# Automação de Testes em Aplicações de BPMS: um Relato de Experiência

**Jéssica 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 usando dois diferentes BPMS: Bonita e Activiti. Submeteu-se as aplicações Web resultantes a testes de carga e testes funcionais, utilizando-se as ferramentas Apache JMeter, Selenium e Cucumber. Os resultados evidenciam a viabilidade e as limitações na automação de testes deste tipo de aplicação.*

***Abstract.*** *This article describes an experience of test automation of an application developed with the support of Business Process Management Systems - BPMS. For this purpose, we have implemented the same process using two different BPMS: Bonita and Activiti. We submit the resulting Web applications to two types of tests (load tests and functional tests), using test tools Apache JMeter, Selenium and Cucumber. The results show the feasibility and limitations of test automation for this type of application.*

## 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]. Associados a isso, surgiram os sistemas BPM (*Business Process Management Systems* – BPMS), que são ferramentas de software para apoio ao ciclo de vida da gestão de processos de negócio.

Dentre os diversos BPMS disponíveis atualmente, é comum encontrar ferramentas com suporte à modelagem, configuração e execução de processos de negócio. Em muitos casos, os BPMS abreviam o desenvolvimento de software, entregando aplicações Web completas para execução dos processos, usando tecnologias atuais e exigindo pouca escrita de código. Por outro lado, algumas tarefas como verificação e testes ainda são consideradas um desafio nesta área [van der Aalst 2013]. Em particular, o teste automatizado de aplicações de BPMS é pouco abordado, tanto pela comunidade da área de BPM [Weske 2012] como da área de testes de software [Graham e Fewster 2012]. No entanto, a falta de automação nos testes pode levar a problemas durante a implementação e execução de processos de negócio, ainda mais quando se tratam de processos com muitas tarefas e fluxos de trabalho, que levam facilmente a explosões combinatórias. O

propósito deste trabalho foi explorar soluções para teste automatizado de um processo implementado em BPMS. Para isso, partiu-se de uma aplicação real, em que testes manuais se revelaram insuficientes [de Moura et al. 2013]. No presente artigo, relata-se a experiência com testes automáticos nesta aplicação, utilizando-se ferramentas *open source*. A aplicação é apresentada na seção 3, após uma discussão sobre BPM e testes na seção 2. Na sequência, a seção 4 apresenta os métodos, ferramentas e resultados obtidos em cada tipo de teste. Por fim, a seção 5 resume as lições aprendidas.

## 2. BPM e Testes

O termo BPM pode ser usado com ênfases diferentes, às vezes com foco em tecnologia (software) e outras vezes em gestão. Mesmo assim, a área tem convergido no entendimento do ciclo de vida de aplicações de BPM, que envolve as atividades de análise, modelagem, execução, monitoramento e otimização [ABPMP 2012]. Também há convergência sobre o padrão BPMN (*Business Process Model and Notation*) para expressar a modelagem de processos.

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 de processos em BPMN, controle da execução e monitoramento de atividades dos processos [Forrester Research 2013]. Há uma tendência dos BPMS em abreviar o desenvolvimento de software, por exemplo através de geradores de formulários Web associados a tarefas dos processos [Winter Green Research 2013]. Nota-se, no entanto, que a preocupação com testes não fica evidente nas ferramentas BPMS. De fato, examinando-se o material promocional e a documentação disponível sobre os principais BPMS, observa-se uma ênfase em etapas de modelagem e execução.

Por outro lado, a importância dos testes é amplamente reconhecida em engenharia de software [Bourque e Fairley 2014]. Considerando que aplicações de BPMS são geralmente sistemas baseados na Web, pode-se supor que sejam testadas com sucesso usando-se abordagens consagradas, como por exemplo testes de carga ou testes funcionais do tipo caixa-preta. Há também quem argumente que o teste de aplicações de BPM difira do teste de aplicações Web tradicionais [Chetty 2014], porém não foram encontradas mais referências aprofundando esse ponto de vista. Essa constatação reforçou a motivação para o presente trabalho.

## 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, implementação 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 BPM[1], um BPMS de código aberto reconhecido no mundo corpora-

[1]Bonita BPM (antes denominado Bonita Open Solution). Disponível em: www.bonitasoft.com.

tivo [Forrester 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 (Aluno solicita ACG). 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.

