Geração de Cenários de Testes a partir de Modelos em BPMN para Aplicações Web criadas com Sistemas de Gerenciamento de Processos de Negócio

Generation of Test Scenarios from BPMN Models for Web Applications created with Business Process Management Systems

Jéssica L. de Moura
Universidade Federal de Santa Maria
jmoura@inf.ufsm.br

Andrea S. Charão
Universidade Federal de Santa Maria
andrea@inf.ufsm.br

João Carlos D. Lima
Universidade Federal de Santa Maria
caio@inf.ufsm.br

RESUMO

Sistemas de BPM (*Business Process Management*) têm facilitado a rápida produção de aplicações Web que executam processos expressos em BPMN. O teste automatizado de tais aplicações, no entanto, continua sendo um desafio. Neste artigo, propõe-se uma abordagem para geração de cenários para testes para aplicações Web implementadas com o apoio de BPMS, a partir de modelos BPMN. A abordagem visa abreviar o esforço de construção de *scripts* de teste e é focada em testes funcionais, usando as ferramentas Selenium e Cucumber. A abordagem foi aplicada a processos encontrados em diferentes repositórios e mostrou-se capaz de gerar os cenários desejados.

Palavras-Chave

BPM.BPMN.Teste funcional.Teste automatizado.

ABSTRACT

Business Process Management (BPM) systems have facilitated the rapid production of web applications that execute processes expressed in BPMN. Automated testing of such applications, however, remains a challenge. In this article, we propose an approach for generating scenarios for testing Web applications implemented with the support of BPMS, from BPMN models. The approach is intended to abbreviate the effort to construct test scripts and is focused on functional testing using the Selenium and Cucumber tools. The approach was applied to processes found in different repositories and was able to generate the expected scenarios.

SBSI 2017 June 5^{th} – 8^{th}, 2017, Lavras, Minas Gerais, Brazil

CCS Concepts

•Information systems → Information systems applications; •Applied computing → Business process management; •Software and its engineering → Software testing and debugging;

Keywords

BPM.BPMN.Functional testing.Automated testing.

1. INTRODUÇÃO

Em modernos sistemas de informação, o gerenciamento de processos de negócio (*Business Process Management – BPM*) tem suscitado o interesse de empresários e da comunidade científica, tanto por seus benefícios como por seus desafios. Designa-se por BPM o conjunto de conceitos, métodos e técnicas para suportar a análise, modelagem, execução, monitoramento e otimização dos processos de negócio [29]. Nesse contexto, surgiram os sistemas de BPM (*Business Process Management Systems* ou *Suites* – BPMS), que tipicamente oferecem recursos para definição e modelagem de processos em BPMN (*Business Process Model and Notation*), controle da execução e monitoramento de atividades dos processos [11]. Os BPMS tendem a abreviar o desenvolvimento de software [31], gerando aplicações Web que executam os processos expressos em BPMN.

As aplicações geradas com auxílio de BPMS estão sujeitas a defeitos durante a execução de um processo. Por exemplo, uma entrada não tratada em um formulário pode causar um erro e abortar o processo. Testes de software podem ser empregados neste contexto, mas percebe-se que, na comunidade interessada em BPM, a pesquisa em verificação e testes costuma se concentrar na etapa de modelagem e não de execução [28].

Em um trabalho anterior [8], buscou-se explorar este tema realizando testes automatizados em aplicações Web para execução de um mesmo processo, implementadas com o auxílio de dois BPMS *open source* distintos: Bonita BPMS e Activiti. Os resultados revelaram dificuldades já levantadas por outros autores interessados em testes automatizados [22, 30], relacionadas ao tempo necessário para escolha e configuração de ferramentas de teste. No caso específico das aplicações de BPMS, notou-se que a criação de *scripts* de teste

para processos com muitas tarefas poderia exigir um esforço demasiado.

Neste contexto, o presente trabalho propõe uma abordagem para geração de cenários para testes a partir de modelos BPMN, visando abreviar o esforço de construção de *scripts* de teste. O trabalho é focado principalmente em testes funcionais e tem como objetivos específicos: (i) gerar uma tabela de caminhos de execução da aplicação a partir da análise de fluxos no modelo BPMN e (ii) gerar o código inicial dos cenários de teste, a serem executados em um dado arcabouço de teste automatizado de aplicações Web. O restante do texto está organizado como segue: na seção 2, apresenta-se a fundamentação do trabalho, que apoia-se em dois pilares: BPM e testes automatizados; na seção 3, discute-se trabalhos relacionados que associam BPM e testes; na seção 4, apresenta-se a abordagem proposta e suas particularidades; na seção 5, apresenta-se uma avaliação da abordagem e, por fim, na seção 6, apresentam-se considerações finais sobre o trabalho.

