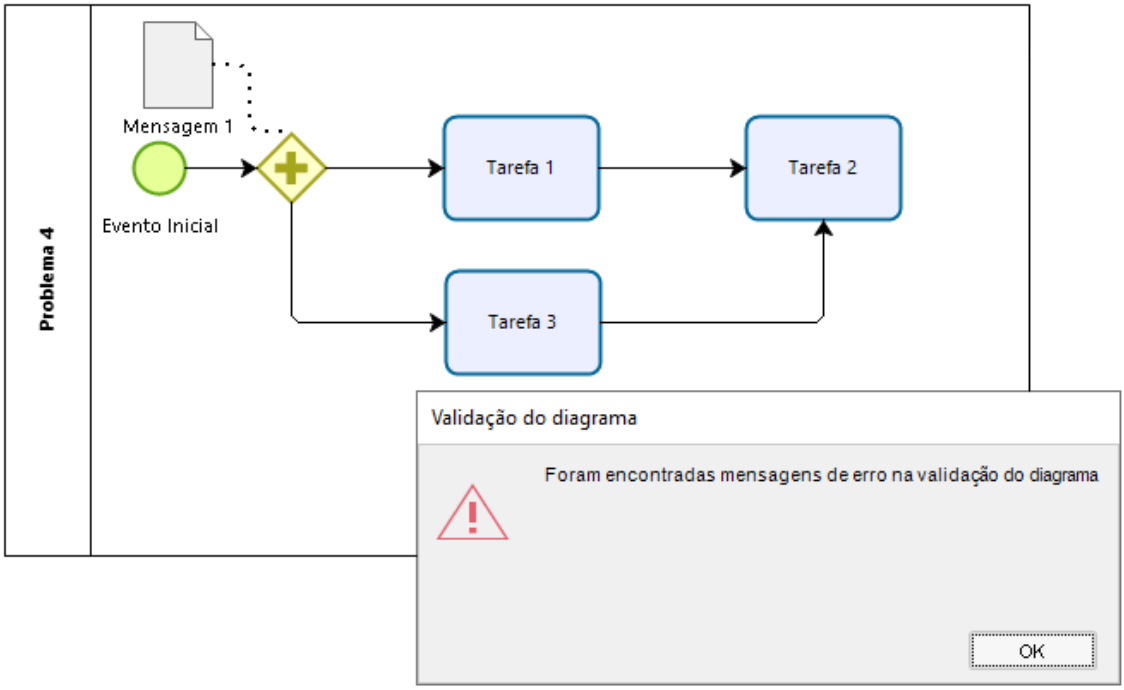
Totalizando 1,5 pontos de 4 possíveis, as ferramentas tem uma média de apenas 0,375 ou 37,5%. A média abaixo dos 50% indica que muitas ferramentas ainda tem dificuldades em detectar esse problema e por isso o problema é relevante para o estado da arte. Dessa forma podemos avaliar que é importante para o trabalho a correção desse problema de maneira correta e que possa ser utilizada por todos.

4.4 Avaliação do Problema 4

Podemos passar agora para o problema 4, que foi descrito como um *gateway* receber ou gerar mensagens e pode ser considerado um problema sintático e semântico. Por ser um problema semântico o problema 4 dificulta o entendimento do processo, tirando parte do propósito do mesmo. Porém além disso como um problema sintático, o problema 4 é considerado completamente errado para o BPMN e impossível de ser executado.

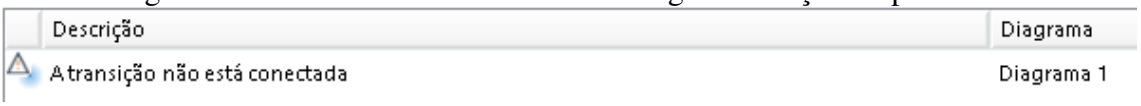
Por esse motivo o problema 4 é o problema mais fácil de ser detectado da lista, já que qualquer ferramenta que aceite a criação e conexão desse problema está diretamente fora das normas da BPMN conforme o Object Management Group(GROUP, 2013). Assim sendo, uma vez conectada na ferramenta uma mensagem em um gateway, o problema pode ser considerado e nenhum arquivo BPMN poderia ser gerado com essa conexão. A partir disso passamos a avaliar o problema 4 para o software Bizagi. Conforme podemos ver na figura 4.16, quando tentamos conectar um gateway a uma mensagem temos um

Figura 4.16: Problema 4 - Ferramenta Bizagi - Apresentação do problema.



Fonte: O Autor

Figura 4.17: Problema 4 - Ferramenta Bizagi - Indicação do problema.



Fonte: O Autor

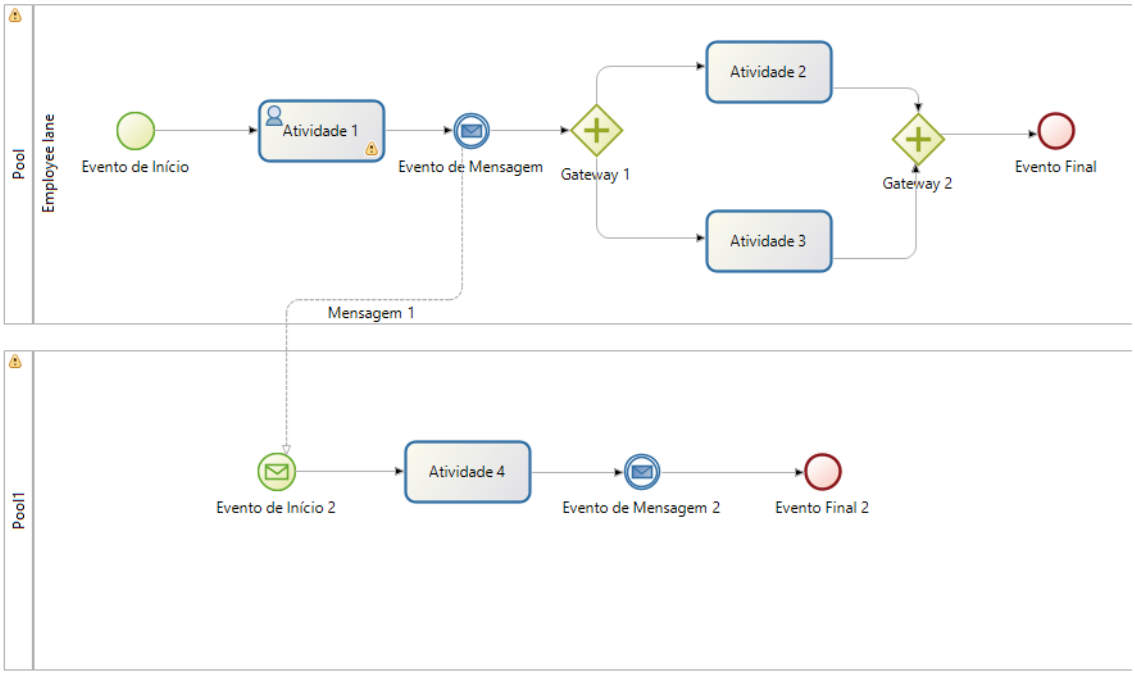
problema apontado pelo software. A ferramenta Bizagi aponta na figura 4.17 que a transição não está conectada, e quando avaliamos a transição é possível notar que a mesma está solta na ponta do gateway.

Isso acontece pois a ferramenta Bizagi não permite que a transição esteja conectada nesse ponto, e é até difícil conseguir parar o fluxo no gateway da forma que foi feito na figura 4.16, já que a própria ferramenta parece afastar a transição do gateway. Isso indica uma correta interpretação da ferramenta na qual uma mensagem não pode ser conectada a um gateway.

Dessa forma, a mensagem de saída da ferramenta nunca poderia indicar que não é possível conectar uma mensagem em um gateway já que a própria ferramenta nunca permite que isso aconteça. Tal fato faz com que a solução do software Bizagi para o problema 4 seja considerada correta. A partir daí podemos testar a ferramenta Bonita para o problema 4 conforme a figura 4.18.

Conforme apresentado na figura 4.18 podemos observar que não foi possível co-

Figura 4.18: Problema 4 - Ferramenta Bonita - Impossibilidade de criação do problema.



Fonte: O Autor

nectar o Evento de Mensagem 2 ao Gateway 1 ou 2. Apesar disso foi possível conectar um evento de mensagem em outro evento em outra Pool, garantindo assim que o sistema de mensagens estava funcionando, porém evitando o problema 4.