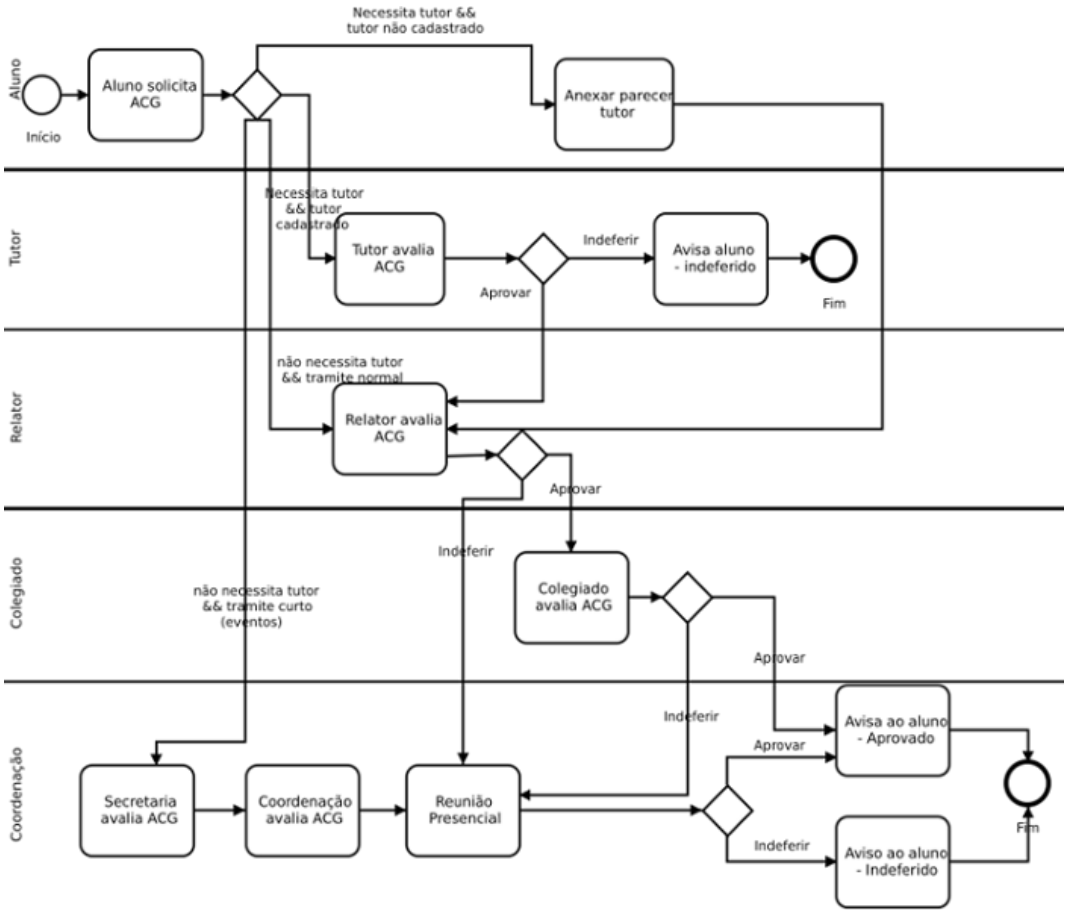


**Figura 1. Diagrama do processo em BPMN**

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* ou *freeware*, que pudessem implementar o processo em questão e que oferecessem suporte a testes. A preferência por este tipo de licença foi devida à sua flexibilidade e viabilidade financeira para o "cliente" deste projeto: uma instituição pública de ensino. Dentre as ferramentas

analisadas (TIBCO[2], Activiti[3], Process Maker[4], Intalio[5]), nenhuma oferecia evidente suporte a testes automatizados. Apenas a ferramenta Activiti citava a possibilidade de testes de unidade aliados ao JUnit, porém sem muitas informações sobre essa alternativa. Por isso partiu-se para outra opção: utilizar ferramentas de teste externas ao BPMS.

Adicionalmente, 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. Porém, não tinha-se como objetivo eleger o melhor BPMS, mas sim avaliar suporte aos testes considerados.