2. FUNDAMENTAÇÃO

2.1 Termos e Conceitos Associados a BPM

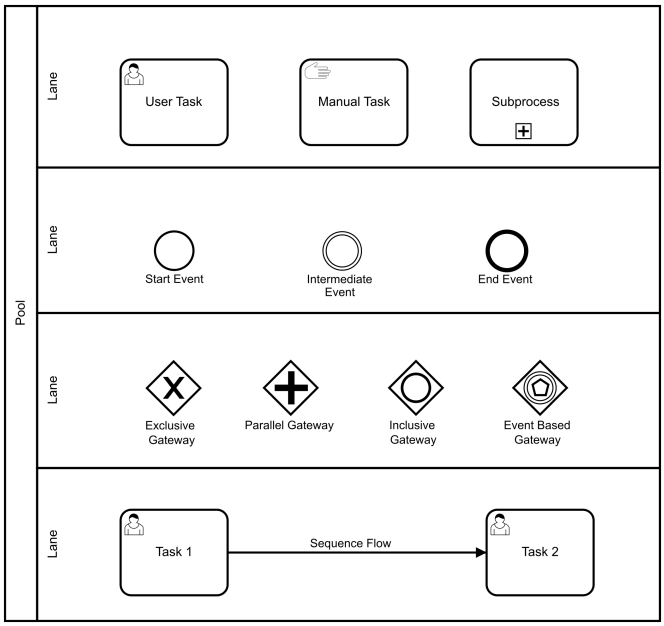
O termo BPM pode ser usado com ênfases diferentes, dependendo do contexto. Em geral, é um termo usual em Administração e Engenharia de Produção, embora também seja de importância para as áreas de Computação e Tecnologia da Informação [23]. No suporte a BPM, os sistemas de BPM (BPMS) têm se afirmado como ferramentas essenciais. Desde as origens do termo BPMS [26], surgiu uma ampla diversidade de sistemas no mercado, incluindo desde BPMS proprietários de grandes fabricantes de software como IBM, SAP e Oracle, até sistemas com versões de código aberto, como por exemplo Bonita BPM, Activiti e Camunda. Tais ferramentas oferecem recursos para modelagem, controle de execução e monitoramento de processos, cada uma com suas especificidades. Esses e outros BPMS possuem atualmente um aspecto em comum: a representação e exportação de processos em BPMN.

O *Business Process Model and Notation*, ou BPMN, foi criado com o objetivo de "*fornecer uma notação facilmente compreensível por todos os usuários, desde os analistas que criam os rascunhos iniciais dos processos até os desenvolvedores responsáveis por implementar os processos e, finalmente, para os usuários que irão gerenciar e monitorar esses processos*" [19]. A versão BPMN 2.0, a mais recente, define um padrão XML para arquivos contendo dados sobre o modelo e o funcionamento do processo, bem como a sua representação visual [16]. Isto permite que o XML seja analisado para obter-se informações sobre os processos. Em BPMN, um processo é descrito como um diagrama de elementos de fluxo. Os principais elementos de fluxo que compoem um diagrama de um processo podem ser divididos entre: Tasks (Tarefas), Events (Eventos), Gateways e Sequence Flows. Na Figura 1, podem ser vistos alguns elementos que compõem um diagrama BPMN e que serão importantes para este trabalho.

As *Tasks*, que também podem ser chamadas de *Tarefas*, constituem a menor parte de um diagrama em BPMN, sendo utilizadas quando o diagrama não pode ser dividido mais detalhadamente. Um elemento do tipo *SubProcess* é uma abstração de um pequeno conjunto de *Tasks*.

Existem diferentes tipos de *Tasks* para representar diferentes comportamentos. Por exemplo: uma *Script Task* pode executar um *script* criado em uma linguagem que o BPMS pode interpretar, uma *Service Task* pode usar algum tipo de serviço e uma *Business Rule Task* fornece um mecanismo para o processo fornecer entrada a uma regra de negócio.

Figura 1: Principais elementos de um processo de acordo com o padrão BPMN.



Para esse trabalho os tipos de *tasks* mais relevantes são as tarefas *User Task*, pois essas tarefas precisam da execução do usuário para serem finalizadas, ou seja, são efetivamente executadas por um usuário. Uma *User Task* geralmente é finalizada após a execução de um formulário, enquanto uma *Manual Task*, por exemplo, é executada sem a ajuda de uma aplicação.

Um *Event* é algo que "acontece" durante o curso de um processo [19]. Existem três tipos principais de eventos: *Start Event*, *End Event* e *Intermediate Event*. Um *Start Event* indica o início de um processo e um *End Event* indica o fim de um processo. Um *Intermediate Event* indica um acontecimento entre o início e o fim de um processo – este tipo de evento pode ser usado, por exemplo, para indicar atrasos no processo.

Gateways são usados para controlar o fluxo do processo. Um *Gateway* implica que haverá uma "decisão" que será tomada para habilitar ou desabilitar um fluxo. Um *Gateway* pode indicar a divisão ou a união dos fluxos. Em *Gateways* do tipo *Exclusive*, apenas um dos fluxos que partem do *Gateway* poderão ser seguidos, já em *Gateways* do tipo *Inclusive* um ou mais fluxos podem ser seguidos, dependendo da decisão tomada. Em *Gateways* do tipo *Parallel*, todos os fluxos serão executados. Em um *Event-Based Gateway*, um ou mais caminhos poderão ser executados, mas a decisão depende de um evento como, por exemplo, um recebimento de uma mensagem.

Um *Sequence Flow* é usado para exibir a ordem em que os demais elementos são executados em um processo. Cada *Sequence Flow* possui apenas uma origem (ou fonte) e um

destino (ou alvo), que podem ser *Tasks*, *Events* ou *Gateways*. Analisando os elementos *Sequence Flow* de um diagrama, é possível identificar todo o fluxo de execução do processo.

Um *Pool* é um elemento BPMN que define os limites de um processo, um *Pool* pode conter apenas um processo de negócio. Uma *Lane* geralmente é utilizada para organizar os elementos utilizados no processo (tarefas, eventos, etc) possibilitando identificar quem será responsável pela execução de cada *Task* do processo. Um *Pool* pode conter quantas *Lanes* forem necessárias para identificar os envolvidos na execução das *Tasks*.