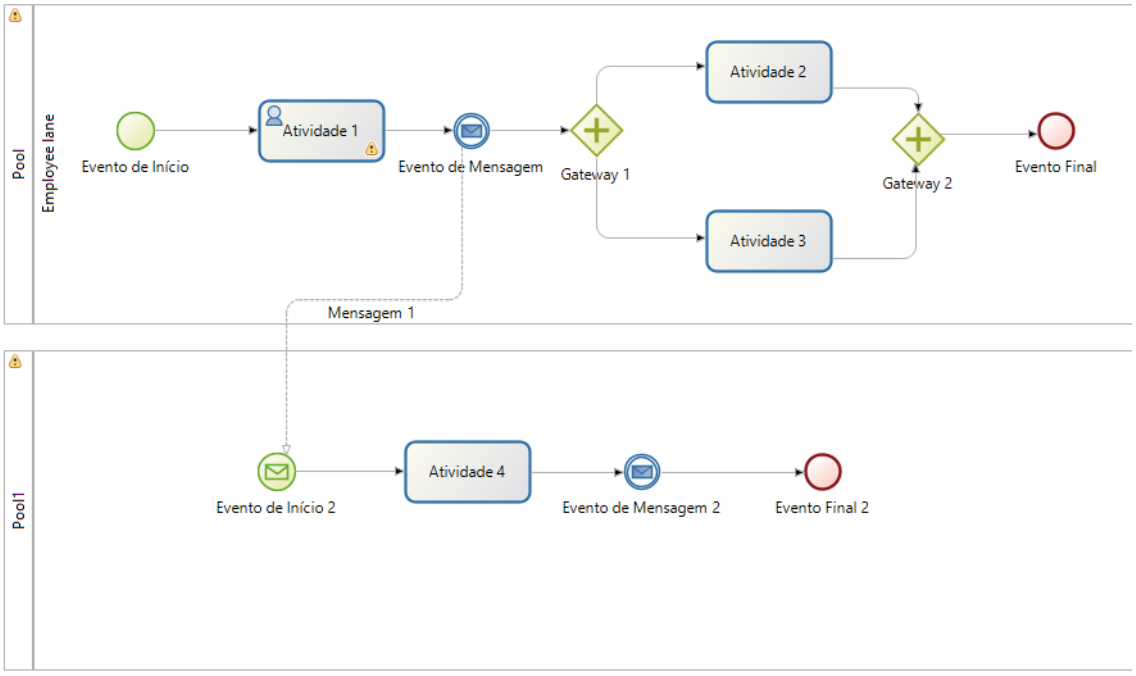
Não foi apresentada nenhuma mensagem ao usuário na tentativa de conexão do Evento de Mensagens 2 a um dos gateways, porém é possível considerar que somente a impossibilidade dessa conexão ocorrer seja mensagem o suficiente de que isso não é permitido.

Dessa forma a ferramenta Bonita teve uma avaliação correta para o problema 4, já que conseguiu evitar que os usuários cometessem esse problema ao mesmo tempo que seu sistema de mensagens é funcional.

A próxima ferramenta a ser testada para o problema 4 é a ferramenta Camunda, conforme figura 4.19. A ferramenta Camunda não possui avaliação de problemas e por isso qualquer atividade permitida em sua ferramenta é aceita como correta para a mesma. Porém no teste do problema 4 a ferramenta não permitiu a conexão entre uma mensagem e um gateway, como apresentado na figura acima.

Dessa forma, mesmo sem apresentar nenhuma avaliação de problema, a ferramenta Camunda ainda assim conseguiu impossibilitar o problema 4 de ocorrer. A ferramenta tem um sistema de mensagens que se conecta corretamente, porém não funciona quando o receptor de uma mensagem é um evento, o que é a solução correta para esse

Figura 4.19: Problema 4 - Ferramenta Camunda - Impossibilidade de criação do problema.



Fonte: O Autor

problema.

Dessa forma a solução do software Camunda foi considerada correta sem o software enviar nenhuma mensagem ao usuário, já que é impossível criar o problema 4 nesse software.

Por fim podemos analisar a ferramenta Signavio para o problema 4, conforme figura 4.20. A figura 4.20 apresenta a impossibilidade do problema ser criado na ferramenta Signavio, onde a ferramenta não possibilita a criação de uma ligação entre uma mensagem e um gateway.

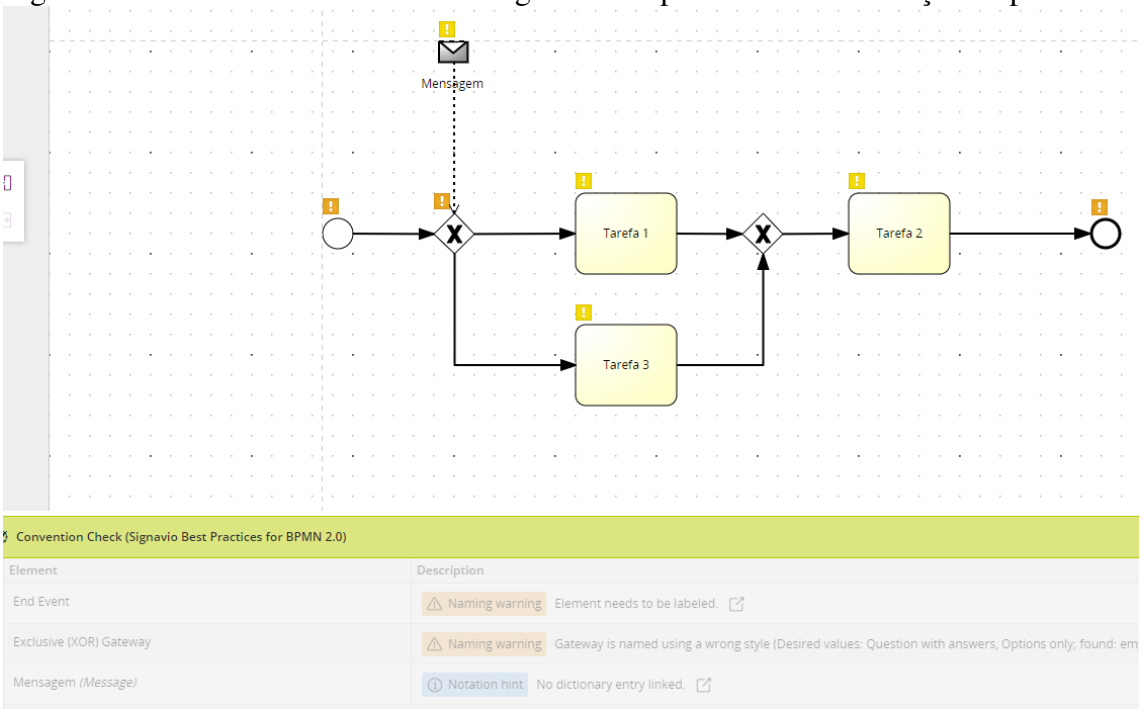
Apesar de possuir um sistema de mensagens funcionando corretamente, o Signavio não possibilita a criação do problema 4, fazendo com que a ferramenta apresente um resultado correto no que abrange a esse problema.

Dessa forma os resultados para o problema 4 podem ser observados na tabela 4.7.

Como podemos observar na tabela 4.7, todas as ferramentas avaliadas impossibilitaram a criação do problema 4, sendo assim consideradas corretas. Isso ocorre pois em nenhum dos softwares foi permitido criar a ligação entre a mensagem e um evento, o que é uma resposta correta para o problema.

Essa avaliação positiva ocorreu pois por ser um problema sintático o problema 4 é uma obrigação de um modelador BPMN e seria impossível gerar um arquivo BPMN

Figura 4.20: Problema 4 - Ferramenta Signavio - Impossibilidade de criação do problema.



Fonte: O Autor

Tabela 4.7: Problema 4 - Comparação Entre Ferramentas.

Problema/Ferramenta	Bizagi	Bonita	Camunda	Signavio
4				
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

correto com esse problema. Ao contrário dos outros problemas que podem ser cometidos por usuários e partem geralmente da falta de entendimento do processo, esse problema poderia ser considerado da ferramenta, conectando partes que não deveriam ser conectadas.

As estatísticas do problema 4 na Tabela 4.8 mostram como as ferramentas obtiveram sucesso em evitar o problema 4.

Conforme a avaliação e considerando uma ferramenta com sucesso com valor 1, as quatro ferramentas obtiveram sucesso e assim a média de sucesso é calculada como 1, ou 100%. Isso indica que todas as ferramentas testadas estão corretas quanto a esse problema, o que pode ser considerado uma resposta extremamente positiva principalmente quando comparado aos outros problemas.

Uma resposta tão positiva somada as conclusões retiradas disso sugere que não é

Tabela 4.8: Problema 4 - Avaliação de Resultados.

