Test automation on BPMS applications: an Experience Report

Jessica Lasch de Moura and Andrea Schwertner Charao

Ncleo de Cincia da Computao
Universidade Federal de Santa Maria – UFSM

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 Introduction

Business Process Management has aroused the interest of companies and the scientific community, both for their benefits as per their challenges. Is called of BPM a set of concepts, methods and techniques to support the modeling, administration, configuration and analysis of business processes [10]. Associated with this, emerged the BPM systems (*Business Process Management Systems* – BPMS), which are software tools to support the life cycle of business process management.

Among the many BPMS currently available, it is common to find tools that support modeling, configuration and execution of business processes. In many cases, BPMS shorten the software development, delivering complete Web applications for process execution, using current technologies and requiring little code writing. On the other hand, some tasks such as checking and tests are still considered a challenge in this area [9]. In particular, the automated test of BPMS applications is rarely addressed by both the BPM community [10] as software testing area

However, the lack of automation in testing can lead to problems during implementation and execution of business processes, especially when treating processes with many tasks and workflows, which easily lead to combinatorial explosion.

The purpose of this study was to explore solutions for automated testing of a process implemented in BPMS. For this, it started with a real application where manual testing proved inadequate [5]. In this article, we report the experience with automated testing this application, using open source tools. The application is presented in section 3, after a discussion of BPM and tests in section 2. Further , the section 4 presents the methods, tools and results of each type of test. Finally, the section 5 summarizes the lessons learned.

2 BPM and Testing

The BPM term can be used with different emphases, sometimes focusing on technology (software) and other times management. Still, the area has converged in understanding the lifecycle of BPM applications, involving the analysis of activities, modeling, execution, monitoring and optimization [1]. There is also convergence on the pattern BPMN (*Business Process Model and Notation*) to express the process modeling.

BPM systems (BPMS) has been claimed as essential tools to support the activities of this life cycle. Currently, it can be said that a typical BPMS provides resources for definition and process modeling in BPMN, execution control and activity monitoring of processes [6]. There is a tendency of BPMS in shortening software development, for example via web forms generators associated with process tasks [11]. Note, however, that the concern with testing is not evident in the BPMS tools. Indeed, examining the promotional material and documentation available on the main BPMS, there is an emphasis on modeling and execution stages.

Moreover, the importance of the tests is widely recognized in software engineering [2]. Whereas BPMS applications are usually Web-based systems, it can be assumed that it can be successfully tested using dedicated approaches, such as load tests or functional testing of the black-box type. There are also those who argue that BPM applications test differs from traditional Web applications testing [3], but there were no more references to deepen this view. This finding reinforced the motivation for this work.

3 Test Target Application

The target application of this work refers to a common process in higher education institutions: the appreciation of complementary activities, ie, activities that form the flexible part of the undergraduate curriculum (participation in lectures, events, projects , etc.). In a previous work [5], presented the modeling, implementation and deployment of this process. Their representation in BPMN, in Figure 1, reveals a total of 11 tasks span responsibility 5 divisions.

In previous work, we implemented the process with the Bonita BPM tool [1], an open source BPMS recognized in the corporate world [6]. The resulting application has several Web forms for each division of responsibility, being the first one for the data population by the student. According to the type of complementary activities, the flow is directed to those responsible for the assessment and validation of activity. Note, in Figure 1, the process has several gateways, which leads to over 15 possible paths in the process.

The application was subjected to functional testing performed manually, in addition to acceptance testing with a group of real users. However, a few weeks in operation, problems arose: Process instances failed due to unexpected inputs,

[1] Bonita BPM (formerly Bonita Open Solution). Available at:. Www.bonitasoft.com