2.2 Testes Automatizados de Aplicações Web

Testes automatizados têm como propósito a aplicação de estratégias e ferramentas tendo em vista a redução de trabalho manual em testes de aplicações [14]. O teste manual de um software pode permitir encontrar vários erros em uma aplicação, mas também pode exigir um grande trabalho e gasto de tempo. A automação dos testes permite que o teste seja repetido, principalmente durante os estágios de desenvolvimento e integração, onde os *scripts* podem ser executados um grande número de vezes, oferecendo um retorno significativo [10]. Uma vez automatizado, um grande número de casos de teste podem ser validados rapidamente.

A relevância crescente das aplicações Web aumentou a importância de controlar e melhorar a sua qualidade, principalmente através de metodologias e ferramentas de teste [24]. No entanto, os testes automatizados de aplicações Web revelaram novos desafios, devido à sua natureza distribuída, a heterogeneidade e a dinamicidade das aplicações [12]. Estas características estão presentes em grande parte das aplicações de BPM construídas com o apoio de BPMS, uma vez que a maioria destes sistemas é voltado para a Web. Porém, testes automatizados de software não fazem parte dos recursos comumente oferecidos pela maioria dos BPMS, restando assim a opção de se utilizar ferramentas de testes automatizados voltadas para aplicações Web em geral.

A automação de testes pode envolver tanto testes funcionais, para entender como o sistema se comportaria durante a navegação de um usuário, ou testes não-funcionais, que podem validar requisitos não funcionais relacionados ao uso do sistema como desempenho e disponibilidade [15, 6, 25]. Neste trabalho, escolheu-se abordar o teste funcional associando as ferramentas Selenium e Cucumber, por serem bastante difundidas entre profissionais e também em trabalhos acadêmicos [20].

O processo para a realização de um teste funcional utilizando o Selenium aliado ao Cucumber é composto de cinco etapas: captura da interação do usuário com a aplicação e exportação do código gerado, criação dos cenários de teste, criação das definições dos passos de teste, implementação dos métodos para cada passo e, por fim, execução do teste.

Para efetuar a captura da interação do usuário com a aplicação é utilizado o Selenium IDE (*Integrated Development Environment*), que permite gravar as ações do usuário conforme elas são executadas. Assim, a funcionalidade que se deseja testar deve ser executada para que os passos sejam gravados. Após a captura da interação é possível exportar o *script* utilizando diversas linguagens de programação disponíveis. No presente trabalho, como forma de delimitá-lo, utiliza-se somente a linguagem Java.

Os testes utilizando Cucumber são compostos, basicamente, por dois arquivos: arquivos que especificam as funcionalidades (*features*) e por arquivos de definição de passos (*steps*). Os arquivos com as funcionalidades (*features*) são escritos na linguagem Gherkin e são compostos por cenários. Os cenários representam uma fração da aplicação que vai ser testada. Um cenário também pode ser descrito como a definição, em ordem de execução, das etapas que são executadas nessa fração, bem como dos resultados esperados para validar a aplicação. Por exemplo, após um *Gateway* exclusivo com dois fluxos disponíveis, tem-se dois cenários possíveis, um cenário executando cada fluxo.

Para que cada etapa do cenário seja executada, é necessária a criação de *steps* que irão traduzir os passos definidos na linguagem Gherkin para ações que vão interagir com o sistema. Cada *step*, geralmente, invocará um método em Java que irá efetivamente executar a interação com a aplicação. A implementação destes métodos pode ser feita apenas utilizando o código extraído através do Selenium após a captura das ações do usuário.

Assim, mesmo utilizando Cucumber e Selenium para facilitar a criação dos testes, ainda é preciso analisar o processo e extrair os cenários de teste que devem ser criados manualmente no arquivo de *features*. Além dos cenários, também deve ser criado manualmente o arquivo de *steps*, contendo um "passo" correspondente a cada etapa de todos os cenários criados. Esta etapa pode ser trabalhosa, principalmente quando for necessário testar diversos cenários que um processo pode ter ou quando o processo for extenso.

3. TRABALHOS RELACIONADOS

Os sistemas BPM geralmente oferecerem recursos para definição e modelagem de processos em BPMN, controle da execução e monitoramento de atividades dos processos [11]. Nota-se, no entanto, que a preocupação com testes não fica evidente nas ferramentas BPMS e em trabalhos acadêmicos. De fato, examinando-se o material promocional, documentação disponível sobre os principais BPMS e analisando trabalhos da área, observa-se uma ênfase em etapas de modelagem, execução e monitoramento dos processos. No entanto, aplicações de BPM também estão sujeitas a defeitos e, por isso, podem se beneficiar de avanços na área de testes de software.

Existem alguns trabalhos que abordam diferentes aspectos do teste de aplicações relacionados com BPM [5, 17, 18]. Seja explorando abordagens para facilitar a seleção de casos de teste de processos de negócio [5], ou abordando testes como o teste de unidade ou teste de regressão de diferentes tecnologias relacionadas à BPM como BPEL (*Business Process Execution Language*) ou SOA (*Service Oriented Architecture*) [17, 18], estes trabalhos reforçam a importância de testes direcionados ao Gerenciamento de processos de Negócio. Os trabalhos encontrados se relacionam com o presente trabalho do ponto de vista da realização de testes pois, além de fortalecerem a importância da execução de testes, chamam atenção para suas dificuldades, corroborando para a ideia de gerar automaticamente elementos para o teste.

