Automação de Testes em Aplicações de BPM: um Relato de Experiência

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

[1]Núcleo de Ciência da Computação
Universidade Federal de Santa Maria – UFSM

{jmoura, andrea}@inf.ufsm.br

Resumo. *Este artigo aborda questões relacionadas ao teste automatizado de aplicações desenvolvidas com o apoio de sistemas de gestão de processos de negócio (Business Process Management Systems – BPMS). Para isso, apresenta-se um relato de experiência de automação de testes de carga e de testes funcionais em uma aplicação de BPM, utilizando-se as ferramentas de código-aberto Apache JMeter e Selenium. Os resultados evidenciam limitações e oportunidades na automação de testes de aplicações de BPM.*

Abstract. *This article discusses the automated testing of applications developed with the support of Business Process Management Systems – BPMS. We present an experience report on the automation of load tests and functional tests over a real BPM application, using the Apache JMeter and Selenium open-source tools. Results show the limitations and opportunities in test automation of BPM applications.*

1. Introdução

Atualmente, a gestão de processos de negócio (*Business Process Management* – BPM) tem suscitado o interesse de empresas 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 modelagem, administração, configuração e análise de processos de negócio [Weske 2012, van der Aalst 2013]. Associados a isso, surgiram os sistemas BPM (*Business Process Management Systems* ou *Suites* – BPMS), que são ferramentas de software para apoio ao ciclo de vida da gestão de processos de negócio. Tais ferramentas, quando bem aplicadas, têm o potencial de alavancar aumentos de produtividade e redução de custos nos mais variados tipos de organizações.

Dentre os diversos BPMS disponíveis atualmente, é comum encontrar ferramentas com suporte a modelagem, configuração e execução de processos de negócio. Por outro lado, tarefas relacionadas à simulação, monitoramento, verificação e testes ainda são consideradas um desafio nesta área [van der Aalst 2013]. Em particular, o teste automatizado de aplicações de BPM é pouco abordado, tanto pela comunidade da área de BPM [van der Aalst 2013] como da área de testes de software [Graham e Fewster 2012]. Diante disso, estima-se que muitas organizações se limitem a testes manuais em suas aplicações de BPM. No entanto, a falta de automação nos testes pode levar a vários problemas durante a implementação e execução de processos de negócio, como baixa aderência aos requisitos, maior esforço dos desenvolvedores, desperdício de tempo e aumento do risco de duplicação de esforços e de erro humano.

Nesse contexto, o presente artigo relata uma experiência de automação de testes em uma aplicação de BPM desenvolvida para agilizar um processo em uma instituição pública de ensino. Todas as etapas do desenvolvimento da aplicação, utilizando o BPMS Bonita Open Solution (BOS), foram apresentadas em um trabalho precedente [de Moura et al. 2013].

2. BPM e Testes

Em muitos casos, o termo BPM pode ser usado com significados diferentes [Ko 2009], às vezes com mais ênfase em tecnologia (software) e outras vezes mais associado a gestão. Mesmo assim, a área tem convergido sobre o ciclo de vida de aplicações de BPM, que envolve as atividades de análise, modelagem, execução, monitoramento e otimização [ABPMP 2012].

Os sistemas de BPM (BPMS) têm se afirmado como ferramentas essenciais para suporte a atividades desse ciclo de vida. Atualmente, pode-se dizer que um típico BPMS oferece recursos para definição e modelagem gráfica de processos, controle da execução e monitoramento de atividades dos processos. Alguns exemplos de BPMS que se destacam neste cenário são IBM Websphere, Oracle BPM Suite, Intalio, Bizagi, TIBCO BPM, Activiti e Bonita Open Solution.

Nota-se que a preocupação com testes não fica evidente na ferramentas BPMS. De fato, analisando-se o material promocional e a documentação publicamente disponível sobre os principais BPMS, observa-se uma ênfase em etapas de modelagem e execução. Visivelmente, tais recursos são um diferencial no desenvolvimento de aplicações de BPM, em comparação ao desenvolvimento de software em geral. No entanto, aplicações de BPM também estão sujeitas a defeitos e, por isso, podem se beneficiar de avanços na área de testes de software.