Problema/Resultados	Maior	Menor	Total	Media
4			4	1
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

necessário que o analisador de processos BPMN a ser desenvolvido nesse trabalho solucione esse problema, já que ele é evitado pelas ferramentas de criação.

Além disso não é possível gerar arquivos BPMN com esse problema em nenhum dos quatro softwares testados, o que também impossibilitaria os testes do analisador de processos BPMN. Dessa forma, o problema 4 pode ser considerado resolvido pelo estado da arte e não será mais avaliado no analisador de processos do capítulo seguinte.

4.5 Avaliação do Problema 5

Podemos analisar também o problema 5, que é caracterizado por *eventos* de mensagem representando dados do *processo*. Esse problema é considerado um problema semântico pois dificulta a correta interpretação do processo. Um evento de mensagem que representa um dado do processo pode fazer com que o dado nunca seja gerado e dessa forma informação seja perdida pelo processo.

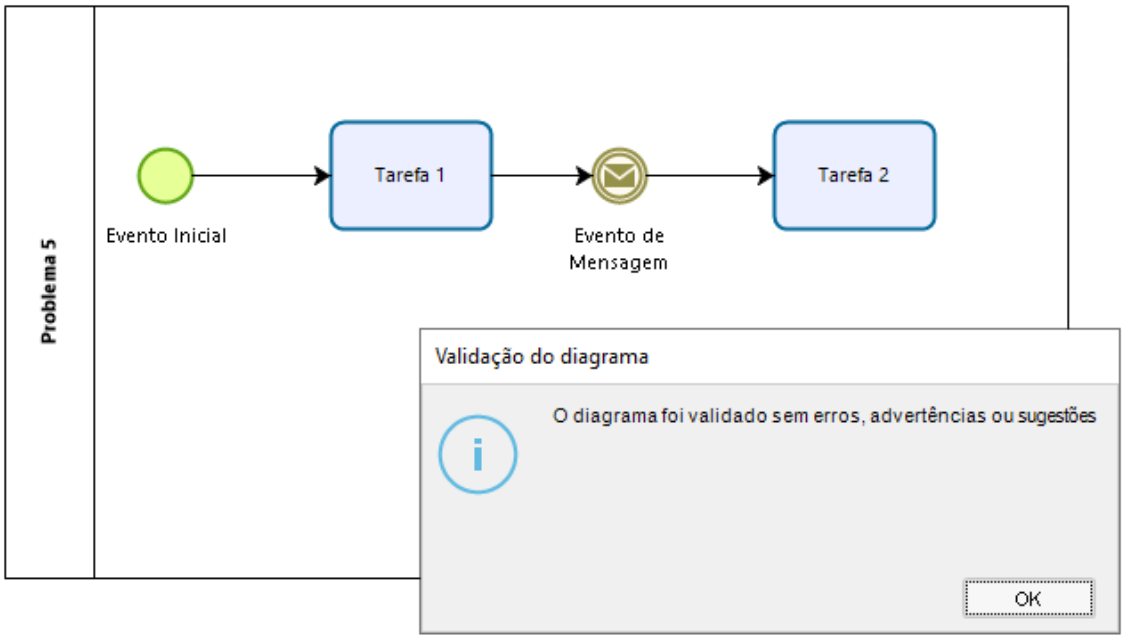
Entre os problemas abordados nesse trabalho esse pode ser o mais complicado de ser detectado, já que o mesmo acontece em situações que não são necessariamente erradas no BPMN. Detectar o problema pode ser considerado um desafio pois o problema acontece em uma situação aceita e correta pelo BPMN.

A melhor maneira de avaliar que a intenção do usuário quando colocado um evento de mensagem poderia ser de guardar dados é avaliando se o evento de mensagem possui um fluxo mensagem a partir do mesmo. Esse fluxo indicaria que uma mensagem realmente está sendo planejada para algum outro processo. Quando o fluxo não existe, podemos entender que esse evento de mensagem pode estar colocado por engano no lugar de dados do processo.

A partir dessas constatações podemos testar a ferramenta Bizagi conforme imagens 4.21.

Como podemos observar na figura 4.21, a ferramenta Bizagi não apresenta ne-

Figura 4.21: Problema 5 - Ferramenta Bizagi - Sem apresentação do problema.



Fonte: O Autor

nhum problema no processo criado. O software não indica que existe um evento de mensagem que não envia nem recebe nenhuma mensagem pois não está conectado em um fluxo de mensagens.

Dessa forma um usuário que posicionou um evento de mensagens por engano e por isso não conectou o mesmo a nenhum outro processo ficaria desavisado de seu engano e poderia não encontrar o seu problema, criando um processo incompreensível para outros já que não existe nenhuma mensagem a ser enviada ou recebida nesse momento.

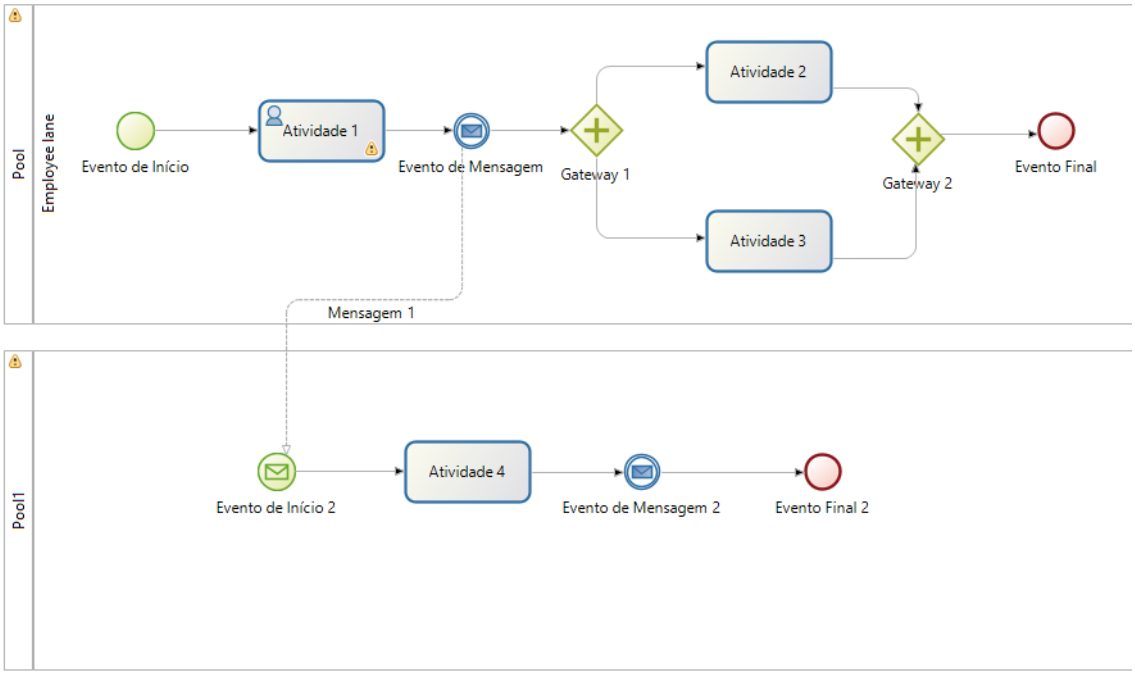
Por esse motivo a ferramenta Bizagi foi considerada incorreta já que permite ao usuário realizar o problema 5 sem nenhum aviso de que ele poderia ter se enganado. Sem esse aviso um dado que pode ser crucial para o processo pode nunca ser criado.

Conforme a figura 4.22 podemos ver o teste do software Bonita.

Pela primeira vez a ferramenta Bonita não apresenta nenhuma mensagem de problema e possibilita a criação de um processo BPMN que pode não ser entendível. A ferramenta aceita a criação do processo e não avisa que existe um evento de mensagem sem nenhuma mensagem, de forma que o usuário pode não notar quando usar um evento de mensagem quando deveria estar usando dados do processo.

Sem nenhuma mensagem a ferramenta não oferece suporte para que o usuário corrija o seu problema e por isso é considerada incorreta. Um evento de mensagem sem nenhum fluxo de mensagem seria um indicativo de que pode ter ocorrido um problema e

Figura 4.22: Problema 5 - Ferramenta Bonita - Sem apresentação do problema.



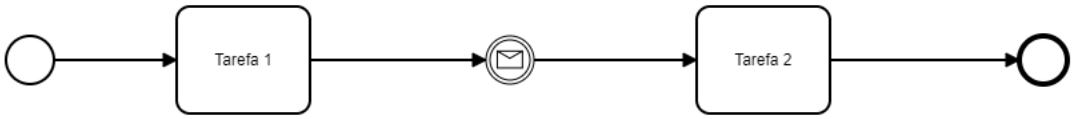
Fonte: O Autor

a ferramenta não oferece nenhuma ajuda ao usuário nesse caso.