Model-based testing ou Teste baseado em modelo é uma técnica para criar elementos para executar o teste de aplicações através de modelos em UML, máquinas de estado ou algum outro modelo formal da aplicação [9]. Trabalhos já apontaram os benefícios relativos a essa abordagem, principalmente benefícios relativos a diminuição de custos, aumento de qualidade e maior detecção de falhas [1, 21]. Geralmente os trabalhos [7, 27, 2] sobre o assunto focam na

análise de modelos baseados em UML ou semelhantes.

Existem trabalhos que abordam o conceito de *Model-based testing* com outros tipos de notações, como a obtenção de casos de teste a partir de diagramas de sequência representados como uma sequência de chamadas de métodos [13]. No entanto, poucos trabalhos em *Model-based testing* abordam a fundo a automação e execução dos casos ou elementos de teste criados.

Não foram encontrados trabalhos que abordassem a análise de modelos BPMN para gerar elementos para o teste e, neste ponto, o presente trabalho se destaca. O presente trabalho também tem o diferencial de abordar a versatilidade dos casos criados, analisando a possibilidade de execução através de diferentes ferramentas. No entanto, existem trabalhos [3, 4] que abordam a obtenção de outros elementos como requisitos, por exemplo, através de modelos em BPMN.

4. ABORDAGEM PROPOSTA

O objetivo deste trabalho é gerar código para testes funcionais automatizados de aplicações Web construídas com o apoio de um BPMS. Neste sentido, um aspecto importante para o teste funcional é identificar quais fluxos ou etapas serão testadas, pois estes determinam caminhos de execução da aplicação e podem definir ou limitar o teste, interferindo assim na qualidade e na cobertura dos testes.

Com o objetivo de obter informações capazes de auxiliar na identificação dos fluxos, projetou-se e implementou-se uma ferramenta que analisa os arquivos XML no formato BPMN, exportados por ferramentas BPMS. Para tratar essas informações, optou-se por analisar o processo de modo a extrair uma tabela contendo todos os possíveis fluxos que podem ser executados no processo, para então permitir que sejam realizadas outras análises.

A partir da tabela, pode-se gerar elementos para teste, conforme a necessidade de cada domínio. Neste trabalho, os resultados da tabela de fluxos possíveis são aproveitados como entrada para gerar o código de cenários de teste, para uso com a ferramenta Cucumber aliada ao Selenium. Uma visão geral da abordagem criada é ilustrada na Figura 2, em que as etapas de análise do arquivo BPMN e de tratamento de resultados da tabela correspondem a funcionalidades das ferramentas implementadas, descritas nas seções a seguir.

Figura 2: Visão geral da abordagem proposta

4.1 Geração da Tabela de Fluxos

Para analisar um arquivo BPMN, foi elaborada uma estratégia que percorre sua representação em XML e manipula as informações necessárias. Enquanto o arquivo é percorrido, os elementos BPMN são analisados através da nomenclatura padrão especificada nas *tags* XML. Ao fim da execução, é gerado um arquivo em formato Excel, contendo uma tabela com todos os possíveis cenários/fluxos do processo. Na implementação da ferramenta de análise, foi utilizada a linguagem Java, que possui uma opção de API consolidada para a análise de documentos XML (JAXP).

O elemento BPMN mais importante para a análise é o *Sequence Flow*. Cada *Sequence Flow* possui um atributo `sourceRef`, que indica de onde este *Sequence Flow* vem, ou sua "fonte", e um atributo `targetRef`, que indica para onde ele vai, ou seja, seu alvo. Por conter esses atributos, os elementos deste tipo permitem percorrer todo o diagrama. Ao iniciar a execução da ferramenta de análise, é solicitado ao usuário o caminho para o arquivo BPMN e a identificação do processo a ser avaliado. Um diagrama pode conter mais de um processo e, nesse caso, o usuário pode escolher que sejam analisados todos ou somente alguns processos.

Na implementação da ferramenta, foi criada a classe *Node* que define um tipo de objeto que guarda informações básicas sobre as tarefas do processo e um *array* de objetos dessa classe. Durante a execução, esse *array* de objetos da classe Node é preenchido, formando assim um *array* de adjacências. O método principal percorre o processo recursivamente através dos elementos do tipo *Sequence Flow*. Enquanto houver tarefas encontradas no XML, um objeto da classe Node é criado e o método é chamado recursivamente de modo a retornar o *array* de adjacências para este objeto.

A partir do *array* de adjacências, são criados os fluxos possíveis pelos quais o processo pode passar. O método que cria os fluxos percorre recursivamente o *array* de adjacências criado anteriormente e "constrói" um novo fluxo possível, tendo como condição de parada o encontro de uma das últimas tarefas a serem executadas. Uma tarefa é uma das últimas quando o elemento que a sucede no fluxo for elemento do tipo *End Event*. Ao encontrar uma tarefa final, a informação é armazenada e é iniciada a construção de um novo fluxo possível.

Para criar a tabela de fluxos, cada tarefa existente no processo representará uma coluna na tabela e cada fluxo representará uma nova linha. Para cada fluxo, se a tarefa estiver presente a coluna será marcada com "X", caso contrário a coluna será marcada com um "-". Caso mais de um processo seja avaliado ao mesmo tempo, a tabela referente a cada processo estará separada individualmente no arquivo final.