Em testes de software, há muitas tarefas que podem ser trabalhosas e propensas a erros quando realizadas manualmente. Por este fato, vários autores relatam a importância dos testes automatizados em ambientes de desenvolvimento [Chiavegatto et al. 2013]. Com a evolução das áreas de qualidade e teste de software, foi surgindo uma variedade de soluções para automação de testes. Ferramentas para testes unitários, por exemplo, são numerosas e costumam se integrar aos ambientes de desenvolvimento [Kolawa e Huizinga 2007]. Outro exemplo é o teste de aplicações baseadas em interfaces gráficas ou Web [Nguyen et al. 2003].

Assumindo que aplicações de BPM podem ser tratadas como software em geral, é possível testá-las sob diferentes aspectos, por meio de tipos de testes já consagrados em engenharia de software, como por exemplo testes funcionais do tipo caixa-preta ou teste de carga. Sob esta ótica, pode-se empregar ferramentas de automação de testes alinhadas com cada tipo de teste. No entanto, a adoção esta abordagem pode ter limitações e dificuldades, pois não leva explicitamente em conta o ciclo de vida de aplicações de BPM.

3. Processo alvo dos testes e ?

Os testes realizados neste trabalho referem-se a um processo realizado com frequência em instituições de ensino superior, que é a apreciação de Atividades Complementares

de Graduação (ACGs), ou seja, atividades que formam a parte flexível do currículo dos graduandos (participação em palestras, eventos, projetos, etc.). Em um trabalho anterior [de Moura et al. 2013], esse processo foi modelado e implantado utilizando a ferramenta Bonita Open Solution. Como mostra a Figura 1. É um processo complexo que permite analisar diversas funcionalidades dos BPMS e, justamente por já ter sido alvo de um trabalho, é um processo no qual já se tem uma grande experiência.