A ferramenta escolhida foi Activiti, um BPMS baseado em tecnologias Java, assim como Bonita, permitindo trabalhar com as mesmas tecnologias do lado servidor (Tomcat e MySQL, no caso em questão). Esse BPMS também trabalha com a mesma versão da notação BPMN usada no Bonita, permitindo importar na íntegra o processo originalmente criado. Os formulários Web criados com ajuda do Bonita, no entanto, não puderam ser importados e tiveram de ser recriados com Activiti.

## 4. Descrição e Execução dos Testes

No planejamento de testes automatizados, priorizou-se o teste de aspectos que de fato revelaram problemas durante a implantação, no trabalho precedente. 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. Embora o Activiti citasse suporte a testes de unidade, não explorou-se esta opção por entender-se que seria menos prioritária frente aos problemas observados. Assim, realizou-se um levantamento de ferramentas destinadas ao teste de aplicações Web e selecionou-se as julgadas mais promissoras, antes de partir-se para o detalhamento e execução dos testes. Os *scripts* que configuram os testes estão disponíveis para consulta em http://www.inf.ufsm.br/ andrea/bpmtest-scripts.zip.

### 4.1. Testes de Carga

Testes de carga em aplicações Web são tipicamente realizados gerando-se múltiplas requisições HTTP ao servidor, de forma controlada. Para isso, uma etapa crítica é a identificação das requisições que devem ser reproduzidas. Existem diversas ferramentas que se propõem a facilitar este tipo de teste, dentre as quais pode-se citar: JMeter[6], The Grinder[7] e WebLOAD[8]. Todas são ferramentas *open source* e possuem diversas opções, mas escolheu-se a ferramenta JMeter pela sua funcionalidade "proxy server" que é de grande auxílio à captura das requisições.

[2]TIBCO. Disponível em: www.tibco.com.
[3]Activiti. Disponível em: www.activiti.org.
[4]Process Maker. Disponível em: www.processmaker.com.
[5]Intalio. Disponível em: www.intalio.com.
[6]Apache JMeter. Disponível em: www.jmeter.apache.org.
[7]The Grinder. Disponível em: www.grinder.sourceforge.net/.
[8]WebLOAD. Disponível em: radview.com/webload-download/.

| 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 1. Tempos médios de resposta, em milissegundos**

Um teste da aplicação com JMeter consiste em quatro etapas: capturar as requisições HTTP, exportar as requisições (formato .jrxml para JMeter), configurar o plano de teste e, por fim, executar o teste no JMeter. A aplicação gera diferentes requisições HTTP que precisam ser identificadas e interpretadas, para serem reproduzidas automaticamente. O emprego de tecnologias Web com processamento assíncrono, do lado do cliente, pode dificultar esta etapa, pois uma ação do usuário pode não gerar imediatamente uma requisição ao servidor. Além disso, em aplicações de BPMS, diversos usuários atuam sobre diferentes tarefas de um mesmo processo, de modo que as requisições HTTP carregam chaves identificando usuários e processos.

No caso da aplicação criada com Bonita, verificou-se que 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, que identifica cada execução do processo como única, sendo criada pelo servidor no momento em que o usuário inicia o processo. Para executar os testes, portanto, foi necessário localizar a requisição em que essas chaves são geradas e utilizar a ferramenta "Extrator de Expressão Regular" do JMeter para obter seus valores. Já na aplicação criada com Activiti, foi inviável identificar a requisição em que as chaves são geradas, pois não há uma requisição cujo retorno (resposta do servidor) contenha valores de chaves. 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 consequência, esta não pode ser capturada e reproduzida no JMeter.

### 4.1.1. Resultados dos Testes de Carga

Devido aos problemas relatados na seção anterior, os testes de carga só puderam ser realizados com a aplicação criada com Bonita. A fim de testar o comportamento do sistema com diferentes níveis de carga, foram executados testes com 1, 50, 100 e 200 usuários virtuais/acessos simultâneos e foram analisadas as requisições referentes às etapas: efetuar login, exibir a página inicial do Bonita, selecionar o processo, exibir o formulário inicial (Aluno solicita ACG) 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.