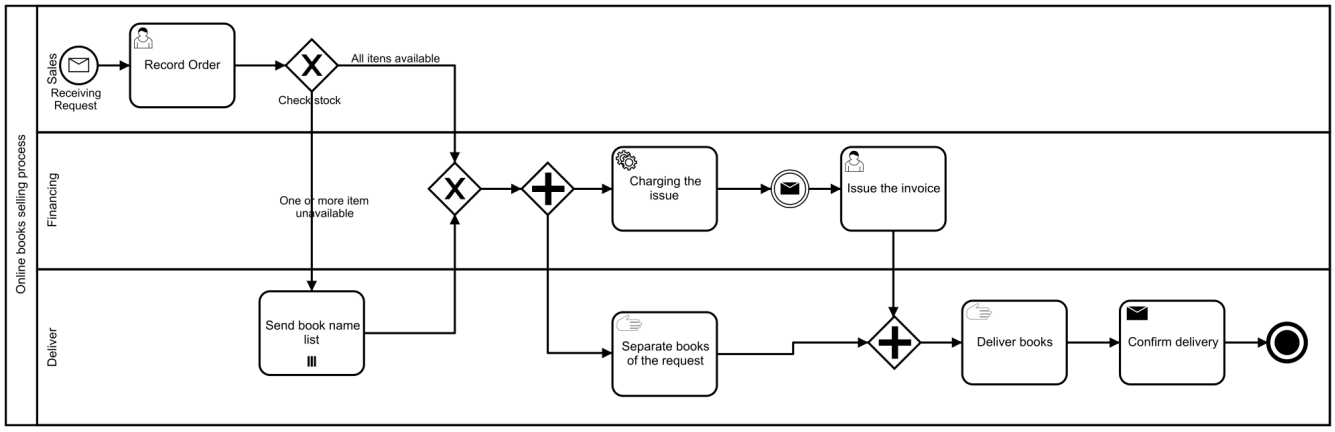
Na Figura 3, pode ser visto um resultado da execução da ferramenta de análise para o processo apresentado, que possui sete *Tasks* e, após e execução, resultou em quatro fluxos possíveis. Com a geração desta tabela, podem ser realizadas diversas análises sobre o processo: número de fluxos possíveis, identificar gargalos, dentre outras.

4.2 Geração de Código para os Testes Funcionais

A tabela de fluxos contém informações úteis para a criação de *scripts* de teste. A partir dela, pode-se automatizar a geração de elementos importantes para testes usando Selenium e Cucumber: o arquivo de *features*, contendo os cenários de teste, e o arquivo de *steps*.

O arquivo contendo os cenários de teste é essencial para o teste funcional com Cucumber, no entanto a criação dos arquivos de *features* contendo os cenários e do arquivo de *steps* pode ser trabalhosa, como já foi mencionado. Assim, de acordo com a visão geral da abordagem proposta (Figura 2), a tabela resultante da análise do arquivo BPMN pode ser processada para gerar os arquivos com códigos, que posteriormente deverão ser completados para se tornarem *scripts* de teste completos.

Para criar estes arquivos, a ferramenta de geração de código faz um tratamento dos resultados da tabela de fluxos.



Record Order	Send book name list	Charging the issue	Issue the invoice	Separate books of the request	Deliver books	Confirm delivery
X	X	X	X	-	X	X
X	X	-	-	X	X	X
X	-	X	X	-	X	X
X	-	-	-	X	X	X

Figura 3: Tabela de fluxos resultante para o processo *Books Selling Process* adaptado de https://www.edrawsoft.com/bpmn-diagram-examples.php.

Cada fluxo que foi obtido na análise, e que consta na tabela de fluxos, foi considerado um cenário diferente. Como se trata de testes funcionais, apenas tarefas que podem ser executadas por um usuário foram ser avaliadas (`userTask`). Assim, para cada fluxo é criado um novo cenário no arquivo de *features*, utilizando a notação do Cucumber, e contemplando apenas as tarefas que podem ser executadas por um usuário. O método correspondente a cada passo do cenário é criado ao mesmo tempo e inserido no arquivo de *steps*.

4.3 Dificuldades e Limitações da Análise

A maioria das dificuldades encontradas para a implementação da ferramenta para análise dos arquivos BPMN foram relacionadas à estrutura dos processos e como a implementação deve interpretar essas características. Para contornar estas dificuldades foram necessárias algumas escolhas relativas à representação dos processos, assim como foi preciso implementar algumas estratégias que estão descritas a seguir.

4.3.1 Gateways

Na criação dos fluxos possíveis, uma dificuldade ocorreu devido aos desvios causados por elementos do tipo *Gateway*. Na criação dos fluxos, elementos que partem de um desvio de fluxo precisaram ser tratados para criar os fluxos corretamente, por exemplo: se dois elementos, X e Y, vêm de um *Gateway* do tipo exclusivo, os fluxos obtidos até estes elementos serão duplicados, ou seja, metade dos fluxos passarão apenas por X e metade dos fluxos passarão apenas por Y.

No caso de *Gateways* do tipo paralelo e inclusivos foi necessário fazer uma escolha para representar estes fluxos na tabela. Em *Gateways* do tipo paralelo todos os fluxos serão executados "ao mesmo tempo", enquanto no tipo inclusivo um ou mais fluxos podem ser executados em paralelo.

Assim, decidiu-se por expressar cada fluxo que parte desses *Gateways* separadamente, como se fosse um *Gateway* do tipo exclusivo. Estimou-se que esta opção facilitaria na visualização dos fluxos para os testes já que, para a execução dos testes funcionais automatizados, é analisado cada fluxo como um "cenário" diferente. Também corroborou para esta opção que, geralmente, tarefas que partem de um *Gateway* e podem ser executadas ao mesmo tempo estão localizadas em *Lanes* diferentes, o que indica que usuários diferentes irão executar cada tarefa. Outros tipos de *Gateways* como o *Event Based Gateway* também são tratados da mesma forma que *Gateways* exclusivos.