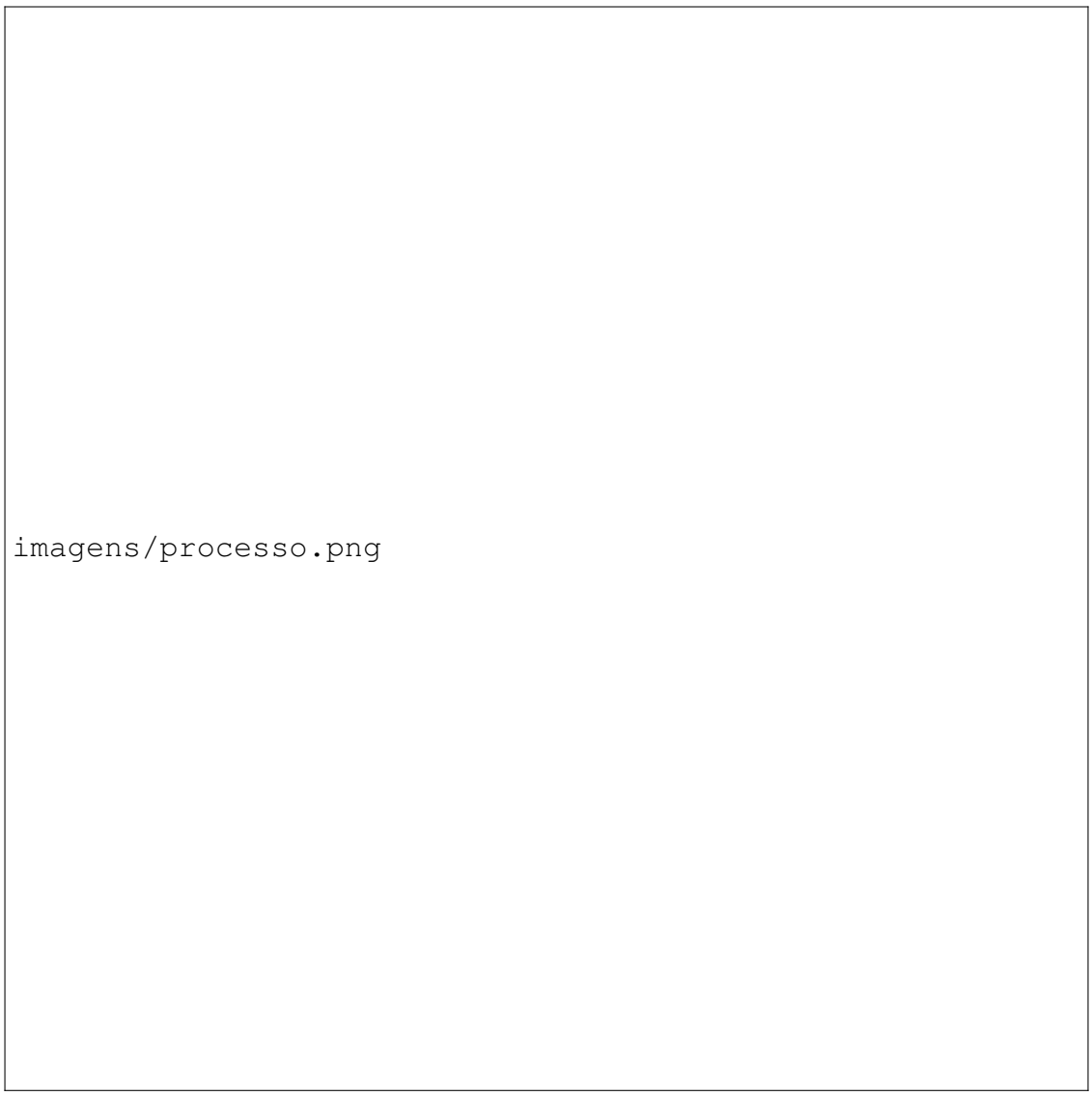

Figura 1. Diagrama da modelagem do processo

Devido ao tempo limitado para a conclusão deste trabalho, decidiu-se por escolher duas ferramentas para o estudo. Primeiramente, foi escolhida a ferramenta Bonita Open Solution, devido a esta já ter sido usada em um trabalho anterior [de Moura et al. 2013] e, por isso, tem se uma vasta experiência nesta ferramenta.

O Bonita Open Solution (BOS) é uma ferramenta distribuída sob uma licença de software livre, desenvolvida em Java, pela empresa BonitaSoft [BonitaSoft 2012]. A ferramenta BOS oferece componentes tanto para a modelagem como para a implementação e transformação de processos. A modelagem e customização de processos é realizada

Aluno Solicita ACG	Anexar parecer tutor	Tutor avalia	Avisa aluno (indeferido)	Relator avalia	Colegiado avalia	Atualiza BD	Secretaria avalia	Coordenador avalia	Reunião presencial	Avisa aluno (indeferido)	Avisa aluno (aprovado)
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	-	-	-	-	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	-	-	-	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	-

Tabela 1. Caminhos possíveis

através do Bonita Studio, um componente com interface gráfica tipo desktop que agrupa ferramentas de desenvolvimento. Também é possível agregar funcionalidades aos processos através de conectores que podem, inclusive, receber personalizações feitas através de códigos.

Analisando as ferramentas disponíveis e levando em conta os trabalhos e livros publicados [**?**] que abordam o uso da ferramenta, o segundo software escolhido foi o Activiti. O Activiti [**?**] é um BPMS de código aberto, distribuído sob a uma licença Apache, criado em Java e usa BPMN 2.0 para a modelagem dos processos, ele pode ser executado em qualquer plataforma, servidor, cluster ou na nuvem. A ferramenta Activiti oferece componentes distintos para modelagem, implementação e execução de processos. A modelagem e customização dos processos é feita no Activiti Designer que é um plugin para a plataforma Eclipse, o que torna o ambiente de criação fácil de ser usado, a execução é feita no componente chamado Activiti Explorer.

Na Tabela 1 são exibidos todos os possíveis caminhos pelos quais o processo alvo dos testes poderia passar, sendo 'x' a representação de que o caminho passou pela determinada etapa do processo. Esta tabela foi criada durante a implantação da aplicação citada neste trabalho para auxiliar na execução dos testes (que foram executados manualmente quando a aplicação foi implantada).

4. Descrição e Execução dos Testes

No planejamento dos testes automatizados, priorizou-se o teste de etapas que de fato revelaram problemas durante a operação. Com isso, buscou-se verificar se os problemas poderiam ser facilmente identificados antes de colocar-se a aplicação em produção. Os testes escolhidos foram: (a) testes de carga, que são um tipo de teste de desempenho, visando avaliar o comportamento do sistema frente a um grande número de solicitações e (b) testes funcionais, a fim de verificar as saídas do sistema produzidas a partir de entradas