Os tempos de resposta de cada etapa, em função do número de usuários, podem ser vistos na Tabela 1. Os resultados mais alarmantes são para 200 usuários virtuais, em que o tempo médio de resposta na requisição de login foi de 10.149 ms, ou seja, aproximadamente 10 s, o que é um tempo de resposta alto. A média de tempo de resposta para todas requisições foi de 3.111 ms (ou seja, 3 s), 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, as requisições obtiveram uma taxa de erro de 7.82%. Os erros ocorreram

devido a requisições sem resposta.

De forma geral, portanto, o teste de carga com JMeter atingiu seus objetivos e ajuda a explicar a sobrecarga que ocorreu com a aplicação em produção, quando muitos alunos tentaram acessar o formulário inicial numa data limite. No entanto, é importante ressaltar que este teste foi executado apenas em etapas iniciais do processo e, mesmo assim, já foi trabalhoso e consumiu algumas horas de preparação, por exigir uma análise profunda das requisições HTTP para executar os testes com sucesso. Ao todo, foram capturadas cerca de 100 requisições só nestas etapas, portanto estima-se que o teste de uma tarefa mais ao final do processo possa se tornar inviável com JMeter, por demandar a identificação e interpretação de muitas requisições. Outra observação importante, nesta experiência, é que esta abordagem de teste sofre com a dependência das tecnologias Web empregadas pelo BPMS.

### 4.2. Testes Funcionais

Para executar testes funcionais em aplicações Web, pode-se utilizar ferramentas livres como Selenium[9], Watir[10] ou Geb[11]. Para este trabalho, escolheu-se a ferramenta Selenium, aliada ao Cucumber-JVM[12] para descrição dos testes. A escolha foi motivada pelo grande número de referências ao Selenium na Web, confirmadas por um trabalho que apresentou resultados satisfatórios com essas ferramentas [Chiavegatto et al. 2013].

Com estas ferramentas, o processo para a execução de um teste funcional é composto das seguintes etapas: captura da interação do usuário com o navegador (Selenium IDE), exportação do código gerado (Selenium IDE), criação do cenário de teste (Cucumber), criação das definições dos passos do teste (Cucumber), criação dos métodos para cada passo (Java) e execução do teste (Selenium WebDriver). O cenário de teste é a definição, em ordem de execução, dos passos que são executados na aplicação, bem como dos resultados esperados. Para este trabalho, definiu-se um cenário em que o aluno faz login e preenche 2 formulários referentes à primeira tarefa do processo (Aluno solicita ACG). Para este cenário, foram criados métodos fornecendo diferentes entradas nos formulários. Como o processo testado é o mesmo em ambos os BPMS, o cenário de teste também é o mesmo.

Para as aplicações de ambos os BPMS, inicialmente ocorreram erros na execução dos testes, relativos à localização de campos nos formulários Web. Verificou-se que os campos estavam localizados dentro de um *iframe* e, embora a captura da interação ocorresse sem problemas, o código gerado não selecionava o *iframe* e por isso não encontrava o campo. Assim, foi necessário utilizar um método do Selenium para acessar o *iframe* antes de selecionar o elemento desejado.

Na aplicação com Activiti, ocorreu também um outro problema. Diferente do que ocorreu com os formulários Web gerados pelo Bonita, o Selenium IDE não capturou toda a interação do usuário com a aplicação. De fato, na etapa de login, o Selenium capturou apenas o acesso à página e o "clique" ao botão de login, ou seja, não capturou o preenchimento dos campos "Usuário" e "Senha". Este problema se repetiu com alguns

[9]Selenium. Disponível em: www.seleniumhq.org.
[10]Watir. Disponível em: www.watir.com.
[11]Geb. Disponível em: www.gebish.org.
[12]Cucumber-JVM. Disponível em: www.github.com/cucumber/cucumber-jvm.

| | Bonita | 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 2. Resumo comparativo sobre o teste funcional**

outros elementos do formulário Web durante a gravação da interação. Acredita-se que o problema ocorra devido à estrutura das página Web, que pode conter elementos que o Selenium não identifique automaticamente, tais como *divs*, *frames* e *scripts*, por exemplo. No entanto, isso não impossibilita a criação e execução dos testes. Para contornar o problema, foi necessário estudar a estrutura das páginas Web em questão, localizar os elementos faltantes e então adicionar o código para acessá-los nos respectivos métodos.