Também foi necessário tratar os casos em que o *Gateway* representa apenas a união de fluxos. Para esses casos, a abordagem analisa a quantidade de fluxos ou *Sequence Flows* que chegam e partem deste elemento. Caso um *Gateway* tenha como entrada mais de um *Sequence Flow* e como saída apenas um, trata-se esse elemento como a representação da união de fluxos.

4.3.2 Outras dificuldades

Devido ao fato da estratégia implementada ser executada recursivamente e percorrer o processo baseado no fluxo dos elementos do tipo *Sequence Flow*, ocorreram problemas em processos onde uma parcela do processo também é executada recursivamente ou quando existe algum tipo de laço. Estes problemas foram causados devido ao fluxo nunca encontrar um "fim". Para solucionar esse problema, decidiu-se por parar a análise de um determinado *Sequence Flow* quando este tivesse como alvo uma tarefa que já havia sido executada durante a construção do caminho/fluxo atual. Essa solução mostrou-se efetiva pois evitou caminhos repetidos, bem como removeu a dependência de alterações manuais.

Algumas ferramentas exportam o diagrama para o formato BPMN inserindo um *namespace* antes no nome de cada *tag* XML. Por exemplo, a *tag* de nome `task` pode estar representada como "`semantic:task`" e isso pode impedir que o *parser* XML do Java identifique os elementos. Assim, foi necessário preparar métodos para tratar este tipo de situação, analisando o nome das *tags* BPMS juntamente com os respectivos *namespaces*.

Também optou-se por tratar elementos *Subprocess*, que representam um pequeno conjunto de tarefas, como uma única tarefa. Assim um *Subprocess* será representado como uma tarefa na tabela de fluxos possíveis.

5. AVALIAÇÃO DA ABORDAGEM

Para avaliar a abordagem proposta, foram buscados processos já existentes e de diferentes domínios. Procurou-se também processos que tivessem algumas diferenças de notação, para verificar como a solução implementada trataria essas diferenças. Visando utilizar um repositório de processos amplamente acessível, os processos escolhidos foram obtidos em diferentes *sites*: *site* da *Object Management Group*[1], *site* do BPMS Camunda[2], no repositório *Gen My Model*[3] e no site da empresa *Edraw Soft*[4]. Um resumo dos processos utilizados no teste pode ser visualizado na Tabela 1.

Ao todo, foram utilizados 10 processos de diferentes fontes para avaliação. Todos os processos foram analisados manualmente, gerando tabelas de fluxos que depois foram comparadas com aquelas geradas automaticamente. Em todos os casos, os resultados foram idênticos.

Neste artigo, apresenta-se um dos casos em maior detalhe. O processo deste caso é apresentado na Figura 4. Este processo possui algumas particularidades: dois inícios possíveis que são unidos por um *Gateway* exclusivo; existe um desvio, resultando em mais um fluxo, que é causado por *Event*; e duas divisões de fluxos criadas a partir de um *Event based gateway* que é tratado pela abordagem como um *Exclusive Gateway*.

Ao executar a ferramenta sobre o arquivo BPMN referente ao processo da Figura 4, foi obtida a Tabela 2 representando os fluxos possíveis dentro do processo. Na Tabela 2, cada coluna representa uma tarefa dentro do processo e cada linha representa um fluxo possível. Neste caso, temos oito fluxos possíveis dentro do processo, dentro destes fluxos também está representada a divisão de caminhos baseada em um evento: se o evento ocorrer a tarefa *acknowledge error* é executada; e se o evento não ocorrer a tarefa não é executada.

Levando em conta que foram obtidos oito caminhos possíveis, a geração da tabela pode auxiliar a reduzir o tempo de análise necessário para executar o teste funcional dos processos, possibilitando que os fluxos possíveis sejam vistos rapidamente. A tabela de fluxos também pode ser utilizada para identificar outros pontos importantes para o teste do processo como gargalos de execução. Por exemplo, pode se dizer que o teste de todas as tarefas que podem ser executadas por usuários (*acknowledge default start*, *acknowledge message 'start'*, *acknowledge error*, *acknowledge answer* e *acknowledge timeout*) é importante, pois todas estas tarefas podem ser executadas em um total de quatro fluxos diferentes. As tarefas *call some service* e *call some async service* são executadas em todos os fluxos exibidos na tabela, no entanto, estas são tarefas do tipo *script* e não são executadas diretamente por um usuário.

[1]Object Management Group (OMG). Acessível em http://www.omg.org/spec/BPMN/20100602/2010-06-03/.
[2]. Acessível em: https://camunda.org/examples/.
[3]GenMyModel. Acessível em https://repository.Gen My Model.com/online-example.
[4]Edraw Visualization Studios. Acessível em www.Edraw Soft.com/bpmn-diagram-examples.php

Tabela 1: Resumo dos processos utilizados no teste da aplicação.

Processo	*Tasks*	*Gateways*	Fluxos
Recruitment and selection - Candidate interviews	3	2 *Exclusive Gateways*	3
Counter example	5	3 *Exclusive Gateways*	3
Incident Management - Trouble Ticket System	6	2 *Exclusive Gateways*	3
Employment application - Employer	8	2 *Exclusive Gateways*	3
Good Example of a Symmetric Model 2	6	2 *Parallel Gateways* e 2 *Exclusive Gateways*	4
Books Selling Process	7	2 *Parallel Gateways* e 2 *Exclusive Gateways*	4
Hardware Retailer	8	2 *Parallel Gateways*, 2 *Exclusive Gateways* e 2 *Inclusive Gateways*	4
Recruitment and selection - Employer selection and recruitment	13	4 *Parallel Gateways* e 3 *Exclusive Gateways*	6
Call complaint	15	4 *Exclusive Gateways* e 2 *Parallel Gateways*	6
Camel + Camunda	7	1 *Event Based Gateway* e 1 *Exclusive Gateway*	8

Além da tabela com os fluxos possíveis, a abordagem também cria alguns elementos importantes para o teste funcional. Executando a ferramenta desenvolvida sobre o arquivo BPMN referente ao diagrama da Figura 4, obtém-se cenários de teste como os representados na Figura 5.