	Teste de Carga	Teste Funcional
Motivo da escolha	• Durante a implantação do processo testado foi a falha do sistema perante um grande acesso de usuários em um dia específico.	• Durante a utilização do processo dentro da instituição, ocorreram alguns erros como: campo sem suporte a caracteres especiais, erro devido a nome de arquivos muito grandes, entre outros.
BPMS Bonita Open Solution (BOS)	• Houveram problemas com a execução de requisições que utilizavam a tecnologia Google Web Toolkit (GWT), foi resolvido atráves de estudo das requisições e utilização de ferramentas auxiliares para a captura [**?**]; • Em sistemas BPM as tarefas de um processo são interligadas e/ou dependentes entre si, no BOS existe uma chave que identifica cada execução do processo como única, e é criada no momento em que o usuário inicia o processo. Essa particularidade causou no problemas na execução dos testes e, para contorna-los, foi necessário estudar a fundo as requisições e o processo e utilizar opções da ferramenta JMeter bem como scripts para adaptar as requisições.	• A captura da execução funciona bem para a estrutura das páginas web da aplicação; • Ao executar o teste, alguns erros ocorrem devido a ferramenta Selenium "Buscar" elementos que ainda não foram carregados na página. Isto pode ser resolvido a partir da inserção de pequenos scripts nos métodos.
BPMS Activiti	Ocorreram os mesmos erros relativos à dependência entre as tarefas do processo, as chaves identificadoras foram encontradas, no entanto, foi impossível identificar em que requisição as mesmas eram geradas, não havia uma requisição cujo o retorno (resposta do servidor) contivesse as chaves utilizadas. Esta situação leva a crer que a geração das chaves identificadoras é feita internamente pelo BPMS, ou seja, não em uma requisição HTTP e, por sequência, esta não pode ser capturada e importada no Jmeter.	Alguns elementos das páginas que compõem a aplicação não são capturados pelo Selenium, acredita-se que esse problema ocorra devido a estrutura da páginas web que pode conter elementos com os quais o Selenium não trabalhe bem como divs,frames e scrips, por exemplo. Para contornar esse problema, foi necessário estudar a estrutura das páginas que web, localizar os elementos faltantes e então adicionar o código para acessá-los nos respectivos métodos.

Tabela 2. Comportamento de cada tipo de teste em cada ferramenta

pré-definidas. **Nenhum destes tipos de teste possui suporte no BPMS utilizado** (Bonita Open Solution), que inclui somente funcionalidades limitadas de simulação e depuração de execução dos processos. Assim, realizou-se um levantamento de ferramentas de teste disponíveis e selecionou-se as mais promissoras, antes de partir-se para o detalhamento e execução dos testes.

4.1. Teste de Carga

Para executar os testes de carga foi selecionada a ferramenta JMeter [**?**], e escolha desta ferramenta para testar aplicações BPM deu-se por diversos motivos.

Existem diversas ferramentas para automação de testes, especificamente se tratando de testes de carga e desempenho existem duas ferramentas próximas ao JMeter que são o The Grinder [**?**] e o WebLOAD [**?**], ambas são ferramentas open source e tem diversas funcionalidades. No entanto, a maior desvantagem dessas ferramentas é que não existe a possibilidade de capturar as requisições que devemos testar de uma forma automática, todas requisições precisam ser inseridas manualmente no plano de teste. No JMeter existe a opção 'Servidor de Proxy' que permite capturar o tráfico de requisições e este é automaticamente transformado em requisições no plano de teste.

O processo para realizar o teste, em ambos os BPMS, consiste de quatro etapas: Capturar as requisições HTTP, Exportar requisições (Formato .jrxml para JMeter), Configurar o script de plano de teste e, por fim, Executar o teste no JMeter. Na Tabela 2 são exibidas as principais observações sobre o comportamento dos testes em cada BPMS.

Ocorrem dois principais problemas na execução do teste de carga em ambas as ferramentas: (a) Captura e interpretação das requisições que utilizam a tecnologia GWT e (b) dependência entre as tarefas do processo cuja implementação muda conforme o BPMS.

No caso deste segundo problema, no BOS existe uma chave identificadora de sessão que é gerada no momento em que usuário acessa o sistema e outra chave identificadora de instância, ou seja, identifica cada execução do processo como única, e é criada no momento em que o usuário inicia o processo. Para executar o teste com sucesso, foi necessário localizar a requisição em que essas chaves são geradas, utilizar a ferramenta "Extrator de Expressão Regular"do JMeter para obter o valor da chave. Já na ferramenta Acitivi foi impossível identificar em que requisição as mesmas eram geradas, pois não havia uma requisição cujo o retorno (resposta do servidor) contivesse as chaves utilizadas. Esta situação leva a crer que a geração das chaves identificadoras é feita internamente pelo BPMS, ou seja, não em uma requisição HTTP e, por sequência, esta não pode ser capturada e importada no Jmeter.