### 4.2.1. Resultados dos Testes Funcionais

Os testes funcionais mostraram-se mais viáveis do que os testes de carga, pois uma boa parcela da interação é executada no lado cliente, sem necessidade de lidar explicitamente das interações com o servidor. Pode-se dizer que este teste atingiu todos seus objetivos, pois permitiu reproduzir a interação do usuário, bem como criar o código para testar as aplicações com diferentes entradas, num cenário envolvendo a tarefa inicial do processo, ampliando a cobertura dos testes. O teste funcional também reproduziu erros encontrados em produção e que tinham passado despercebidos nos testes manuais (entradas não previstas corretamente nos formulários), esses erros foram os mesmos em ambas as ferramentas BPMS, pois os formulários foram configurados de modo equivalente.

O teste funcional também é menos dependente do BPMS. Na Tabela 2 apresenta-se um resumo das principais semelhanças e diferenças encontradas. Em ambos os casos, foram necessárias poucas modificações no código gerado pelo Selenium e Cucumber, bastando para isso inspecionar a estrutura das páginas Web. O Cucumber torna a implementação dos testes mais rápida e menos trabalhosa do que se fosse usado apenas o Selenium, pois abrevia a geração de código alinhado com os cenários de teste. Mesmo assim, caso seja necessário estender os testes a muitas tarefas de um processo, as intervenções manuais no código de teste podem se tornar trabalhosas.

## 5. Considerações Finais

Neste trabalho, explorou-se soluções de teste automatizado em uma aplicação de BPMS. Na ausência de suporte a testes de carga e funcionais nos BPMS Bonita e Activiti, aplicou-se ferramentas de teste voltadas a aplicações Web em geral.

No que concerne ao teste de carga, este mostrou-se útil para explicar falhas observadas no trabalho anterior. Também mostrou-se um teste trabalhoso, ou até inviável, dependendo do BPMS usado. A experiência com dois BPMS fortaleceu essa conclusão, pois ocorreram situações distintas: com Bonita o teste foi bem sucedido, porém com Ac-

tiviti o teste não pôde ser executado, já que não se conseguiu reproduzir todas requisições.

No teste funcional, com a abordagem adotada, obteve-se maior sucesso na execução dos testes e observou-se uma menor dependência dos BPMS, em comparação com o teste anterior. A tarefa de teste pode vir a ser trabalhosa, principalmente quando deseja-se testar muitas tarefas e fluxos que um processo de negócio pode ter.

De modo geral, como lições aprendidas temos que, sob certas condições, é viável testar aplicações de BPMS com ferramentas de teste para sistemas Web. A principal condição, no caso considerado, foi o direcionamento dos testes a uma tarefa inicial do processo, identificada como gargalo. Como o tempo e esforço para uma única tarefa foi significativo, essa abordagem pode se tornar inviável caso seja necessário estender os testes a muitas tarefas de um processo.

Outro aspecto a ser considerado é que, quando o BPMS implementa a interação com o usuário e/ou com servidores, o desenvolvedor deixa de escolher e controlar todas as tecnologias utilizadas. Essa facilidade, no entanto, pode dificultar a automação de testes com ferramentas externas, que se beneficiam de dados sobre a implementação (por exemplo, *iframes*, Ajax, na experiência em questão).

Embora a experiência deste trabalho não tenha sido exaustiva, pôde-se notar que o suporte a testes automatizados é pouco explorado em BPMS. Entende-se que esta seria uma funcionalidade bem-vinda, supondo-se que abreviaria uma etapa essencial para garantir a qualidade do software resultante.

## Referências


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

Bourque, P. e Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0*. IEEE Computer Society.

Chetty, N. K. (2014). How to perform workflow testing for BPM applications. Disponível em: http://www.evoketechnologies.com.

Chiavegatto, R. et al. (2013). Especificação e automação colaborativas de testes utilizando a técnica BDD. In *XII Simpósio Brasileiro de Qualidade de Software*, pp. 334–341.

de Moura, J. L. et al. (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*, pp. 206–217.

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

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

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.

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