Ao todo foram gerados oito cenários possíveis para o processo em questão, ou seja, o mesmo número de fluxos obtidos. Isso ocorreu pois, analisando apenas as tarefas do tipo *User Task*, todos os fluxos gerados resultavam em um cenário diferente. A criação destes cenários manualmente poderia ser trabalhosa, principalmente devido à análise necessária para identificar as divisões de fluxos e as tarefas alvo de testes.

Para o teste com o Cucumber-JVM também é criado um arquivo contendo os métodos correspondentes à cada etapa dos cenários. Um trecho do arquivo `StepsDefinition` utilizado para o teste com o Cucumber, escrito em linguagem Java, contendo os métodos relativos a cada etapa dos cenários pode ser visualizado na Figura 6. Os métodos apresentados podem ser preenchidos posteriormente com o código exportado após a captura utilizando o Selenium.

Neste exemplo, foram gerados os cenários e códigos para teste referentes a todos os fluxos possíveis dentro do processo. Sendo assim, todos os fluxos obtidos na tabela podem ser testados. Por outro lado, também pode se utilizar a tabela gerada como apoio para selecionar tarefas e fluxos específicos que se deseja testar. Como é gerado o código de teste referente a todos os fluxos que contêm *Tasks* executadas por usuários, é possível executar o teste para todos os fluxos ou, por outro lado, pode-se utilizar a tabela gerada para verificar quais os fluxos mais importantes e utilizar os códigos gerados referentes apenas a esses fluxos.

No exemplo apresentado também havia apenas uma processo dentro do arquivo, mas arquivos BPMN podem conter

Figura 4: Processo *Camel + Camunda*, adaptado de *Camunda BPMN Workflow Engine*.

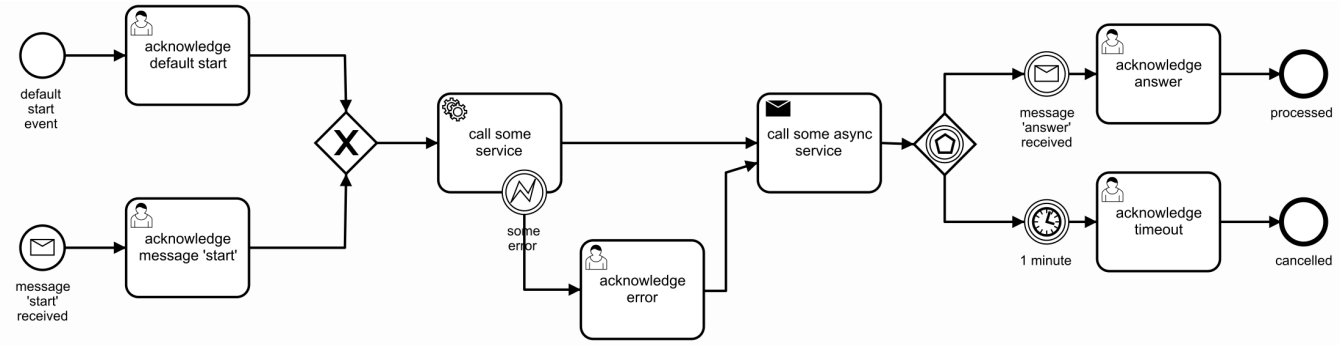


Tabela 2: Tabela de fluxos resultante do processo da Figura 4.

acknowledge default start	acknowledge message 'start'	call some service	acknowledge error	call some async service	acknowledge timeout	acknowledge answer
X	-	X	X	X	X	-
X	-	X	X	X	-	X
X	-	X	-	X	X	-
X	-	X	-	X	-	X
-	X	X	X	X	X	-
-	X	X	X	X	-	X
-	X	X	-	X	X	-
-	X	X	-	X	-	X

Figura 5: Exemplo de cenários de teste gerados para o processo *Camel + Camunda*.

```
Scenario: Process Engine 0
Given I am on task acknowledge default start
When
Then
When I am on task acknowledge error
Then
When I am on task acknowledge timeout
Then
Scenario: Process Engine 1
Given I am on task acknowledge default start
When
Then
When I am on task acknowledge error
Then
When I am on task acknowledge answer
Then
...
```

Figura 6: Exemplo de código gerado para o teste com o Cucumber-JVM utilizando os cenários da Figura 5.

```
@Given("^I am on the task acknowledge default start$")
public void method0() throws Exception {}
@when("^name$")
public void method1() throws Exception {}
@Then("^name$")
public void method2() throws Exception {}
@When("^I am on the task acknowledge error$")
public void method3() throws Exception {}
@Then("^name$")
public void method4() throws Exception {}
@When("^I am on the task acknowledge timeout$")
public void method5() throws Exception {}
...
```

vários processos. Caso seja necessário, é possível gerar os dados para todos os processos dentro dos arquivos BPMN ao mesmo tempo.