4.1.1. Resultados

Com o objetivo de obter o comportamento do sistema em diferentes níveis de carga, foram executados testes com 1, 50, 100 e 200 usuários virtuais e foram analisadas as requisições referentes aos seguintes passos: efetuar login, exibir a página inicial do BOS, selecionar o processo, exibir o formulário inicial e enviar o formulário preenchido. Os testes foram executados em um servidor com 24 GB de RAM e 2 processadores Intel Xeon E5520, com 4 núcleos. Foi possível executar o teste de carga apenas na ferramenta Bonita Open Solution, pois o software Activiti demonstrou os problemas relatados acima.

Reproduzindo o teste com um usuário virtual, nas requisições correspondentes a login, exibição da página inicial, seleção do processo, exibição do formulário inicial e envio do formulário preenchido, a média do tempo de resposta foi de, respectivamente, 126, 32, 38, 80 e 73 milissegundos. A média do tempo de resposta total foi de 75 milissegundos e o desvio padrão foi igual a 21. Aumentando o número de usuários virtuais para cinquenta os valores do tempo de resposta subiram para 597, 191, 179, 368 e 152 milissegundos, a média foi igual a 329 milissegundos e o desvio padrão 401. Com 100 usuários o aumento do tempo de resposta fica ainda mais visível, sendo 1972, 571, 552, 760 e 694 milissegundos, com média 774 milissegundos e desvio padrão de 1449.

A saída da execução do teste com 200 usuários pode ser visualizada na Figura 2. Simulando 200 usuários virtuais, o tempo de resposta médio na requisição que executa o login no sistema foi de 10.149 milissegundos, ou seja, aproximadamente 10 segundos, o que pode ser considerado um tempo de resposta alto. Os tempos de resposta para exibição da página inicial, início do processo de solicitação, exibição do formulário e envio do formulário foram de, respectivamente, 3.239, 934, 2.122 e 1.918 milissegundos, a média de tempo de resposta para todas requisições foi de 3.111 milissegundos (ou seja, 3 segundos), e o desvio padrão foi de 13.088. Além dos altos tempos de resposta, o teste com 200 usuários virtuais apresentou taxas de erro, em algumas requisições, que não foram encontradas com um número menor de usuários. Por exemplo, a requisição que executou login apresentou uma taxa de 2% de erro e, ao todo, todas requisições obtiveram uma taxa de 7.82%, o que é preocupante.

Rótulo	# Amostras	Média	Mediana	Linha de 90%	Mín.	Máx.	% de Erro
Login	200	10149	1873	23551	139	128259	2,00%
Página Inicial	200	3239	433	3657	50	127286	1,00%
http://bpm-si.inf...	600	708	176	1242	23	21646	0,00%
http://bpm-si.inf...	200	2903	678	3080	93	64949	2,00%
http://bpm-si.inf...	2000	1421	642	2793	82	127194	0,20%

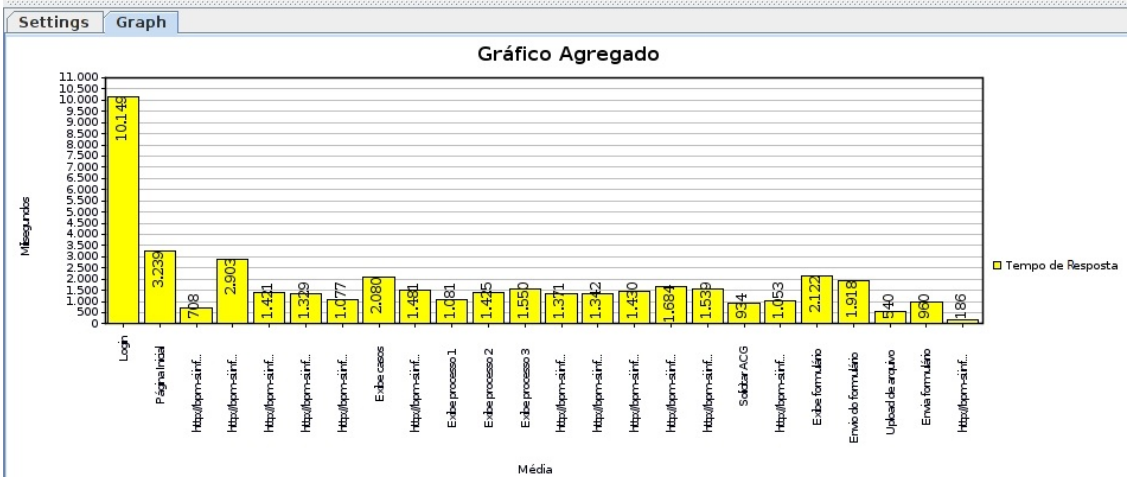

Figura 2. Resultado dos testes de carga com 200 usuários no JMeter.

