Automação de Testes em Aplicações de BPMS: 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 relata uma experiência de teste automatizado de uma aplicação desenvolvida com o apoio de sistemas de gestão de processos de negócio (Business Process Management Systems – BPMS). Para isso, implementou-se um mesmo processo em dois diferentes BPMS: Bonita e Activiti. Buscou-se submeter este processo a dois tipos de teste: testes de carga e testes funcionais, utilizando-se as ferramentas de teste Apache JMeter, Cucumber e Selenium. Os resultados evidenciam limitações e oportunidades na automação de testes deste tipo de aplicação.*

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

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. 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.

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]. 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. Aplicação Alvo de Testes

A aplicação alvo deste trabalho refere-se a um processo comum em instituições de ensino superior: a apreciação de Atividades Complementares de Graduação (ACGs), ou seja, atividades que formam a parte flexível do currículo de graduandos (participação em palestras, eventos, projetos, etc.). Em um trabalho anterior [de Moura et al. 2013], apresentou-se a modelagem e implantação desse processo. Sua representação em BPMN, na Figura 1, revela um total de 11 tarefas distribuídas em 5 divisões de responsabilidade (Aluno, Tutor, etc.).

No trabalho anterior, implementou-se o processo com a ferramenta Bonita Open Solution [BonitaSoft 2012], um BPMS de código aberto reconhecido no mundo corporativo [Gartner 2010, Research 2013]. A aplicação resultante possui vários formulários Web relativos a cada divisão de responsabilidade, sendo o primeiro deles destinado ao preenchimento de dados pelo aluno. De acordo com o tipo de atividade complementar (eventos,

projetos, etc.), o fluxo é direcionado para os responsáveis pela avaliação e validação da ACG. Nota-se, na Figura 1, que o processo possui diversos desvios condicionais, o que leva a mais de 15 caminhos possíveis no processo.