6. REFERÊNCIAS

[1] L. Apfelbaum and J. Doyle. Model based testing. In *Software Quality Week Conference*, pages 296–300, 1997.

[2] P. Baker, Z. R. Dai, J. Grabowski, I. Schieferdecker, and C. Williams. *Model-driven testing: Using the UML testing profile*. Springer Science & Business Media, 2007.

[3] A. S. Bitencourt, D. M. B. Paiva, and M. I. Cagnin. Elicitação de requisitos a partir de modelos de processos de negócio em bpmn: Uma revisão sistemática. In *XII Simpósio Brasileiro de Sistemas de Informação (SBSI)*, 2016.

[4] J. P. Bittencourt, E. Proencia, and R. S. P. Maciel. Modelando serviços a partir de processos de negócio: Uma abordagem dirigida a modelos. In *XII Simpósio Brasileiro de Sistemas de Informação (SBSI)*, 2016.

[5] K. Böhmer and S. Rinderle-Ma. A genetic algorithm for automatic business process test case selection. In *On the Move to Meaningful Internet Systems: OTM 2015 Conferences*, pages 166–184. Springer, 2015.

[6] S. A. Correia and A. R. Silva. Técnicas para construção de testes funcionais automáticos. In *QUATIC*, pages 111–117, 2004.

[7] Z. R. Dai. Model-driven testing with uml 2.0. *Computer Science at Kent*, page 179, 2004.

[8] J. L. de Moura and A. Charão. Automação de testes em aplicações de BPMS: um relato de experiência. In *XIV Simpósio Brasileiro de Qualidade de Software*,

pages 212–219, 2015.

[9] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos. A survey on model-based testing approaches: a systematic review. In *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, pages 31–36. ACM, 2007.

[10] E. Dustin, J. Rashka, and J. Paul. *Automated software testing: introduction, management, and performance*. Addison-Wesley Professional, 1999.

[11] Forrester Research. The forrester wave: BPM suites, Q1 2013, 2013.

[12] V. Garousic, A. Mesbahb, A. Betin-Canc, and S. Mirshokraie. A systematic mapping study of web application testing. *Information and Software Technology*, 55(8):1374–1396, Aug. 2013.

[13] A. Z. Javed, P. A. Strooper, and G. Watson. Automated generation of test cases using model-driven architecture. In *Automation of Software Test, 2007. AST'07. Second International Workshop on*, pages 3–3. IEEE, 2007.

[14] C. Kaner, J. Bach, and B. Pettichord. *Lessons learned in software testing*. John Wiley & Sons, 2008.

[15] J. Kasurinen, O. Taipale, and K. Smolander. Software test automation in practice: empirical observations. *Advances in Software Engineering*, 2010, 2010.

[16] M. Kurz. Bpmn model interchange: The quest for interoperability. In *Proceedings of the 8th International Conference on Subject-oriented Business Process Management*, S-BPM '16, pages 6:1–6:10, New York, NY, USA, 2016. ACM.

[17] Z. J. Li, W. Sun, and B. Du. Bpel4ws unit testing: Framework and implementation. *International Journal of Business Process Integration and Management*, 3(2):131–143, 2008.

[18] H. Liu, Z. Li, J. Zhu, and H. Tan. Business process regression testing. In *International Conference on Service-Oriented Computing*, pages 157–168. Springer, 2007.

[19] OMG. Business process model and notation (BPMN), 2011. Disponível em: http://www.omg.org/spec/BPMN/2.0.

[20] V. S. F. Pinheiro, N. M. C. Valentim, and A. M. R. Vincenzi. Um comparativo na execução de testes manuais e testes de aceitação automatizados em uma aplicação web. In *XIV Simpósio Brasileiro de Qualidade de Software*, pages 260–267, 2015.

[21] A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch, and T. Stauner. One evaluation of model-based testing and its automation. In *Proceedings of the 27th international conference on Software engineering*, pages 392–401. ACM, 2005.

[22] D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In *Proceedings of the 7th International Workshop on Automation of Software Test*, AST '12, pages 36–42, Piscataway, NJ, USA, 2012. IEEE Press.

[23] L. P. Ramos and A. Bessa. Uma abordagem de gestão e desenvolvimento de automatização de processos de negógio com apoio de BPMS. In *XIV Simpósio Brasileiro de Qualidade de Software*, pages 137–151, 2015.

[24] F. Ricca and P. Tonella. Analysis and testing of web applications. In *Proceedings of the 23rd international conference on Software engineering*, pages 25–34. IEEE Computer Society, 2001.

[25] S. Shenoy, N. A. A. Bakar, and R. Swamy. An adaptive framework for web services testing automation using jmeter. In *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*, pages 314–318. IEEE, 2014.

[26] H. Smith and P. Fingar. *Business Process Management: The Third Wave*. Meghan-Kiffer Press, 2003.

[27] A. Stefanescu, S. Wieczorek, and A. Kirshin. Mbt4chor: A model-based testing approach for service choreographies. In *European Conference on Model Driven Architecture-Foundations and Applications*, pages 313–324. Springer, 2009.

[28] W. Van der Aalst. Business process management: A comprehensive survey. *ISRN Software Engineering*, 2013(507984), 2013.

[29] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2nd edition, 2012.

[30] K. Wiklund, D. Sundmark, S. Eldh, and K. Lundvist. Impediments for automated testing – an empirical analysis of a user support discussion board. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*, pages 113–122, March 2014.

[31] Winter Green Research. Business process management (BPM) cloud, mobile, and patterns: Market shares, strategies, and forecasts, worldwide, 2013 to 2019, 2013.