Passamos a testar o software Camunda conforme a figura 4.23.

Novamente a ferramenta Camunda não apresenta nenhuma indicação de pro-

Figura 4.23: Problema 5 - Ferramenta Camunda - Sem apresentação do problema.



Fonte: O Autor

blema. O problema 5 segue corretamente as normas do BPMN e por ser um problema semântico, só faz com que o processo gerado esteja incorreto e não seja entendível, dessa forma o único recurso de proteção da ferramenta Camunda, que é a impossibilidade de criar alguns problemas no processo não funcione nesse caso.

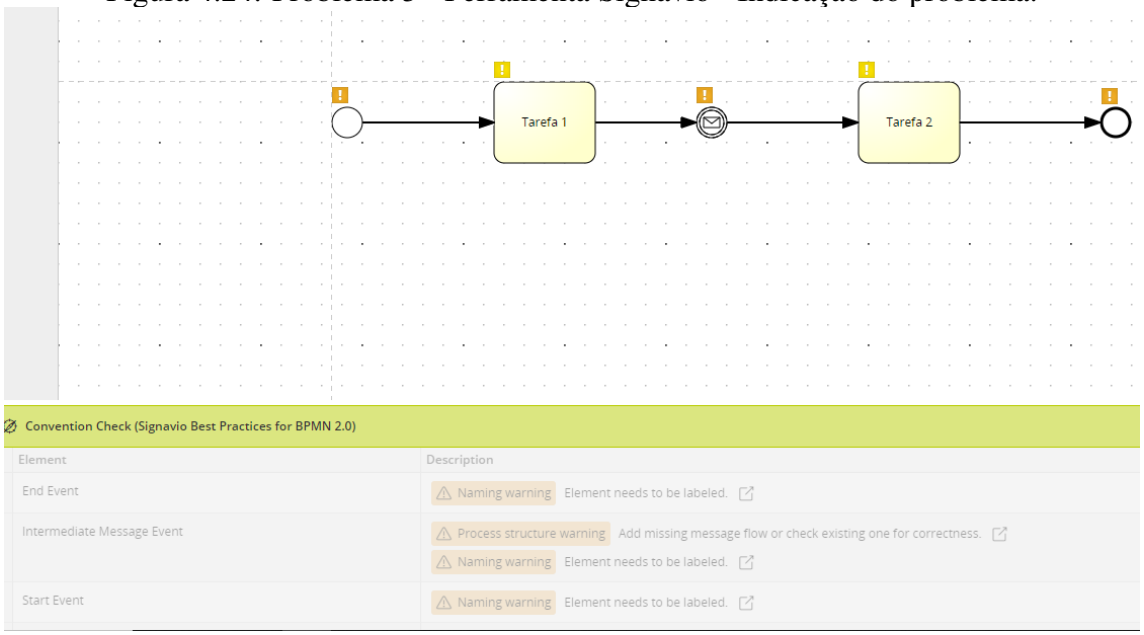
Assim sendo o software Camunda pode ser considerado incorreto para o problema 5 já que não oferece nenhum aviso ao usuário de que ele possa estar cometendo um problema no processo. Os eventos de mensagem sem envio de mensagem são aceitos sem nenhum problema.

Por fim podemos testar a ferramenta Signavio para o problema 5 conforme figura

4.24.

A ferramenta Signavio indica um problema quando apresentada ao problema 5

Figura 4.24: Problema 5 - Ferramenta Signavio - Indicação do problema.



Fonte: O Autor

com uma mensagem que pode ser traduzida livremente como "Adicione fluxo de mensagem ausente ou cheque fluxo existente para correção". Essa mensagem indica que o evento de mensagens não possui fluxo e por isso não está sendo corretamente utilizado no processo.

Dessa forma a ferramenta avisa o usuário que possivelmente exista um problema no evento de mensagens, o que deve fazer com que o usuário reavalie o evento de mensagens. Nesse caso específico não é possível que a ferramenta imagine o que o usuário realmente precisaria fazer e por isso não é necessária uma indicação de que o evento de mensagem poderia na verdade ser dados do processo.

Dessa forma a ferramenta Signavio pode ser considerada correta já que apresentou uma mensagem que indique um problema no processo, não indicando exatamente o problema 5 pois não é necessário que esse seja o caso. Assim caso o usuário repare que utilizou incorretamente o evento de mensagens esse evento pode ser substituído por dados do processo.

Após testar todas as ferramentas podemos comparar as mesmas conforme figura abaixo. As ferramentas avaliadas com a cor vermelha foram consideradas incorretas enquanto a ferramenta avaliada com a cor verde foi considerada correta.

Conforme a tabela 4.9 observamos que a única ferramenta capaz de detectar o

Tabela 4.9: Problema 5 - Comparação Entre Ferramentas.

Problema/Ferramenta	Bizagi	Bonita	Camunda	Signavio
5				
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

problema 5 foi a ferramenta Signavio. A mesma apresentou um aviso que não foi apresentado por nenhuma das outras ferramentas e por isso é a única que pode indicar para o usuário quando o problema 5 pode estar acontecendo.

Nenhuma das outras três ferramentas apresentou nenhum tipo de aviso ou mensagem indicando que o evento de mensagem não se encontrava conectado a um fluxo de mensagem. Dessa forma o usuário poderia cometer o problema 5 e o processo gerado não poderia ser entendido por outros usuários além de possivelmente não gerar os dados necessários para o processo.

Na tabela 4.10 podemos observar as estatísticas para o problema 5. As ferramentas avaliadas como incorretas recebem 0 pontos enquanto as ferramentas avaliadas como corretas recebem 1 ponto para a soma.

Conforme podemos observar nas estatísticas da figura tabela 4.10 as ferramen-

Tabela 4.10: Problema 5 - Avaliação de Resultados

Problema/Resultados	Maior	Menor	Total	Media
5			1	0,25
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

tas não obtiveram um bom resultado para o problema 5. Mesmo uma ferramenta tendo conseguido apresentar a solução para o problema 5 corretamente, a somatória de todas as ferramentas foi apenas 1, o que representa uma média de 0,25 ou 25%. Essa é a menor média de todos os problemas apresentados até então.

Considerando que a média das ferramentas para esse problema foi baixa o analisador de processos a ser criado no próximo capítulo deve ter também como objetivo avisar esse problema já que o mesmo normalmente não é avisado pelas ferramentas.

Além disso o analisador pode considerar uma mensagem melhor do que as obser-

vadas nesse tópico, avisando da falta de conexão do evento de mensagem porém também avisando que possivelmente o evento de mensagem foi utilizado por engano quando na verdade a intenção do usuário poderia ser criar dados do processo.

4.6 Considerações Finais

O problema 6 não pode ser testado e avaliado pelas ferramentas do estado da arte já que o processo de detecção de um deadlock é complexo e não é apresentado por nenhuma das ferramentas. Mesmo ferramentas que fazem avaliação do fluxo de tokens como a ferramenta Signavio não apresentam nenhuma detecção de deadlock.

Considerando isso a detecção de deadlocks é útil para o analisador de processos já que as ferramentas comerciais testadas não tem a capacidade de detectar esse problema.

Por fim podemos analisar os resultados finais do capítulo resumidos nas tabelas 4.11 e 4.12. Na tabela 4.11 as linhas representam cada um dos problemas enquanto as colunas representam as ferramentas testadas. Um retângulo de cor verde representa uma avaliação correta para um devido problema por parte de determinada ferramenta, enquanto um retângulo amarelo representa uma avaliação parcialmente correta. Um retângulo vermelho representa uma avaliação incorreta.

Como podemos observar na tabela 4.11 não existiu um padrão entre as ferramentas. Nenhuma ferramenta foi capaz de solucionar todos os problemas e nenhuma das ferramentas não conseguiu solucionar nenhum problema.

Tabela 4.11: Todos os Problemas - Comparação Entre Ferramentas.

Problema/Ferramenta	Bizagi	Bonita	Camunda	Signavio
1				
2				
3				
4				
5				
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

Considerando que nenhuma ferramenta conseguiu solucionar todos os problemas

propostos pode-se avaliar que a criação de um analisador sintático que pudesse se conectar a essa ferramenta teria utilidade para todas as ferramentas testadas e estaria em conformidade com o estado da arte das ferramentas de modelagem atuais.