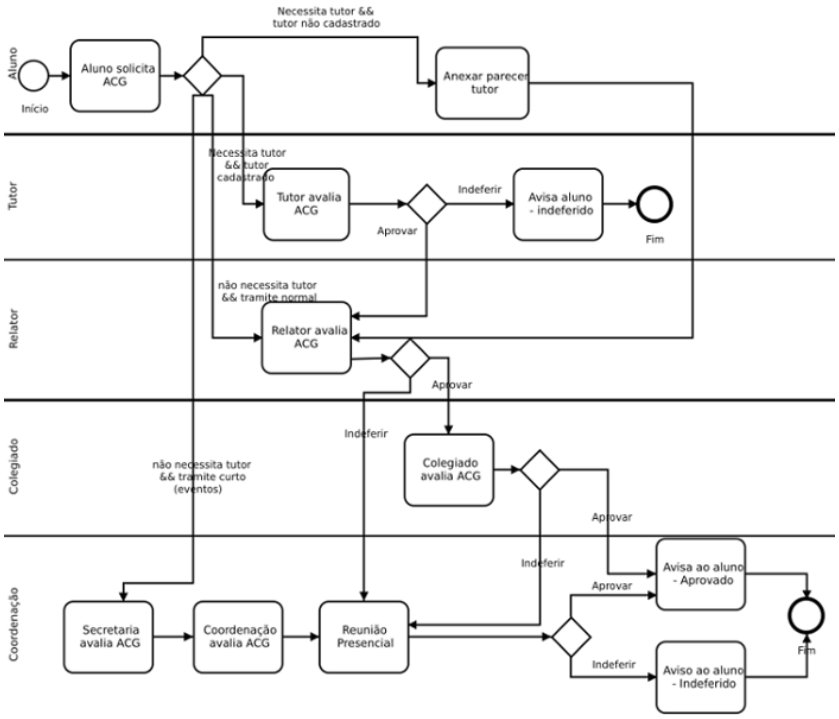

Fig. 1. Diagrama do processo em BPMN

services were not restored properly after being interrupted and there was overloaded because of the number of cases opened on a deadline. This experience led to the search for solutions for test automation.

It was found, however, that the BPMS used does not support any type of automated testing. We attempted to other BPMS with open source or freeware licenses, which could implement the process in question and offered testing support. The preference for this type of license was due to its flexibility and affordability to the "client" this project: a public educational institution. Among the analyzed tools (TIBCO[2], Activiti[3], Process Maker[4], Intalio[5]), offered no clear support for automated testing. Just Activiti tool cited the possibility of allies unit tests to JUnit, but not much information about this alternative. So we went to another option: use test tools external to BPMS.

In addition, it was decided to deploy the same application using other BPMS in order to broaden the experience and possibly identify similarities and dif-

[2] TIBCO. Available at: www.tibco.com
[3] Activiti. Available at: www.activiti.org
[4] Process Maker. Available at: www.processmaker.com
[5] Intalio. Available at: www.intalio.com

ferences in automated test implementations with different BPMS. Yet it had intended to elect the best BPMS, but to assess support for both tests.

The tool chosen was Activiti, a BPMS based on Java technologies, like Bonita, allowing you to work with the same server-side technologies (Tomcat and MySQL, in this case). This BPMS also works with the same version of the BPMN notation used in Bonita, allowing import in full the process originally created. Web forms created with Bonita's help, however, could not be imported and had to be recreated with Activiti.

4 Description and Performance of tests

In planning automated tests, priority was given to the test of aspects that in fact revealed problems during deployment, in previous work. The selected tests were: (a) load tests, which are one type of performance testing, to evaluate the opposite behavior of the system to a large number of requests, and (b) functional tests to verify the produced system output from pre-defined inputs. None of these types of tests is supported in BPMS Bonita and Activiti, which include only limited functionality simulation and debugging process execution. Although Activiti quoting support unit tests not be exploited for this option to be understood that priority would be less compared to the observed problems. Thus, there was a survey of tools intended for Web application testing and was selected the most promising judged before departing to the detailing and execution of tests. The scripts that configure the tests are available for consultation in http://www.inf.ufsm.br/ andrea/bpmtest-scripts.zip.

4.1 Load Tests

Load tests on web applications is typically performed by generating multiple HTTP requests to the server in a controlled manner. For this, a critical step is to identify the requests that should be played. There are several tools that purport to facilitate this type of testing, among which we can mention: JMeter[6], The Grinder[7] and WebLOAD[8]. All tools are open source and have several options, but chose to JMeter tool for its functionality "proxy server" which is a great help to the capture of requests.

An application test with JMeter consists of four steps: capture HTTP requests, export requests (.jrxml format for JMeter), set up the test plan and, finally, run the test in JMeter. The application generates different HTTP requests that need to be identified and interpreted to be played automatically. The use of Web technologies with asynchronous processing, client-side, can hinder this step because a user action can not immediately generate a request to the server. Moreover, in applications BPMS, many users act on different tasks

[6] Apache JMeter. . Available at: www.jmeter.apache.org
[7] The Grinder. Available at: www.grinder.sourceforge.net/
[8] WebLOAD. Available at: www.radview.com/webload-download/

in a same process, so that the HTTP requests that carry user identifying keys and processes.

In the case of application created with Bonita, it was found that there is a session identifier key that is generated at the time user accesses the system and other instance identifying key, that is, identifies each execution of the process as only being created by the server when the user initiates the process. To run the tests, so it was necessary to locate the request in which these keys are generated and use the "Regular Expression Extractor" tool of JMeter to get their values. In the application created with Activiti, it was impossible to identify the request in which the keys are generated, as there is not a request whose return (server response) contains key values. This suggests that the generation of the identifying key is made internally by the BPMS, ie not in an HTTP request and therefore this can not be captured and reproduced in JMeter.

