Gerando códigos para melhoria do Teste Funcional Automatizado de ferramentas de Gerenciamento de Processos de Negócio

Jéssica Lasch de Moura[1], **Andrea Schwertner Charão**[1]

[1]Centro de Tecnologia – Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brazil

[2]Laboratório de Sistemas de Computação (LSC) – Universidade Federal de Santa Maria

{jmoura,andrea}@inf.ufsm.br

Abstract.

Resumo.

1. Introdução

-Importância dos testes -Trabalho anterior: teste de carga + teste funcional; -Teste funcional com selenium e cucumber mais promissor, mas pode ser trabalhoso; -Também é necessário criar manualmente cada cenário de teste -Alternativa pra auxiliar na criação dos testes: analisar o diagrama no formato BPMN e criar alguns códigos; -Objetivo: melhorar e facilitar o teste funcional automatizado;

2. BPM e Teste

-Teste é pouco abordado -Teste é importante -Teste funcional ajuda a aumentar a qualidade do software; -Trabalhos relacionados;

3. Business Process Model and Notation - *BPMN*

O padrão Business Process Model and Notation, ou BPMN, foi criado com o objetivo de fornecer uma notação fácilmente 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[Model 2011]. 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[Kurz et al.], isto permite que o XML seja analisado para obter-se informações importantes sobre os processos. A maioria das ferramentas BPM disponibiliza a exportação do processo em formato XML seguindo o padrão BPMN. No padrão BPMN um processo é descrito como um diagrama de elementos de fluxo, que são: Tasks (Tarefas), Events (Eventos), Gateways e Sequence Flows [Kurz et al.].

Uma *Activiti* é a menor parte de um processo, e é utilizada quando o processo não pode ser dividido mais detalhadamente. Quando uma *Activiti* está inserida no contexto de um processo, ela é chamada de *Task*. Geralmente as Tasks são executadas por um usuário final ou por uma aplicação. Existem diferentes tipos de Tasks para representar o diferente

comportamente que tava tarefa pode representar. Por exemplo, uma *userTask* representa uma tarefa em que um usuário deve executar uma ação.

Um *Event* é algo que ”acontece” durante o curso de um processo[Model 2011]. Existem três tipos principais de eventos: Start Event (indica o início do processo), End Event (indica um fim do processo) e Intermediate Events (indica um evento entre o início e o fim do processo).

Gateways são usados para controlar o fluxo do processo. Em gateways do tipo *Exclusive*, apenas um dos caminhos que partem do gateway poderão ser seguidos, já em gateways do tipo *Inclusive* um ou mais caminhos podem ser seguidos. Em gateways do tipo *Parallel* todos os caminhos são tomados em paralelo.

Um elemento *Sequence Flow* é usado para exibir a ordem em que os demais elementos são executados em um processo. Cada *sequenceFlow* possui um atributo *sourceRef* que indica de onde este *sequenceFlow* vem, ou sua ”fonte”, e um atributo *targetRef* que indica para onde ele vai, ou seja, seu alvo. Cada *sequenceFlow* possui apenas possui apenas uma fonte e um alvo. Analisando os elementos *sequenceFlow* e analisando os dois atributos citados acima é possível identificar todo o fluxo do processo.

4. Teste Funcional

-O que é... -Cobertura de teste funcional

4.1. Teste com selenium

-funcionamento, elementos necessários pro teste, código

4.2. Teste com cucumber

-funcionamento, elementos necessários pro teste, cenário, stepDefinition

5. BPMN Parser

-Linguagem, ferramenta,exemplos de diagramas foram obtidos em... -O que o parser faz (resumo): tabelinha + criação dos códigos arquivo bpmn é exportado usa java + parser java para xml The Object Management Group (OMG) has developed a standard Business Process Model and Notation (BPMN). The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

5.1. Funcionamento

O principal elemento para o funcionamento do parser é o *Sequence Flow*. Por conter os atributos targetRef e sourceRef os elementos deste tipo permitem percorrer todo o diagrama. Ao iniciar o parser, é solicitado ao usuário o caminho para o arquivo BPMN e a ID do processo a ser avaliado. Um diagrama pode conter mais de um processo e, nesse caso, o usuário pode escolher que todos processos sejam avaliados.