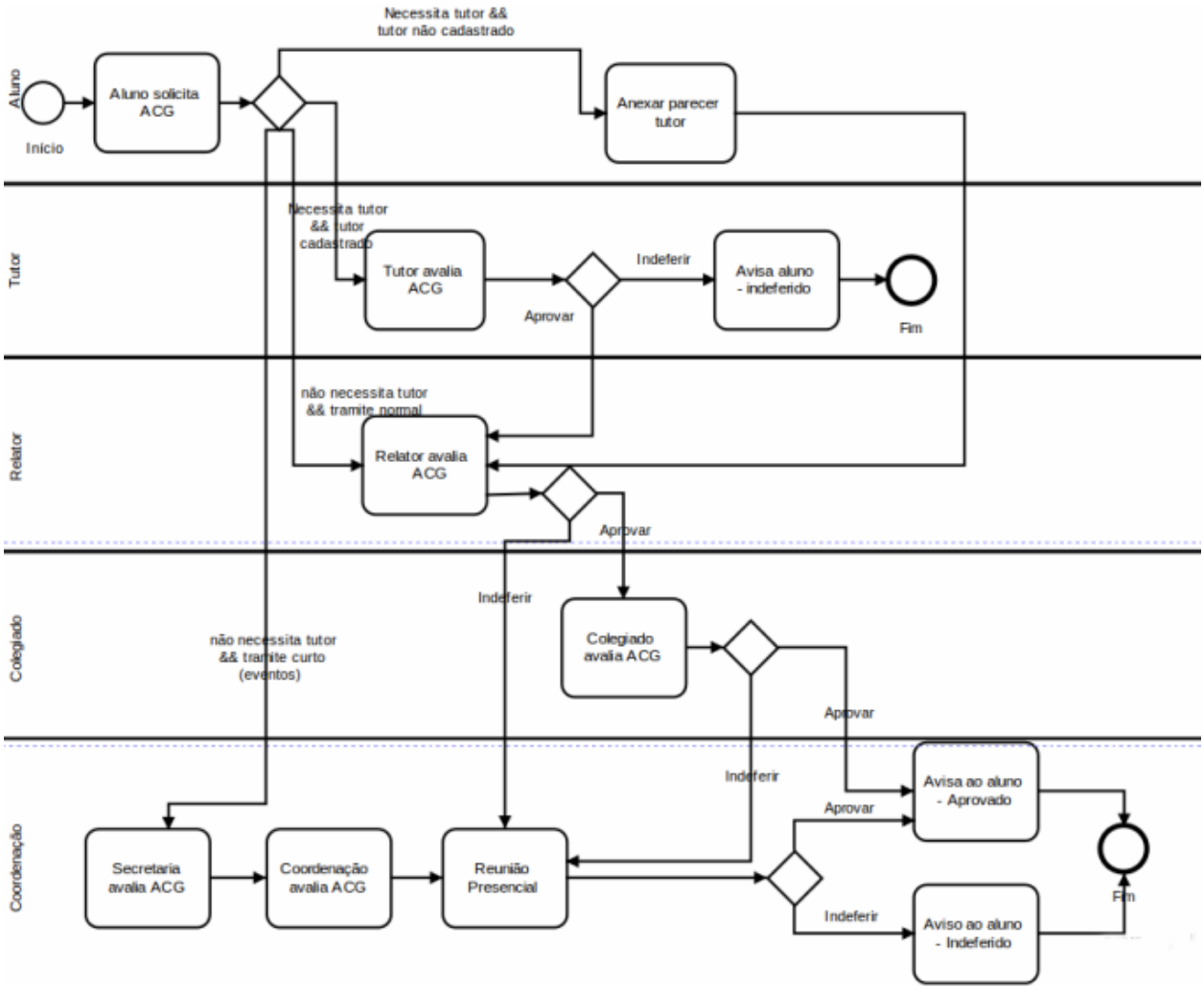

Figura 1. Diagrama do processo em BPMN

A aplicação foi submetida a testes funcionais realizados manualmente, além de testes de aceitação realizados com um grupo de usuários reais. Entretanto, com algumas semanas em operação, surgiram problemas: instâncias do processo falharam devido a entradas inesperadas, serviços não foram restabelecidos corretamente após serem interrompidos e houve sobrecarga devido ao grande número de casos abertos numa data limite. Essa experiência motivou a busca de soluções para automação de testes.

Verificou-se, no entanto, que o BPMS utilizado não possuía suporte a nenhum tipo de teste automatizado. Buscou-se outros BPMS com licenças *open source*, que pudessem implementar o processo em questão e que oferecessem suporte a testes. Dentre as ferramentas analisadas (TIBCO, Syddle Seed, Activiti, Process Maker, Intalio), nenhuma oferecia suporte a testes automatizados, apenas a ferramenta Activiti cita a possibilidade de um teste de unidade aliado ao JUnit porém não existem informações extras sobre essa alternativa. Por isso partiu-se para outra opção: utilizar ferramentas de teste externas ao

	BOS	Activiti
Captura das requisições usando apenas JMeter	Problemas com GWT	Problemas com GWT
Foi possível capturar e exportar as requisições HTTP?	Sim (Utilizando BlazeMeter)	Sim (Utilizando BlazeMeter)
Ocorreu problema com dependência das tarefas do processo?	Sim	Sim
Foi possível identificar a solução para a dependência das tarefas?	Sim	Não
Foi possível configurar e executar os testes?	Sim	Não

Tabela 1. Teste de carga

BPMS. Mesmo assim, decidiu-se implementar a mesma aplicação usando outro BPMS, a fim de ampliar a experiência e, possivelmente, identificar semelhanças e diferenças no teste automatizado de implementações com diferentes BPMS. A ferramenta escolhida foi Activiti [ACT 2014], um BPMS de código-aberto que trabalha com a mesma versão da notação BPMN usada no BOS, permitindo importar na íntegra o processo originalmente criado com o Bonita Open Solution. Os formulários Web, no entanto, não puderam ser importados e foram recriados manualmente.

4. Descrição e Execução dos Testes

No planejamento de testes automatizados, priorizou-se o teste de etapas que de fato revelaram problemas durante a operação. 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 pré-definidas. **Nenhum destes tipos de teste possui suporte nos BPMS Bonita e Activiti**, que incluem somente funcionalidades limitadas de simulação e depuração de execução dos processos. Assim, realizou-se um levantamento de ferramentas externas aos BPMS, destinadas ao teste de aplicações Web. Selecionou-se as julgadas mais promissoras, antes de partir-se para o detalhamento e execução dos testes.

4.1. Teste de Carga

Existem diversas ferramentas para automação de testes de carga e desempenho em aplicações Web, dentre as quais pode-se citar: JMeter [JMeter 2012], The Grinder [**?**] e 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. 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

Etapa x Usuários	Login	Pág. Inicial	Seleção processo	Form. Inicial	Enviar form.
1	126	32	38	80	73
50	597	191	179	368	152
100	1972	571	552	760	694
200	10.149	3.239	934	2.122	1.918

Tabela 2. Tempos médios de resposta (em milissegundos) em cada etapa e quantidade de usuários

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.

Os resultados, em milissegundos, de cada etapa com a variação do número de usuários pode ser visto na Tabela 2. Os resultados mais alarmantes são os que retratam 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. 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.

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

	BOS	Activiti
Componentes WEB	HTML, CSS, Ajax	HTML, CSS, Ajax
Captura da interação do usuário utilizando o Selenium	Total	Parcial (necessitou de inserção manual de alguns campos)
Foi possível exportar o código gerado pelo Selenium?	Sim	Sim
Reconhecimento de todos os campos capturados SEM alteração de código	Parcial	Parcial
Reconhecimento de todos os campos capturados COM alteração de código	Total	Total
Foi possível criar o cenário e executar o teste?	Sim	Sim

Tabela 3. Teste funcional

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 utilizava 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).

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.

Algumas personalizações são necessárias para a execução com sucesso 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* antes de selecionar o elemento desejado, garantindo que não ocorra erro durante a execução.

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.

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. Conclusão

O maior objetivo deste trabalho foi 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.

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 já foi implementado. Também mostrou-se um teste trabalhoso, ou até inviável, dependendo do número de tarefas no processo e do BPMS. A experiência com duas ferramentas fortaleceu essa conclusão pois ocorreram duas situações distintas, 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 testes já que não há tanta dependência das ferramentas BPMS 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.

Com o aprofundamento nas ferramentas abordadas nesse trabalho, pode se afirmar que o campo de teste automatizado em ferramentas BPMS ainda é 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.

Referências

(2014). Activiti bpms.

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

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.

Gartner (2010). Magic quadrant for business process management suites.

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

JMeter (2012). Aplicação desktop projetada para testes de carga e medidas de performance.

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

Research, F. (2013). The forrester wave: BPM suites, Q1 2013.

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.