De forma geral, portanto, este teste atingiu seus objetivos e serve para explicar um problema que ocorreu no sistema em execução que foi a falha do sistema, onde alguns arquivos foram corrompidos, bem como um tempo de espera muito alto para o aluno acessar o formulário de solicitação. Outro ponto importante é que este teste foi executado apenas no login e na primeira tarefa do processo alvo e, mesmo assim, a tarefa de teste já foi trabalhosa pelo fato de exigir uma análise profunda das requisições para executar os testes com sucesso. A captura destas etapas, utilizando o BlazeMeter, resultou em uma média de 100 requisições, considerando que deseja-se fazer o teste de outra etapa/gargalo que esteja localizada mais ao meio do processo a captura desta pode gerar muitas requisições o que tornaria difícil a análise de todas requisições para substituir as chaves geradas e, assim, tornaria inviável essa abordagem de teste.

Um fator importante também é que, pelo fato das requisições terem de ser analisadas para ser possível executar o teste e também ser necessário localizar a forma como a ferramenta BPMS implementa a dependência entre as etapas do processo, essa abordagem de teste pode tornar-se muito dependente do BPMS. Esta ideia é fortalecida pela impossibilidade de executar o teste na ferramenta Activiti afinal, dependendo da ferramenta, pode ser muito trabalhoso executar o teste, pode não ser possível executar o teste ou ainda pode ser possível e vantajoso, como ocorreu com o Bonita Open Solution.

4.2. Teste Funcional

Para executar os testes funcionais, escolheu-se a ferramenta Selenium [**?**]. Esta ferramenta possui basicamente duas partes complementares: Selenium IDE e Selenium WebDriver. A primeira é um plugin para o navegador Firefox, capaz de registrar e reproduzir interações do usuário com o navegador, assim permitindo criar scripts de teste rapidamente, sem escrita de código.

Durante a revisão bibliográfica, encontrou-se um trabalho apresentado no Simpósio Brasileiro de Qualidade de Software de 2013 [Chiavegatto et al. 2013] que uti-

lizava o Selenium WebDriver aliado ao Cucumber-JVM [**?**] para testar uma aplicação web, a partir desse trabalho decidiu-se utilizar a última na execução dos testes funcionais. O Cucumber é uma ferramenta que executa descrições de teste, em texto simples, como testes automatizados.

O processo para a execução do teste funcional é composto por seis etapas: Captura da execução no navegador (Selenium), Exportar código gerado (Selenium), Criar cenário de teste (Cucumber), Criar as definições dos passos do teste (Cucumber), Criar os métodos para cada passo (Java) e Executar o teste (Selenium).

Algumas personalizações são necessárias para a execução do teste funcional, a maioria delas ocorre nas etapas de exportação/execução do teste. Em ambos os BPMS, ao executar o teste gravado, ocorrem erros pois o Selenium não encontra os campos que ele mesmo capturou. Isto ocorre pois os campos estão localizados dentre de *iframes* que, no momento da captura da interação, não exibem problemas mas na execução acabam por ”esconder”os campos, assim é necessário utilizar uma função para ”acessar”o *iframe*, como pode ser visto na Figura 3, antes de selecionar o elemento desejado.

A Figura 3 exibe duas ”personalizações”no código gerado pelo Selenium IDE, a primeira (em vermelho) é uma linha de código que faz com que o driver ”aguarde”60 segundos antes de procurar pelo campo a ser selecionado. Isto evita que o driver busque por um campo, antes da página ser carregada, e não o encontre (o que causa erro na execução dos teste). A segunda linha de código, em verde, exibe o código que ”acessa”o *iframe*, através da ID, antes de executar a ação e isto também evita que o driver não encontre o elemento no momento do teste.

```
public void select_process(){
    driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);

    driver.findElement(By.xpath("//div[@id='ProcessBrowserContainer']/table/tbody/tr[2]/td/div/ta

    driver.switchTo().frame(driver.findElement(By.id("Solicitar_ACG--1.0")));

    driver.findElement(By.cssSelector("button.bonita_form_button")).click();
}
```

Figura 3. Métodos Java

Como o processo testado é o mesmo nos dois BPMS, o cenário de teste também é o mesmo. O cenário nada mais é do que a definição, em ordem de execução, dos passos que são executados na aplicação bem como os resultados esperados. A Figura **??** exibe o cenário utilizado nos testes em questão.