Table 1. Exemplo de tabela resultante

Quotation Handling	Approve Order	Order Handling	Shipping Handling	Review Order
X	X	0	0	0
X	X	X	X	X

A classe *Node* define um tipo de objeto que guarda informações básicas sobre as tarefas do processo e um *array* de objetos da mesma classe. Durante a execução do parser um array de objetos da classe Node é preenchido formando assim um array de adjacências. O método principal do parser percorre o processo recursivamente através dos elementos do tipo *Sequence Flow*, enquanto um tarefa for encontrada, 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 caminhos possíveis pelos quais o processo pode passar. O método que cria os caminhos percorre recursivamente o array de adjacências criado anteriormente e ”constrói” um novo caminho, 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 *End Event*. Ao encontrar uma tarefa final, o caminho é armazenado e é iniciada a construção de um novo caminho.

Os caminhos obtidos são base para a criação dos dois artefatos para o teste funcional: tabela de caminhos possíveis e código para o teste funcional. Para criar a tabela com os caminhos possíveis, cada tarefa existente no processo representará uma coluna na tabela e cada caminho representará uma nova linha. Para cada caminho, se a tarefa estiver presente a coluna será marcada com ”X” caso contrário a coluna será marcada com um ”0”. Caso mais de um processo seja avaliado ao mesmo tempo, a tabela referente à cada processo estará separada individualmente no arquivo final. Na Tabela 1, pode ser visto uma tabela resultante da execução do parser para um processo com cinco *tasks* e dois caminhos possíveis.

Para a criação do código para o teste funcional dois arquivos devem ser criados: o arquivo contendo os cenários e a classe *stepDefinition* contendo os métodos para cada passo do cenário. Para criar estes artefatos, cada caminho obtido é considerado um cenário diferente. Como trata-se de um teste funcional, apenas tarefas que podem ser executadas por um usuário devem ser avaliadas (*userTask*,*manualTask*...). Assim, para cada caminho é criado um novo cenário utilizando a notação do Cucumber e comtemplando apenas as tarefas que podem ser executadas por um usuário. O método correspondente a cada passo do cenário é criado ao mesmo tempo. Na Figura 1, podem ser vistos dois cenários resultante da execução do parser para um processo com dois caminhos possíveis.

5.2. Dificuldades/Limitações

Algumas ferramentas exportam o diagrama para o formato BPMN inserindo um “tipo” 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 do Java identifique os elementos. Assim, foi necessário preparar o parser para tratar este tipo de situação.

Na criação dos artefatos para o teste funcional são levados em conta apenas tarefas que podem ser executadas por um usuário. Apesar de utilizarem o mesmo padrão BPMN,

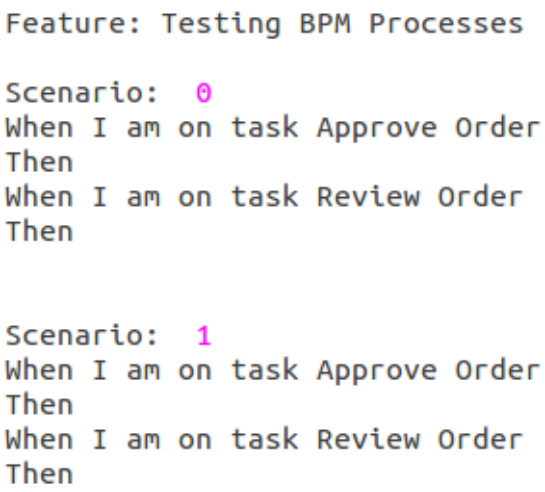

Figure 1. Exemplo de cenários resultantes

algumas ferramentas podem utilizar nomes diferentes parat representar estas tarefas no arquivo XML. Por isso, dependendo da tarefa que for utilizada, pode ser necessário substituir o nome dos tipos a serem utilizados no parser.