Load Test Results Because of the problems reported in the previous section, load tests could only be executed in the implementation created with Bonita. In order to test the system behavior with different load levels, tests were performed with 1, 50, 100 and 200 virtual users and requests were analyzed regarding the essential steps to begin the process: login, view the Bonita's homepage, select the process, show the first form (Student requests ACG) and send the completed form. The application is hosted on a SGI Altix XE 210 server with 24GB of RAM, accessible by a Fast Ethernet network.

The response times of each stage, depending on the number of users, can be seen in Table 1. The most alarming results are for 200 virtual users, where the average response time on the login request was for 10,149 ms, or approximately 10 seconds, which is a high response time. The average response time for all requests was 3111 ms (i.e., 3 s), and the standard deviation was 13,088. Besides high response times, the test showed virtual users 200 with error rates in some requests which were not found with a smaller number of users. For example, a request that has performed login showed a rate of 2% error and, in total, the requests have obtained an error rate of 7.82%. The error occurred because of unanswered requests.

Users	Login	Home Page	Process selection	First form	Send 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

Table 1. Average response times, in milliseconds

Overall, therefore, the load testing with JMeter achieved its objectives and helps explain the overload that occurred with the application in production, when many students tried to access the initial form in a deadline. However, it

is important to note that this test was performed only in the early stages of the process and, even then, was already laborious and consumed a few hours of preparation, because it requires a deep analysis of HTTP requests to run the tests successfully. In all, about 100 requests were captured only in these steps, therefore it is estimated that the testing of a task towards the end of the process can become unwieldy with JMeter, it demands the identification and interpretation of many requests. Another important observation in this experiment is that this testing approach suffers from dependence on Web technologies used by BPMS.

4.2 Functional Tests

To perform functional tests on Web applications, you can use free tools like Selenium[9], Watir[10] or Geb[11].For this work, we picked up the Selenium tool, combined with Cucumber-JVM [12] for a description of the tests.

With these tools, the process for running a functional test consists of: capture the user interaction with the browser (Selenium IDE), export the generated code (Selenium IDE), creation of test scenario (Cucumber), creation of definitions of test steps (Cucumber), creation of methods for each step (Java) and test execution (Selenium WebDriver). The test scenario is a definition, in order of execution of steps that run the application, and the expected results. For this work, was set up a scenario in which the student logs in and fills 2 forms as a first task of the process. For this scenario, methods were created by varying the entries on the forms. As the tested process is the same in both BPMS, the test scenario is also the same.

For applications of both BPMS, initially errors occurred in the tests relating to the location fields in web forms. It was found that the fields were located in a *iframe*, while the capture of the interaction to occur without problems, the generated code did not select the *iframe* and so could not find the fields. Thus, it was necessary to use a Selenium method to access the *iframe* before selecting the desired element.

In application with Activiti also ran into other trouble. Unlike what happened with the Web forms generated by Bonita, the Selenium IDE did not capture all user interaction with the application. Indeed, in the login stage, Selenium just captured the access page and the " click " to the login button, ie not captured filling the fields " User " and " password ". This problem was repeated with some other elements of the Web form when recording the interaction. It is believed that the problem occurs due to the structure of the Web page, which can contain elements that Selenium does not automatically identify, such as *divs*, *frame* and *scripts*, for example. However, this does not preclude the creation and execution of tests. To work around the problem, it was necessary to study the structure of

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

the Web pages in question, locate the missing elements and then add the code to access them in their methods.

Resultados dos Testes Funcionais The functional tests proved to be more viable than load tests, by not requiring analysis interactions with the server. It can be said that this test reached all its goals because it allowed reproduce user interaction and create the code to test applications with different inputs in a scenario involving the initial task of the process. Os testes funcionais mostraram-se mais viveis do que os testes de carga, pois uma boa parcela da interao executada no lado cliente, sem necessidade de lidar explicitamente das interaes com o servidor. Pode-se dizer que este teste atingiu todos seus objetivos, pois permitiu reproduzir a interao do usurio, bem como criar o cdigo para testar as aplicaes com diferentes entradas, num cenrio envolvendo a tarefa inicial do processo, ampliando a cobertura dos testes. O teste funcional tambm reproduziu erros encontrados em produo e que tinham passado despercebidos nos testes manuais (entradas no previstas corretamente nos formulrios), esses erros foram os mesmos em ambas as ferramentas BPMS, pois os formulrios foram configurados de modo equivalente.