Além dos problemas comuns as duas ferramentas citados acima, a ferramenta Activiti demonstrou um outro problema. Diferente da gravação com o BOS, o Selenium IDE não captura toda a interação do usuário com a aplicação, de fato, na etapa de login a IDE captura apenas o acesso a página e o ”clique”ao botão de login, ou seja, não captura o preenchimento dos campos ”Usuário”e ”Senha”. Este problema se repete com alguns outros elementos durante a gravação de teste, acredita-se que esse problema ocorra devido a estrutura da página web em questão, estrutura esta que pode conter elementos com os quais o Selenium não trabalhe bem como *divs*,*frames* e *scripts*, por exemplo. No entanto, este problema não impossibilita a criação e execução do teste.

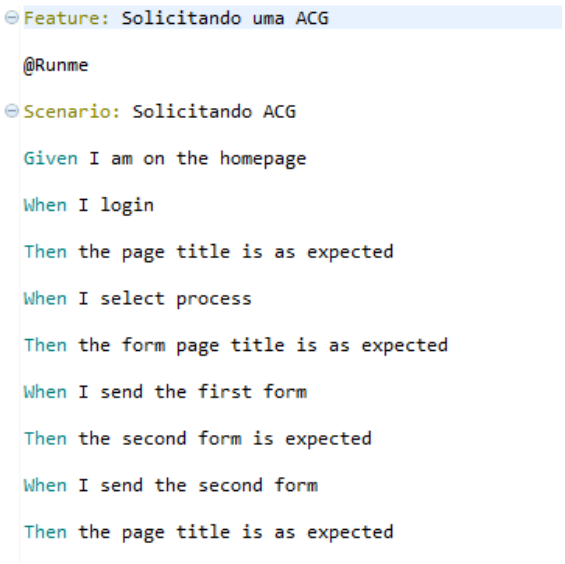

Figura 4. Cenário de teste

Na criação dos métodos correspondentes a cada etapa do processo foi utilizado o código do gerado pelo Selenium IDE mas, como esta não captura todas as interações, a execução do teste falha. Para contornar esse problema, foi necessário estudar a estrutura das páginas que web, localizar os elementos faltantes e então adicionar o código para acessá-los nos respectivos métodos.

4.2.1. Resultados

O teste funcional da ferramenta Bonita Open Solution e da ferramenta Activiti mostrou-se mais viável do que o teste de carga, o teste funcional não utiliza chaves identificadoras ou *ids*, como ocorre no teste com o Jmeter, já que uma boa parcela do teste é executada no lado cliente, o que facilita o processo de teste. Pode-se dizer que este teste atingiu todos seus objetivos pois permitiu executar a interação do usuário, bem como criar o código para simular diferentes entradas e testar o a aplicação de diversas formas.

O teste funcional também não é dependente da implementação do BPMS, são necessárias poucas modificações no código gerado e a realização destas é possível apenas inspecionando a estrutura das páginas web.

O Cucumber-JVM torna a implementação dos testes mais rápida e menos trabalhosa, já que a utilização de cenários auxilia na implementação do teste de forma a diminuir a quantidade de código que precisaria ser escrita sem o uso de uma ferramenta como esta.

O teste funcional com o Selenium+Cucumber pode encontrar problemas ao testar várias tarefas ou todos os possíveis fluxos do processo alvo pois, como já foi citado, é necessário fazer algumas modificações no código gerado pelo Selenium e essas modificações, dependendo do volume de tarefas/fluxos, podem ser muito trabalhosas ou até inviabilizar o teste.

5. Trabalhos Relacionados

6. Conclusão

Um dos objetivos deste trabalho era explorar soluções para teste automatizado de aplicações em BPM, pode se dizer que esse objetivo foi atingido pois, durante a execução do trabalho, foi possível obter várias conclusões sobre o teste automatizado deste tipo de software.

Uma das abordagens adotadas foi de que, por aplicações BPM serem aplicações WEB, poderiam ser tratadas e testadas como um software em geral, utilizando ferramentas consagradas para tal. De fato, esta abordagem mostrou algumas desvantagens e dificuldades relacionadas a particularidades das aplicações BPM.