Na criação dos caminhos, uma dificuldade ocorreu devido aos desvios causados por elementos do tipo *Gateway*. Para isso, para cada Node criado no array é guardada o tipo do elemento que o antecede. Assim, na criação dos caminhos, elementos que partem de um desvio de fluxo precisaram ser tratados para criar os caminhos corretamente, por exemplo: se dois elementos, X e Y, vem de um gateway exclusivo, os caminhos obtidos até estes elementos serão duplicados, ou seja, metade dos caminhos passaram apenas por X e metade dos caminhos passarão apenas por Y.

Devido ao fato de o parser ser executado recursivamente e percorrer o processo baseado no fluxo dos elementos do tipo *sequenceFlow*, ocorreram problemas em processos onde uma parcela do processo também é executada recursivamente. Estes problemas foram causados devido ao fluxo nunca encontrar um ”fim”. Para solucionar esses problema foi criado um ”delimitador de recursão” que define e controla o número de vezes em que o mesmo *sequenceFlow* será executado.

6. Resultados

7. References

References

Kurz, M., Menge, F., and Misiak, Z. Diagram interchangeability in bpmn 2.

Model, B. P. (2011). Notation (bpmn) version 2.0. *OMG Specification, Object Management Group*.

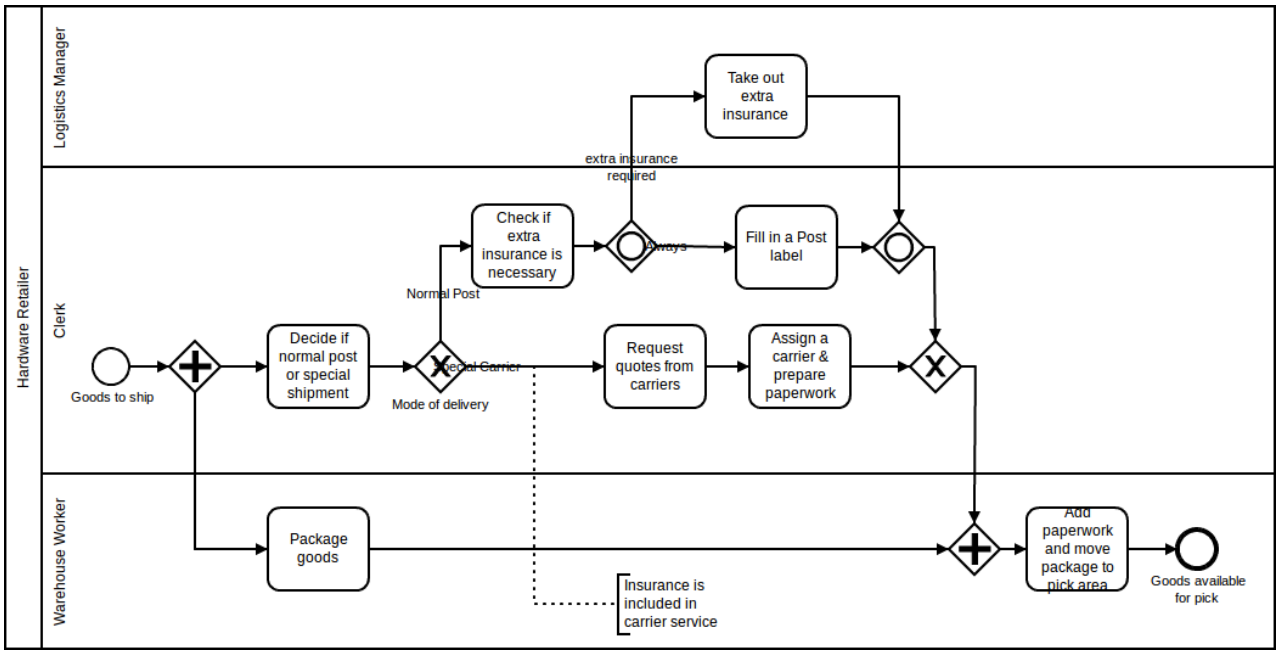

Figure 2. Exemplo de Processo. Fonte: Object Management Group