Além disso podemos observar abaixo as estatísticas de cada problema. Cada linha representa um problema enquanto as colunas representam o que está descrito em seu título. Para calcular o total foram somados os valores de todas as ferramentas para determinado problema, considerando um valor de 1 para uma resposta correta e 0,5 para uma resposta parcialmente correta. Respostas incorretas não somam pontos. Para calcular a média o total de pontos foi dividido pelo possível total caso todas as ferramentas apresentassem uma resposta correta, que no nosso caso é 4.

A tabela 4.12 apresenta resultados variados porém em gerais negativos para a

Tabela 4.12: Todos os Problemas - Estatísticas Gerais.

Problema/Resultados	Maior	Menor	Total	Media
1			1,5	0,375
2			2	0,5
3			1,5	0,375
4			4	1
5			1	0,25
Legenda	Correto	Parcialmente Correto		Incorreto
Cor				

Fonte: O Autor

maioria dos problemas. Somente um problema obteve uma média acima da metade, que foi o problema de número 4 que após avaliação anterior acabou excluído do analisador do capítulo seguinte por falta de necessidade. O problema de número 2 obteve uma média de metade e foi o melhor entre os problemas que serão resolvidos pelo analisador.

Mesmo a média da maioria dos problemas sendo abaixo da metade, todos os problemas tiveram ao menos uma ferramenta que o corrigiu completamente. Além disso somente o problema 4 não teve nenhuma ferramenta que não conseguiu corrigir o mesmo.

A avaliação das estatísticas indica que existe no mercado de softwares de modelagem atual uma necessidade de ferramentas adequadas para avaliar os problemas selecionados, tornando o analisador de processos importante para o estado da arte atual do campo. Além disso a possibilidade de estender o analisador para funcionar com qualquer uma dos ferramentas e solucionar mais problemas do que os apresentados torna o mesmo

uma ferramenta importante para esse campo.

O capítulo seguinte utilizara as conclusões de necessidades do estado da arte realizadas nesse capítulo para construir um analisador de processos capaz de detectar e avisar o usuário no caso dos problemas considerados relevantes por este capítulo.

5 DESENVOLVIMENTO

5.1 Descrição do Analisador de Processos e suas Estruturas

Utilizando o conhecimento obtido no capítulo 2 sobre o BPMN foram detectados 6 problemas distintos baseados em artigos de outros autores e foram propostas possíveis soluções no capítulo 3. A avaliação do estado da arte realizada no capítulo 4 detectou que a maioria dos problemas eram relevantes atualmente e não eram devidamente detectados pela maioria das ferramentas atuais.

Assim dos 6 problemas iniciais foram separados os 5 problemas nos quais as ferramentas atuais de modelagem tiveram dificuldade para avisar o usuário dos seus problemas. Esse capítulo apresenta um analisador de processos baseado nas regras do BPMN que avalia se os processos contêm um dos 5 problemas anteriormente citados.

Os primeiros passos para a criação do analisador foram definidos no capítulo 3, onde foi decidido que seria utilizado como entrada um arquivo .BPMN na linguagem de marcação XML. Esse arquivo descreve devidamente o processo e pode ser gerado em todas as ferramentas modeladoras testadas durante o capítulo 4.

Além disso, foi decidido a utilização da linguagem de programação C++ considerando os benefícios da mesma citados no capítulo 3. Como ferramenta de desenvolvimento (IDE) foi utilizado o Visual Studio 2017 que é desenvolvido pela Microsoft. Essa ferramenta foi escolhida baseada na capacidade de sincronização interna com o GIT, além da capacidade anterior do autor desse trabalho de utiliza-lá. Todo o código e a nomenclatura foram feitos em inglês para que o mesmo possa ser alterado por pessoas de todo o mundo no futuro caso venha a ser do interesse do autor.

A partir disso foi criada uma estrutura para conter e organizar os dados dos arquivos .BPMN que seriam lidos pela ferramenta. Considerando que um arquivo pode possuir mais do que um processo, foi criado uma classe denominada PAContainer que possui uma lista duplamente encadeada de processos.

O PAContainer só é criado uma vez por leitura de arquivo. Dentro da lista duplamente encadeada criada por ele são colocados todos os processos do arquivo do qual a leitura está sendo realizada. A classe que estrutura a lista encadeada é chamada de PADoublyLinkedList e a partir daqui será chamada simplesmente de PADLL.

A estrutura da classe PADLL é construída de forma que seja possível adicionar quantos processos a memória do sistema permita na lista, fazendo com que seja possível

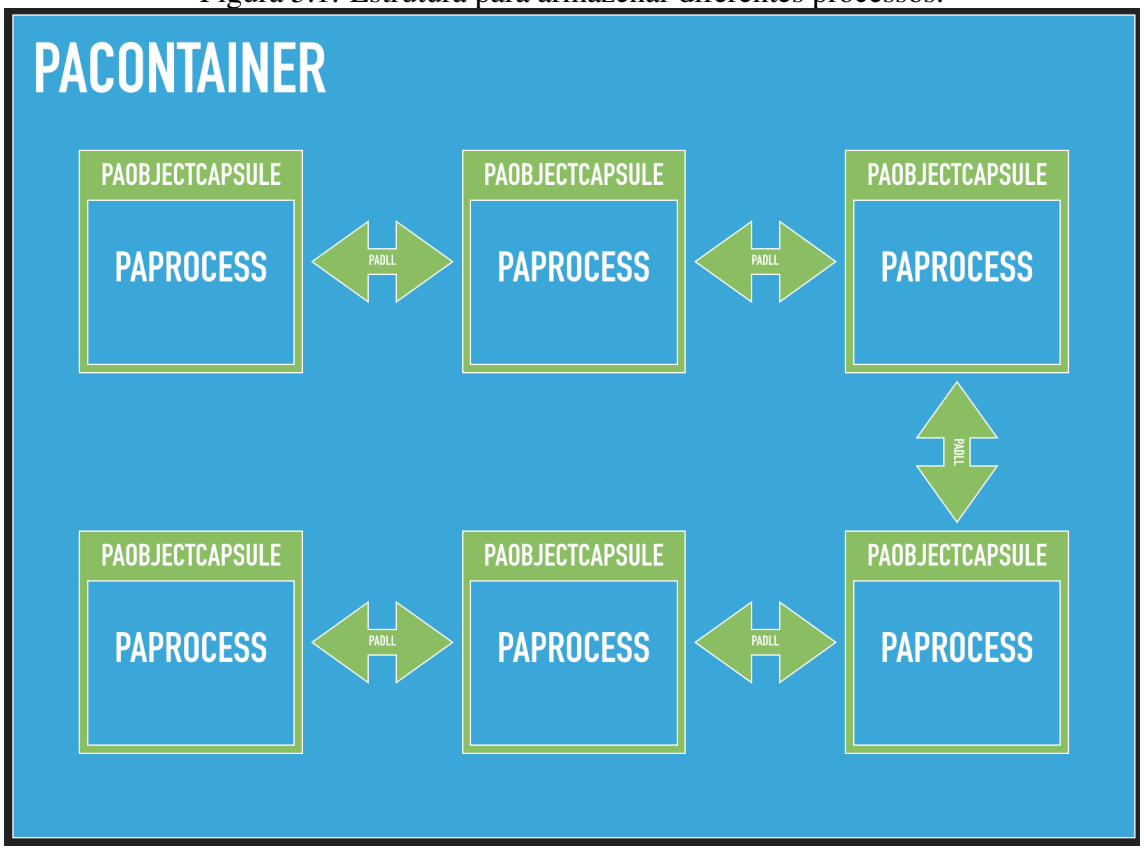
avaliar arquivos com muitos processos. Além disso, a memória só é alocada quando um novo processo é lido de forma que se o arquivo só tenha um processo a memória alocada será somente a necessária para esse.

Para possibilitar o correto reaproveitamento da classe PADLL também foi criada a classe PAObjectCapsule, que encapsula os processos do arquivo a ser lido. Essa classe possui um ponteiro ao objeto que guarda e dois ponteiros para outros PAObjectCapsule, sendo um o próximo da lista e um o anterior (que pode ser nulo no caso do objeto ser a raiz da lista). Para guardar os dados lidos sobre um processo foi criada a classe PAProcess que possui todos os dados do processo.

Podemos observar a estrutura criada para organizar os diferentes processos na figura 5.1. O quadrado maior azul é o objeto PAContainer criado somente uma vez. Dentro dele existem uma série de objetos PAObjectCapsule integrados pela PADLL. Dentro de cada objeto PAObjectCapsule existe um PAProcess.

A partir da estrutura apresentada na figura 5.1 é possível criar inúmeros processos

Figura 5.1: Estrutura para armazenar diferentes processos.