O teste funcional tambm foi menos dependente do BPMS. Na Tabela 2 apresenta-se um resumo das principais semelhanas e diferenas encontradas. Em ambos os casos, foram necessrias poucas modificaes no cdigo gerado pelo Selenium e Cucumber, bastando para isso inspecionar a estrutura das pginas Web. O Cucumber torna a implementao dos testes mais rpida e menos trabalhosa do que se fosse usado apenas o Selenium, abreviando a gerao de cdigo alinhado com os cenrios de teste. Mesmo assim, caso seja necessrio estender os testes a muitas tarefas de um processo, as modificaes no cdigo de teste podem se tornar trabalhosas.

	Bonita	Activiti
Tecnologias Web	HTML, CSS, Ajax	HTML, CSS, Ajax
Captura da interao do usurio utilizando o Selenium	Total	Parcial (necessitou de insero manual de alguns campos)
Foi possvel exportar o cdigo gerado pelo Selenium?	Sim	Sim
Reconhecimento de todos os campos capturados SEM alterao de cdigo	Parcial	Parcial
Reconhecimento de todos os campos capturados COM alterao de cdigo	Total	Total

Table 2. Resumo comparativo sobre o teste funcional

5 Consideraes Finais

Neste trabalho, explorou-se solues de teste automatizado em uma aplicao de BPMS. Na ausncia de suporte a testes de carga e funcionais nos BPMS Bonita e Activiti, aplicou-se ferramentas de teste voltadas a aplicaes Web em geral.

No que concerne ao teste de carga, este mostrou-se til para explicar falhas observadas no trabalho anterior. Tambm mostrou-se um teste trabalhoso, ou at invivel, dependendo do BPMS usado. A experincia com dois BPMS fortaleceu essa concluso, pois ocorreram situaes distintas: com Bonita o teste foi bem sucedido, porm com Activiti o teste no pde ser executado, j que no se conseguiu reproduzir todas requisies.

No teste funcional, com a abordagem adotada, obteve-se maior sucesso na execuo dos testes e observou-se uma menor dependncia dos BPMS, em comparao 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 negcio pode ter.

De modo geral, a experincia mostrou que, sob certas condies, vivel testar aplicaes de BPMS com ferramentas de teste para sistemas Web. A principal condio, no caso considerado, foi o direcionamento dos testes a uma tarefa inicial do processo, identificada como gargalo. Como o tempo e esforo para uma nica tarefa foi significativo, essa abordagem pode se tornar invivel caso seja necessrio aumentar a cobertura dos testes.

De modo geral, como lies aprendidas temos que, sob certas condies, vivel testar aplicaes de BPMS com ferramentas de teste para sistemas Web. A principal condio, no caso considerado, foi o direcionamento dos testes a uma tarefa inicial do processo, identificada como gargalo. Como o tempo e esforo para uma nica tarefa foi significativo, essa abordagem pode se tornar invivel caso seja necessrio estender os testes a muitas tarefas de um processo.

Outro aspecto a ser considerado que, quando o BPMS implementa a interao com o usurio e/ou com servidores, o desenvolvedor deixa de escolher e controlar todas as tecnologias utilizadas. Essa facilidade, no entanto, pode dificultar a automao de testes com ferramentas externas, que se beneficiam de conhecimento sobre a implementao (por exemplo, uso de *iframes*, Ajax, na experincia em questo).

Embora a experincia deste trabalho no tenha sido exaustiva, pde-se notar que o suporte a testes automatizados 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.

References

1. Guide to the Business Process Management Body of Knowledge (BPM CBOK). Association of Business Process Management Professionals, 2nd edition (2012)
2. Bourque, P. e Fairley, R. E.: Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0. IEEE Computer Society (2014)

3. Chetty, N. K.: How to perform workflow testing for BPM applications (2014), `http://www.evoketechnologies.com`
4. Chiavegatto, R. et al.: Especificao e automao colaborativas de testes utilizando a tcnica BDD. In *XII Simpsio Brasileiro de Qualidade de Software*, pp. 334–341, (2013)
5. de Moura, J. L. et al.: Gesto de processos de negcio em curso de sistemas de informao: Relato de experincia utilizando software livre. In *IX Simpsio Brasileiro de Sistemas de Informao*, pp. 206–217, (2013)
6. Forrester Research.: The forrester wave: BPM suites, Q1 (2013)
7. Graham, D. e Fewster, M.: Experiences of Test Automation: Case Studies of Software Test Automation. Addison-Wesley (2012)
8. Myers, G. J. e Sandler, C.: The Art of Software Testing. John Wiley & Sons, (2011)
9. van der Aalst, W. M. P.: Business process management: A comprehensive survey. *ISRN Software Engineering*, 2013(507984)
10. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, 2nd edition, (2012)
11. Winter Green Research.: Business process management (BPM) cloud, mobile, and patterns: Market shares, strategies, and forecasts, worldwide, 2013 to 2019.