Evaluating Current Testing Processes of Web-Portal Applications

Harpreet Bajwa, Wenliang Xiong & Frank Maurer
University of Calgary, Department of Computer Science
Calgary, Alberta, Canada T2N 1N4
+1 (403) 220-7140

{bajwa, xiongw,maurer}@cpsc.ucalgary.ca

Abstract: Web-portal application development needs to be improved by comprehensive testing processes and practices. Building the initial knowledge by evaluating and improving the state of the art in testing will steer the future for better tested portal solutions. In this paper, we present the results of an industrial case study that helped us 1) to understand the testing practices and tools in use and 2) to identify challenges in testing web portal applications.

1 Introduction

Java-based enterprise portal application [1] and JSR 168[1] based [2] portlet development is growing rapidly. Consequently, identifying practices that improve the testing of web portal applications become important in increasing the quality and reducing the time required for the development, deployment and test cycle of these applications. A prerequisite to providing support for better tested applications is an early assessment of the existing testing process and nature of challenges in the real world. The inspiration for the empirical study conducted by us came from this need. To our knowledge, no studies have been undertaken to evaluate how web portal applications are being tested in the real world. Our paper helps to fill this gap.

The paper reports on the results of the case study and makes two contributions. First, the results provide empirical evidence on the nature of existing challenges that impact comprehensive testing of web portal applications and the strategies developers use to cope with these challenges. Second, the results highlight requirements on tool support in areas where portal applications cannot be tested automatically.

2 Industrial Case Study Discussions and Results

We conducted an interpretive industrial case study [6] in collaboration with Sandbox Systems[2] using qualitative methods to explore how developers tested and engineered

[1] JSR 168 standardizes the interface between portlet container and the portlet.

[2] This work was supported under an industrial research grant provided by Sandbox systems.

portal artifacts. Data collection methods included administration of a questionnaire [8], interviews, notes we took and numerous discussions with the chief architect responsible for developing portlet-based e-business tools. We relied on the extensive professional experience of the chief architect at the company to provide deeper insight into the development and testing practices. To gain further understanding on the nature of tests the developer's were writing and running, we inspected an existing web-portal application built by the company.

The challenges in web-portal application testing brought to light are:

Challenge 1: Web portal applications are implemented using the Model-View-Controller (MVC) paradigm. MVC pattern testing for portal applications are automated at the model and controller layers, but it is difficult to test the view layer. Portlets are the chief building block of portal applications forming the view part of the MVC model. Portlet relies extensively on the services provided by the portlet container [1]. Therefore, testing portlets is difficult although important. As reported by the developer's portlets in an application work correctly in the test environment but errors are seen when the same portlets are deployed and executed in the portal server production environment [2]. This is a severe problem as no functionality is available to the end user since the portlet does not display any data. To debug, the portlet is disabled and the logs are examined and analyzed till an eventual fix is determined. This causes the developers a lot of time and effort when deploying the application in the production environment. Existing testing frameworks do not allow a direct fine grained testing of portlets. A testing approach is needed that permits executing the test code inside the container environment and the ability to access and control the environment specific objects. We call this in-container testing (ICT). Although ICT is slow and running tests frequently is infeasible, it might be needed for performing portlet application health checks when the application is deployed on the production environment. Our paper [7] presents a solution to this problem.

Challenge 2: Access to sensitive portlets and pages is controlled by assigning permissions to various user groups. Currently, the administrator setting the permissions must login as a user related to a role and test manually each time the applications are deployed to verify whether the permissions have been correctly assigned. Being able to switch roles quickly is a challenge. Most unit testing of portal applications are done with the current user id for the portal. A framework that allows in setting up a series of ids to use in "role-based" unit testing would be helpful. At this time, this testing needs to be done manually. Our above mentioned paper also addresses this issue.

Challenge 3: As reported by the developers testing frameworks such as HttpUnit [5] geared towards black box testing of web-applications are inconvenient for portal applications because html parsing is slow, there is a lack of detailed control over the environment and constructing an initial state for the HttpUnit tests is time consuming. Portlets unlike servlets are not bound to a single URL which imposes additional problems in using frameworks such as HttpUnit.

Challenge 4: The developers at the company would like to use test driven development (TDD), a practice of agile methods [3], for developing portal applications. TDD encourages that tests must be written first and allowed to fail before the functionality to pass the test is written and testing activities are closely tied in with application development. Unit testing portlets requires the portlet container environment and the

ability to access and control the environment specific objects. Consequently, unit tests developed using TDD cannot easily access the functionality in the portlet application code. In the chief architect's words "developing portlet code in a TDD way would be nice". Besides, small changes to the application code can be tested using the mock portlet API reducing the development, deployment and test cycle of portlet applications. Mocking the portlet API is one way of supporting TDD of portlets which is currently being explored by us.

3 Future Work

To gain further insight into the testing process, we are in the process of conducting a qualitative survey and interviews of independent portal developers. Our current and future efforts are directed towards developing a set of testing practices and tools that aim to meliorate the challenges in automated unit testing of portlets as well as web portal applications. WIT[3] Framework was developed by us to solve one of the testing challenges presented in section 2.

Acknowledgements: We convey our sincere gratitude to Sandbox Systems for their continuous inputs.

References

1. Portal-Introduction-IBM.http://www-106.ibm.com/developerworks/ibm/library/i-portletintro (Last Visited: February 7, 2005).
2. IBM Websphere Portal Zone http://www7b.software.ibm.com/wsdd/zones/portal/ (Last Visited: February 7, 2005).
3. Test Driven Development-Guide David Astels.;Test Driven Development-A practical Guide, ACM Press (2003).
4. JUnit testing Frameworks For Testing http://www.junit.org/index.htm, (Last Visited: February 7, 2005)
5. Client Side Testing using HttpUnit. http://httpunit.sourceforge.net/ (Last Visited: February 7,2005
6. Association For Information Systems, Qualitative research Micheal D Myers http://www.qual.auckland.ac.nz/ (Last Visited: February 7,2005).
7. Wenliang Xiong, Harpreet Bajwa , Frank Maurer: WIT: A Framework For In-Container Testing of Web-Portal Applications. Proc of ICWE 2005.
8. Portlet Testing-EBE. http://ebe.cpsc.ucalgary.ca/ebe/Wiki.jsp?page=Root.Portlettesting (Last Visited: April 15,2005).

[3] WIT-Web Portal In container testing Framework was implemented as a result of this study. Design, implementation and tool usage are reported in the paper published in ICWE 2005 [7].