Fonte: O Autor

e tratar todos eles de forma separada. O analisador cria um processo e avalia o mesmo. Ao acabar o primeiro ele cria o segundo e o liga na lista, e assim por diante. A estrutura

criada é ideal para se manter persistente caso o analisador seja futuramente integrado a uma ferramenta de modelagem.

A classe PAProcess sempre que criada prepara um vetor com 11 posições, onde em cada posição existe uma lista duplamente encadeada que guarda um tipo de dado do processo. Cada uma dessas listas é necessária para o algoritmo para detectar os problemas de modelo de processos. As 11 listas duplamente encadeadas são definidas com os seguintes dados:

1. StartEvent
2. Task
3. SequenceFlow
4. ParallelGateway
5. ExclusiveGateway
6. InclusiveGateway
7. EndEvent
8. Event
9. Data
10. MessageFlow
11. IntermédiateThrowEvent

As listas são instanciadas vazias e aguardam que seja lido um dado do tipo específico para iniciar o preenchimento. São criadas listas pois alguns dos problemas podem ser resolvidos muito mais facilmente através dessas listas, como por exemplo a falta de um evento de início ou final. Para a resolução desses problemas com uma lista podemos simplesmente checar o tamanho da lista duplamente encadeada do processo relacionada a esses eventos.

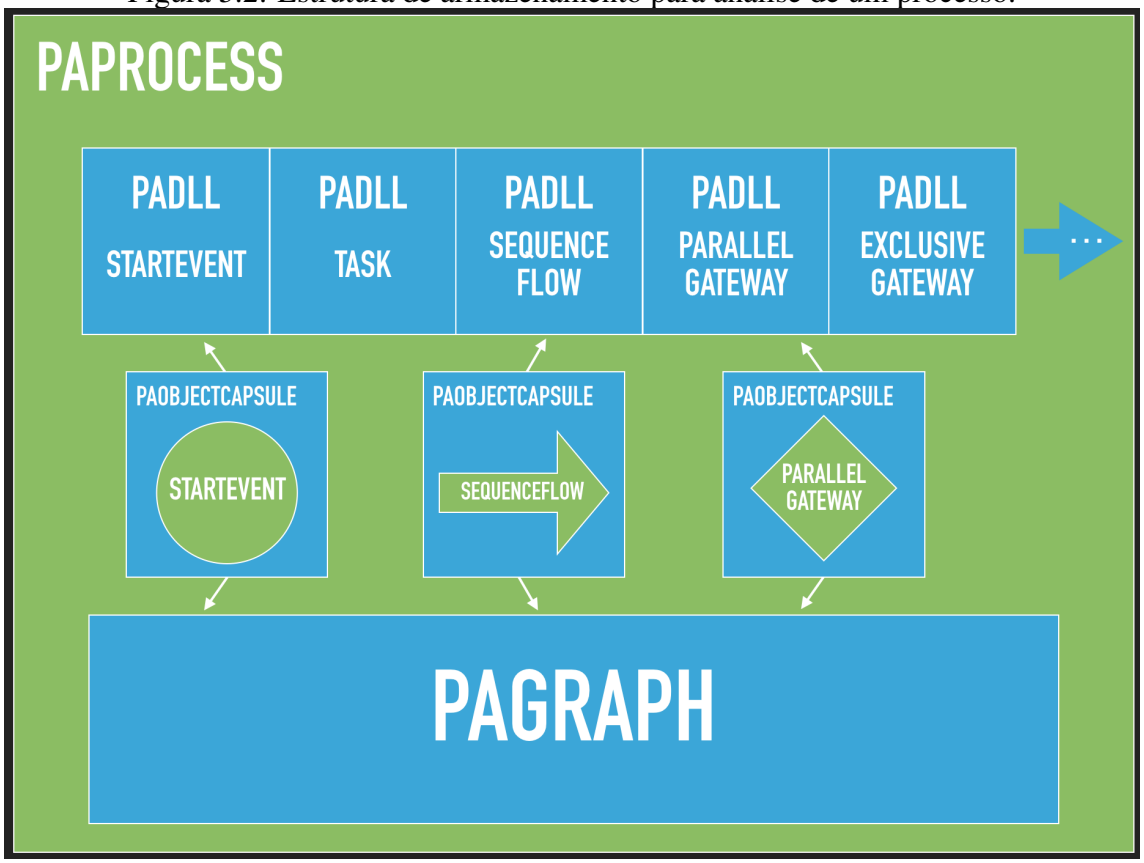
Além das listas duplamente encadeadas cada processo também cria um grafo para relacionar as partes do processo. Esse grafo só é criado após todo o arquivo ser lido e colocado nas respectivas listas encadeadas. Para não duplicar a memória utilizada o mesmo objeto de encapsulação (PAObjectCapsule) criado pela lista duplamente encadeada é utilizado pelo grafo, somente criando ligações diferentes (sem quebrar as anteriores).

Nesse caso duas atividades que não são ligadas pela estrutura do processo, estariam uma ligada a outra quando lidas pela lista duplamente encadeada que poderia acessar facilmente as atividades ou conta-las, porém não estariam ligados pelo grafo pois não tem uma ligação real na estrutura do processo. Dessa forma, é possível utilizar a segunda

forma de acesso, que é importante para resolver alguns dos problemas propostos no capítulo 3. Podemos observar a estrutura descrita na figura 5.2.

Na figura 5.2 o quadrado externo de cor verde representa uma instância da classe

Figura 5.2: Estrutura de armazenamento para análise de um processo.



Fonte: O Autor

PAProcess. Essa classe possuí um vetor de listas duplamente encadeadas representadas pelos quadrados azuis encostados uns nos outros. Cada quadrado é referente a um tipo de objeto, como mostra a figura.

No centro da figura podemos observar objetos na cor verde sendo encapsulados por instâncias da PAObjectCapsule na cor azul. Cada capsula é então incluída tanto na lista a qual o objeto se refere como também no grafo apresentado abaixo por uma instância da classe PAGraph.

A partir da organização de classes apresentada acima é possível guardar todos os dados de forma que a consulta seja feita da forma mais simples para cada problema. Essa definição permitiu algoritmos extremamente simples para os problemas 1 a 5, que puderam ser detectados sem necessidade de percorrer o grafo ou o XML inúmeras vezes. Dessa forma o analisador só precisa percorrer o XML uma única vez.

5.2 Detecção dos Problemas 1 a 4

Para cada problema só foram acessadas as listas encadeadas necessárias, de forma que também se torna muito fácil acrescentar novos problemas no analisador. Com essa estrutura só é necessário para o problema 1 (que acontece quando um processo não possui evento inicial) receber o tamanho da lista duplamente encadeada que contém os eventos iniciais. Se o tamanho da lista for zero, então não temos eventos iniciais e o processo pode definir que o problema 1 é existente.

Também utilizando o tamanho da lista duplamente encadeada que guarda os eventos iniciais podemos observar se o processo possui mais do que um evento inicial. Esse também não é um caso recomendado, apesar de permitido, e por isso o analisador avisa que o mesmo existe mesmo que não seja um dos problemas propostos nesse trabalho. A estrutura montada facilita a construção dessa checagem.

A figura 5.3 apresenta a lógica utilizada para reconhecer o problema 1 com a estrutura de dados criada.

Pseudocódigo Problema 1:

Figura 5.3: Resolução do problema 1

Fonte: O Autor

Se (PADDLEventoInicio.tamanho == 0) o problema ocorre.

Conforme podemos observar na figura 5.3, uma simples checagem do tamanho de uma lista duplamente encadeada conseguiu detectar o problema 1. Essa lógica também será aplicada no problema 3 apresentado nesse capítulo pois os dois problemas se apresentam de forma muito semelhante.

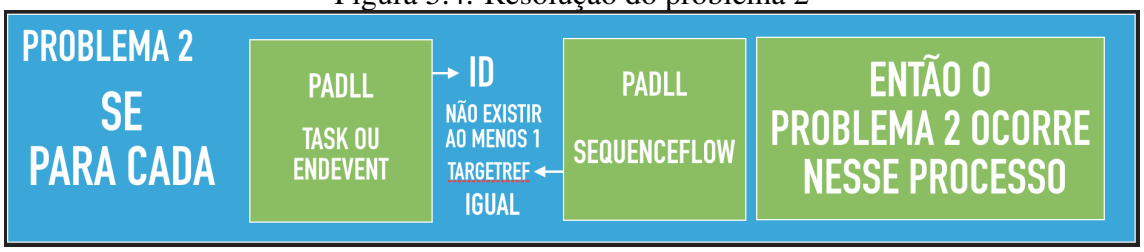
O problema 2 acontece quando um evento ou atividade não possui fluxo de entrada. Para resolver esse problema com a estrutura de classes atuais é necessário percorrer as listas duplamente encadeadas de eventos (menos as de eventos iniciais) e atividades. Para cada um dos eventos e atividades é necessário testar se na lista duplamente encadeada de fluxos *(sequence flows)* existe um fluxo que possui uma *targetRef* com o id do evento ou atividade.