Sobre o ponto de vista do teste de carga, este tipo de teste mostrou-se útil para explicar, e poderia ter sido usado para prever, falhas ocorridas no sistema que deu origem a este trabalho que foi criado utilizando o BPMS Bonita Open Solution. Também mostrou-se um teste trabalhoso para ser executado dependendo do número de tarefas a serem testadas e também inviável, dependendo do BPMS escolhido. A experiência com duas ferramentas fortaleceu essa conclusão pois ocorreram duas situações distinas, com o Bonita Open Solution o teste mostrou-se válido porém, com o BPMS Activiti o teste foi impossível de ser executado. O BPMS deve ser avaliado antes da execução do teste para verificar se este é vantajoso ou não.

Quanto ao teste funcional, a abordagem obteve maior sucesso na execução dos teste já que não há tanto dependência das ferramentas durante a criação dos testes. O maior problema encontrado foi que a tarefa de teste pode vir a ser muito trabalhosa, principalmente quando deseja-se testar todas tarefas e fluxos possíveis que um processo de negócio pode ter. Nas ferramentas abordadas no neste trabalho ou nas ferramentas estudadas e tabeladas, não foi encontrado nenhum tipo de suporte a teste automatizado dos processos, uma particularidade dos processos é que estes podem possuir diversos caminhos e estes caminhos precisam ser testados. Na Tabela 1 foram exibidos cujos testes foram executados manualmente e, com base no estudo executado neste trabalho e nas ferramentas estudadas, pode-se afirmar que atualmente é muito difícil encontrar um BPMS com suporte a este tipo de teste.

Com o aprofundamento nas ferramentas abordadas nesse trabalho, bem como com o estudo realizado para escolhe-las, pode se afirmar que o campo de teste automatizado em ferramentas BPMS ainda é muito pouco explorado mas, como foi visto na aplicação citada, aplicações BPM estão suscetíveis a erros tanto quanto outras aplicações e poderiam se beneficiar com o avanço desta área.

Como trabalhos futuros, pode-se destacar a continuação exploração de ferramentas de geração de casos de teste, na hipótese de que possam ajudar a alinhar os testes com as saídas e entradas do processo. Outra via que merece ser explorada são os testes de regressão, para auxiliar a encontrar possíveis problemas após alterações no processo, que podem ser frequentes dependendo do caso.

Referências

ABPMP (2012). *Guide to the Business Process Management Body of Knowledge (BPM CBOK)*. Association of Business Process Management Professionals, 2nd edition.

RIVEROL, L. et al. Aplicando Design e Avaliacao de Usabilidade para Melhorar a Qualidade de um Aplicativo Web Móvel. In: XIII SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE. **Anais...** [S.l.: s.n.], 2014.

VALENTIM, N. et al. Avaliando a qualidade de um aplicativo web móvel através de um teste de usabilidade: um relato de experiência. In: XIII SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE. **Anais...** [S.l.: s.n.], 2014.

CHIAVEGATTO, R. et al. Especificação e Automação Colaborativas de Testes utilizando a técnica BDD. In: XII SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE. **Anais...** [S.l.: s.n.], 2013. p.334–341.

BonitaSoft (2012). Bonitasoft open source workflow and bpm software.

Chiavegatto, R., Pinheiro, V., Vieira, A. F., Clineu, J., Oliveira, E. H., Barroso, E., Amorim, A., e Conte, T. (2013). Especificação e automação colaborativas de testes utilizando a técnica BDD. In *XII Simpósio Brasileiro de Qualidade de Software*, pages 334–341.

de Moura, J. L., Lunardi, G. M., Charão, A. S., Barcelos, P. P., e de Oliveira Stein, B. (2013). Gestão de processos de negócio em curso de sistemas de informação: Relato de experiência utilizando software livre. In *IX Simpósio Brasileiro de Sistemas de Informação*, pages 206–217.

Graham, D. e Fewster, M. (2012). *Experiences of Test Automation: Case Studies of Software Test Automation*. Addison-Wesley.

Ko, R. K. L. (2009). A computer scientist's introductory guide to business process management (bpm). *Crossroads*, 15(4):4:11–4:18.

Kolawa, A. e Huizinga, D. (2007). *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press.

Nguyen, H. Q., Johnson, B., e Hackett, M. (2003). *Testing Applications on the Web: Test Planning for Mobile and Internet-Based Systems*. Wiley.

van der Aalst, W. M. P. (2013). Business process management: A comprehensive survey. *ISRN Software Engineering*, 2013(507984).

Weske, M. (2012). *Business Process Management: Concepts, Languages, Architectures*. Springer, 2nd edition.