O *targetRef* de um fluxo indica para qual objeto do processo o mesmo aponta. A partir do momento que um dos eventos ou atividades não possua nenhum fluxo apontando para si, podemos avaliar que esse processo contém o problema 2. Dessa forma o analisador percorre todos os eventos e objetos e após isso aponta o nome dos que não possuam nenhum fluxo, caso isso ocorra.

A figura 5.4 apresenta a lógica utilizada para reconhecer o problema 2 com a estrutura de dados criada.

Pseudocódigo Problema 2:

Figura 5.4: Resolução do problema 2



Fonte: O Autor

Para (inteiro numeroTarefa = 0, numeroTarefa < PADLLTarefa.tamanho , numeroTarefa++)

PADLLTarefa[numeroTarefa].problema2 = verdadeiro

Para (inteiro numeroFluxo = 0, numeroFluxo < PADLLFluxo.tamanho, numeroFluxo++)

Se (PADLLTarefa[numeroTarefa].id == PADLLFluxo[numeroFluxo].targetRef)

PADLLTarefa[numeroTarefa].problema2 = falso

Podemos observar na figura 5.4 como o analisador resolve o problema 2 comparando o id de cada evento e atividade com o targetRef apresentado pelos fluxos *(sequence flows)*. Tanto os eventos e atividades quanto os fluxos são armazenados dentro de listas duplamente encadeadas distintas.

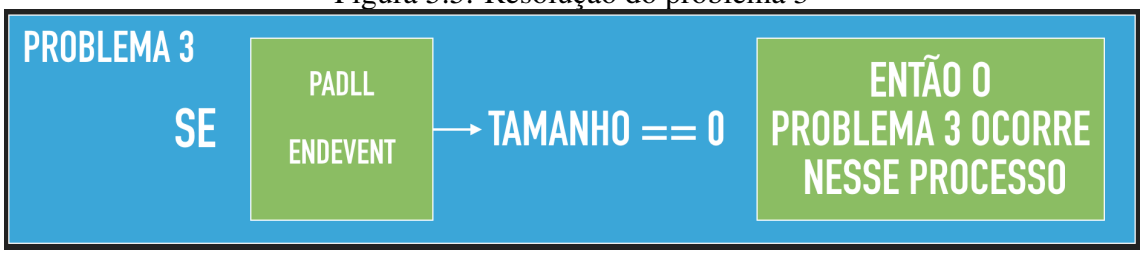
Para detectar o problema 3 apresentado nesse trabalho (falta de evento final no processo) podemos utilizar praticamente os mesmos passos apresentados para o problema 1, porém substituindo a lista duplamente encadeada de eventos iniciais pela lista de eventos duplamente encadeada de eventos finais.

Nesse caso o analisador solicita o tamanho da lista duplamente encadeada de eventos finais e caso o tamanho da mesma seja zero ele considera que o problema 3 acontece nesse processo.

Podemos observar na figura 5.5 a resolução do problema de número 3 considerando a estrutura de dados já gerada pelo analisador na leitura do arquivo XML.

Pseudocódigo Problema 3:

Figura 5.5: Resolução do problema 3



Fonte: O Autor

Se (PADDLEventoFinal.tamanho == 0) o problema ocorre.

A figura 5.5 apresenta uma possível maneira de resolver o problema de falta de um evento final que foi encontrada com a estrutura de dados gerada. Além disso no caso de eventos finais a ferramenta também alerta caso existam mais de um evento final pois múltiplos eventos finais podem não ser recomendados. Apesar disso esse caso é permitido pelas regras do BPMN e o aviso é só uma recomendação.

Conforme definido no capítulo anterior o problema 4 não será analisado considerando que todas as ferramentas testadas foram capazes de evitar que esse problema ocorra. Além disso nenhuma ferramenta é capaz de exportar um arquivo .BPMN que apresente esse problema pois todas evitam o mesmo antes que ele aconteça. Assim nenhum teste poderia ser realizado de forma a examinar o funcionamento do algoritmo para a solução do problema 4.

Para a resolução do problema 5 (dados do processo podem ter sido apresentados como um evento de mensagem) foram consideradas as mesmas características observadas pela ferramenta de modelagem que detectou o problema no capítulo anterior. Dessa forma o aviso acontece quando existe um evento de mensagem que não possui um fluxo de mensagem que inicia ou termina no mesmo.

Essa abordagem é confirmada pelas regras atualizadas do BPMN que especificam a necessidade de que cada evento de mensagem possua um fluxo de mensagem conectado ao mesmo(GROUP, 2013). Para essa detecção são percorridos todos os eventos intermediários que deveriam enviar mensagens (no XML definidos como IntermédiateThrowEvent). Para cada um desses eventos é testado se o id do mesmo se apresenta como o início de um fluxo de mensagens (sourceRef de um MessageFlow).

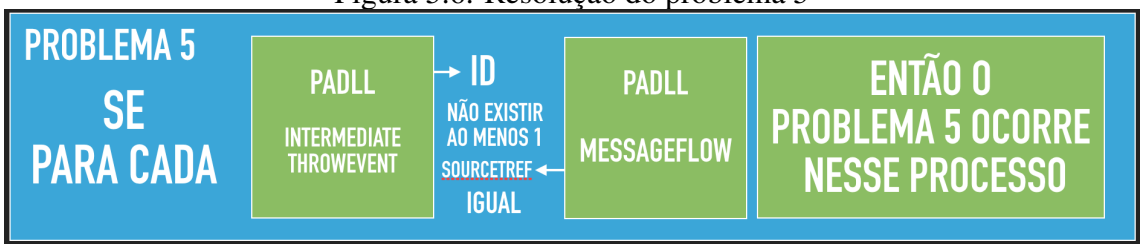
Caso um dos eventos de mensagem não possua nenhum fluxo de mensagem co-

nectado, o problema 5 é considerado detectado e é apresentado um aviso ao usuário. Esse aviso apresenta uma mensagem definindo que o evento em questão não possui nenhum fluxo de mensagem provindo do mesmo e que esse fato pode ocorrer pois na verdade o usuário gostaria de criar dados no processo.

A solução do problema 5 se encontra apresentada de forma gráfica na figura 5.6.

Pseudocódigo Problema 5:

Figura 5.6: Resolução do problema 5



Fonte: O Autor

Para (inteiro numeroEvento = 0, numeroEvento < PADLLEventoIntermédiario.tamanho , numeroEvento++)

PADLLEventoIntermédiario[numeroEvento].problema5 = verdadeiro

Para (inteiro numeroFluxoMensagem = 0, numeroFluxoMensagem < PADLLFluxoMensagem.tamanho, numeroFluxoMensagem++)

Se (PADLLEventoIntermédiario[numeroEvento].id == PADLLFluxoMensagem[numeroFluxoMensagem].targetRef)

PADLLEventoIntermédiario[numeroEvento].problema5 = falso

É possível visualizar na figura 5.6 que a lista duplamente encadeada de eventos de mensagem é percorrida e para cada evento de mensagem o id do mesmo é comparado com a entrada de cada fluxo de mensagem. Dessa forma qualquer evento de mensagem que não possui fluxo de mensagem pode ser detectado, já que não estará presente na entrada de nenhum fluxo de mensagem.

5.3 Detecção do Problema 6

A detecção do problema 6 (possibilidade de deadlock no processo) é a mais complexa apresentada nesse documento. Ela é a única detecção que necessita da utilização do grafo. Caso não existam eventos iniciais ou finais esse teste não é realizado pois o deadlock não poderia ser detectado sem esses eventos.

O primeiro passo para a detecção é gerar um token iniciando no evento inicial no grafo. Caso existam mais de um evento inicial várias instâncias do grafo são criadas, cada uma com um token em um dos eventos iniciais. A partir desse ponto os tokens avançam percorrendo o grafo e só são reavaliados quando encontram um gateway ou evento final.

Ao encontrar um gateway paralelo divergente (que possui uma entrada e várias saídas) o token é dividido em vários tokens diferentes porém o grafo continua possuindo somente as mesmas instâncias que já existiam. Cada um dos tokens continua seguindo um dos caminhos do grafo. Quando um token encontra um gateway exclusivo divergente vários tokens são criados, um para cada possibilidade de saída, porém ao contrário do gateway paralelo são criadas o mesmo número de instâncias e somente um token é enviado por cada instância.

Essa diferença acontece pois em um gateway paralelo cada instância do processo segue para todas as saídas do gateway ao mesmo tempo na mesma instância, enquanto em um gateway exclusivo o token segue somente um dos caminhos. Para testar todas as opções no gateway exclusivo é necessário criar várias instâncias fazendo com que um dos tokens siga para cada uma delas, dessa forma estaríamos testando diferentes processos, cada um seguindo um caminho diferente a partir do gateway.

Quando um token encontra um gateway paralelo convergente, pelas regras do BPMN ele deve esperar que um token chegue por cada caminho conectado ao gateway para poder continuar. Dessa forma quando vários tokens se reúnem completando um gateway paralelo eles são novamente unidos e continuam a percorrer o grafo.

Um token que encontra um gateway exclusivo convergente pode continuar percorrendo o grafo. Quando um token encontra um evento final o mesmo é retirado da contagem de tokens daquela instancia do grafo. Nesse momento é testado se não existem mais tokens no grafo (que possui uma variável que guarda esse valor). Caso não existam outros tokens a instância é dada como concluída corretamente.

Caso existam outros tokens percorrendo o grafo, eles continuam percorrendo até que todos eles se encontrem em espera em gateways paralelos convergentes (existe uma variável que conta quantos tokens estão em espera nesse tipo de gateway). Caso o valor de tokens nesse tipo de gateway seja a quantidade total presente no grafo, porém ao mesmo tempo nenhum desses tokens possa ser unidos e liberados adiante (pois não são o suficiente para completar a quantidade necessária de cada gateway) o processo é considerado com possibilidade de deadlock.

Somente uma instância com possibilidade de deadlock é necessária para que o pro-

cesso seja considerado com essa possibilidade. Para que o processo seja considerado sem possibilidade de deadlock todas as instâncias criadas devem ter concluído com sucesso. A lógica de detecção de um deadlock passa basicamente por testar todas as possibilidades de curso do processo e considerar se em alguma delas um token não consegue encontrar um evento final por ficar preso em um gateway paralelo convergente.

5.4 Utilização e Avisos

Completando a detecção do problema 6, todos os problemas propostos no capítulo 3 e avaliados durante o capítulo 4 como relevantes para o estado da arte atual foram devidamente resolvidos e podem ser detectados e avisados ao usuário. O analisador de processos pode ser utilizado por linha de comando indicando assim o arquivo .BPMN que deve ser avaliado.

O arquivo em questão deve ser um XML válido, preferencialmente exportado de uma ferramenta de modelagem de processos e é avaliado conforme a especificação da BPMN(GROUP, 2013). Quando a avaliação for concluída será apresentado ao usuário os problemas detectados com as seguintes mensagens:

1. "O processo (nome do processo) não contém nenhum evento inicial. É recomendada a inserção de um evento inicial no processo. Não será possível avaliar a existência de deadlocks sem um evento inicial."
2. "O processo (nome do processo) possui contém a atividade (nome da atividade) que não possui fluxo de entrada. Uma atividade sem fluxo de entrada pode não ser realizada."
3. "O processo (nome do processo) não contém nenhum evento final. É recomendada a inserção de um evento final no processo. Não será possível avaliar a existência de deadlocks sem um evento final."
4. "O processo (nome do processo) possui um evento de mensagens chamado (nome do evento de mensagens) que não está conectado a um fluxo de mensagens. É necessário que todos os eventos de mensagens estejam conectados a um fluxo. Confira se não deveria criar dados do processo ao invés de um evento de mensagem."
5. "O processo (nome do processo) possui a possibilidade de deadlock. No momento do deadlock existiam tokens aguardando infinitamente nos gateway(s) chamado(s)

(nome do(s) gateway(s)). Considere substituir os gateways em questão por gateways OR (representados por um diamante com um círculo interno)."

6 CONCLUSÃO

O trabalho apresentado construiu um analisador de modelos de processos e avaliou a necessidade do mesmo no mercado de ferramentas atuais para modelagem de processos. Considerando os resultados obtidos no capítulo 4 podemos observar que o analisador de modelo de processos pode ajudar todos as ferramentas existentes na detecção de problemas. O analisador de processos construído possui a capacidade de resolver todos problemas propostos nesse trabalho no capítulo 3 e considerados relevantes pelo capítulo 4. O analisador pode ser utilizado com arquivos .BPMN exportados de qualquer ferramenta modeladora de processos (desde que os mesmo sigam as regras atuais do formato de arquivos) e por isso pode ser considerado relevante para o estado da arte atual.

Além disso, o analisador de modelos de processo pode ser expandido facilmente para implementar a correção de outros problemas considerando que ele possui uma estrutura de dados que facilita a maioria das detecções. Considerando também a necessidade de uma melhor detecção de erros nas ferramentas de modelagem atual o analisador poderia ser transformado em um framework para essas ferramentas.

As estruturas citadas no capítulo 5 não possuem até o momento a possibilidade de exclusão dos dados criados, e por isso se mantêm persistentes na leitura de um arquivo. Dessa forma toda vez que um arquivo é modificado é necessária a leitura de todo o arquivo novamente e por isso o analisador de modelos de processos não é recomendado em sua forma atual para um acompanhamento direto de ferramentas, atualizando a cada mudança. Caso fosse utilizado nessa função o analisador necessitaria ler o arquivo a cada mudança, o que não é ideal.

A criação de trabalhos futuros a partir desse trabalho são possíveis principalmente na área de expansão das funções do analisador de modelos de processos, incluindo novos problemas considerados relevantes para as ferramentas de modelagem de processos atuais. Além disso a transformação do analisador de modelos de processo em um programa que não seja executado por linha de comando pode colaborar para que o trabalho seja mais difundido e utilizado fora do meio acadêmico.

Os testes para confirmação desse trabalho foram realizados utilizando os processos gerados no capítulo 4 desse trabalho apresentados para avaliação das ferramentas de modelagem do mercado. Dessa forma os testes utilizaram as mesmas circunstâncias que as ferramentas de modelagem para detectar os problemas considerados a partir do capítulo 3.

Dessa forma o trabalho proposto no capítulo 3 foi devidamente concluído e os problemas de modelagem de processos considerados nesse capítulo e avaliados no capítulo 4 foram devidamente detectados pelo analisador de modelos de processo. O analisador de modelos de processos construído no capítulo 5 possui uma estrutura de dados atual e de fácil expansão, tornando esse trabalho uma contribuição viável para a área de análise de problemas em modelagem de processos.

REFERÊNCIAS

AVILA, D. T. Process modeling guidelines: Systematic literature review and experiment. **Universidade Federal do Rio Grande do Sul**, 2018.

BAZHENOVA, E.; TARATUKHIN, V.; BECKER, J. Impact of information and communication technologies on business process management on small and medium enterprises in the emerging countries. **Proceedings of the 11th International Conference of Perspectives in Business Informatics Research Nizhny Novgorod Russia**, 2012.

BIZAGI. **Bizagi BPMN Modeler**. 2019. Available from Internet: <https://www.bizagi.com/pt/produtos/bpm-suite/modeler>.

DANI, V. S. Feedback visual sobre problemas em modelos de processos de negócio: revisão sistemática da literatura, survey, estudos de caso e recomendações. 2019.

DIAS, C. L. de B. Análise de comportamento de ferramentas de modelagem de processos com base em anti padrões. 2018.

DUMAS, M. et al. **Fundamentals of Business Process Management**. [S.l.]: Springer-Verlag Berlin Heidelberg, 2013. ISBN 9783642331428.

GMBH, C. S. **Camunda BPM Workflow and Decision Automation Platform**. 2019. Available from Internet: <https://camunda.com/>.

GROUP, O. M. Business process model and notation (bpmn) version 2.0.2. 2013.

KROGSTIE, J. **Model-Based Development and Evolution of Information Systems: A Quality Approach**. [S.l.]: Springer-Verlag London, 2012. ISBN 9781447129356.

MENDLING, J.; REIJERS, H.; AALST, W. van der. Seven process modeling guidelines (7pmg). **Queensland University of Technology and Eindhoven University of Technology**, 2008.

PRECHELT, L. An empirical comparison of seven programming languages. **University of Karlsruhe**, 2000.

ROZMAN, T.; POLANčIč, G.; HORVAT, R. V. Analysis of most common process modelling mistakes in bpmn process models. **University of Maribor, Slovenia**, 2008.

S.A., B. **Bonitasoft Open-source Business Process automation Platform**. 2019. Available from Internet: <https://www.bonitasoft.com/>.

SIGNAVIO. **Signavio Process Manager BPM Platform for Process Modeling**. 2019. Available from Internet: <https://www.signavio.com/products/process-manager/>.