A Lightweight Semi-automated Acceptance Test-Driven Development Approach for Web Applications

Diego Clerissi, Maurizio Leotta, Gianna Reggio, Filippo Ricca

Abstract:

Applying Acceptance Test Driven Development (ATDD) in the context of web applications is a difficult task due to the intricateness of existing tools/frameworks and, more in general, of the proposed approaches. In this work, we present a simple approach for developing web applications in ATDD mode, based on the usage of Screen Mockups and Selenium IDE.

Digital Object Identifier (DOI):

`http://dx.doi.org/10.1007/978-3-319-38791-8_55`

A Lightweight Semi-automated Acceptance Test-Driven Development Approach for Web Applications

Diego Clerissi, Maurizio Leotta, Gianna Reggio, Filippo Ricca

DIBRIS, Università di Genova, Italy
diego.clerissi@dibris.unige.it, {maurizio.leotta, gianna.reggio, filippo.ricca}@unige.it

Abstract. Applying Acceptance Test Driven Development (ATDD) in the context of web applications is a difficult task due to the intricateness of existing tools/frameworks and, more in general, of the proposed approaches. In this work, we present a simple approach for developing web applications in ATDD mode, based on the usage of Screen Mockups and Selenium IDE.

1 Introduction

The emerging relevance of web-based software in everyday human activities arises the need for effective development approaches helping developers in the realization of high quality products. It is well-known that web applications are prone to frequent changes due to customers' requests and requirements evolution. In this context, agile approaches are considered appropriate for web application development.

Acceptance Test Driven Development [2] (from now, ATDD) is a cornerstone practice that puts acceptance testing on top of the software development process and focuses on the communication among customers and software professionals, such as business analysts and software testers.

Even if ATDD is often tool assisted [1], its usage in the context of web applications is still a problematic task, since the available web testing tools often require the presence of the application itself and the knowledge of technical details related to the used programming languages and frameworks. As a result, a non-trivial manual intervention to convert acceptance test cases into executable test scripts is required (i.e. it is very difficult to define web test scripts before implementing the web application).

In this work, we propose a lightweight semi-automatic approach that, starting from textual requirements and screen mockups, is able to generate executable Selenium IDE[1] functional test scripts (i.e. black box tests able to validate a web application by testing its functionalities), which in turn drive the development of web applications.

2 ATDD in the context of Web Applications

ATDD is an agile development practice adopting the strategy of TDD, where test scripts are defined before software implementation and used as acceptance criteria for validating it through business objectives. However, ATDD is not limited to agile contexts where

[1] http://www.seleniumhq.org/projects/ide/

requirements can be expressed with user stories (i.e. descriptions consisting of one or more short sentences). Indeed, even more formal requirements specifications based on well-structured use cases [8, 9] can be used with the ATDD practice. Usually, in these cases, acceptance tests are defined by following the scenarios composing the use cases.

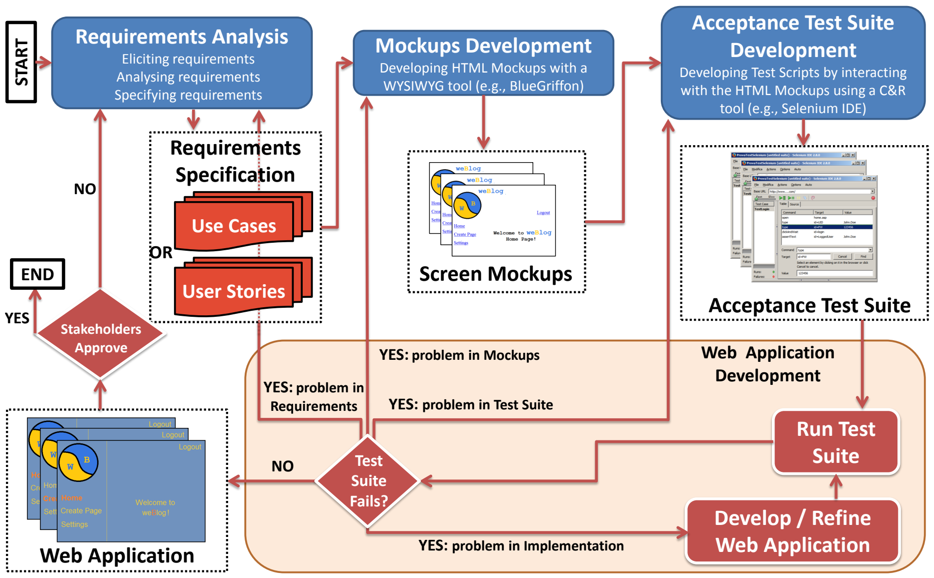
A large number of ATDD tools usable in the web applications context emerged in the last years. For instance, Fitnium[2], an integration of FitNesse[3] (where test cases can be represented in a tabular form by using the natural language), and Selenium IDE (a Firefox plug-in that allows to record, edit, and execute web test scripts). Some other tools require a different template for acceptance test cases (like, e.g., the "given-when-then" template of Cucumber[4]) or a freely HTML format which is later enriched by tags to interpret the text and execute it (as for Concordion[5]). Unfortunately, in all these tools, the connections between test cases and web application under development (i.e. the so-called fixtures) have to be written by the developer. Besson et al. [1] are among the first researchers to describe an ATDD approach for web applications trying to overcome this burdensome activity. In that work, a web application is modelled with a graph of pages and the paths of the given graph are the test cases, which must be validated by the customer and subsequently transformed into test scripts. Conversely to Besson et al., our approach does not require the web application modelling phase, which is substituted by a simpler recording phase of user actions by means of Selenium IDE executed upon the previously produced screen mockups.

3 The Approach

The approach we propose in this work is based on the following tasks (see Fig. 1):
Requirements Analysis aims at producing the requirements specification for the web application under development. It includes the following subtasks: (1) gathering the requirements from future users, customers and other stakeholders, (2) determining whether the requirements are complete, consistent, and unambiguous, (3) writing the requirements specification as use cases or user stories depending on whether, respectively, a more prescriptive or more agile development approach is adopted.
Mockups Development aims at creating a set of screen mockups used for prototyping the user interface of the web application to develop [3, 7]. In order to reduce as much as possible the need of manual intervention required to run the automated acceptance test scripts on the web application under development, the mockups have to represent quite accurately - from a functional point of view - the interfaces of the web application (e.g., all the web elements of the web pages to interact with must be shown in the mockups while the layout or the styles can be just sketched). To produce the mockups, it is possible to follow the method described by Reggio et al. [8] helping in capturing and writing requirements specifications enriched with screen mockups. Since we chose to adopt the state of the practice web testing tool Selenium IDE, mockups can be quickly developed in HTML using a WYSIWYG content editor for web pages such as BlueGriffon[6]. Adopting a capture-replay DOM-based [5] tool like Selenium IDE allows to pay little attention to the mockups graphical aspect and focus on the user interaction, with clear advantages

[2] https://fitnium.wordpress.com/ [3] http://www.fitnesse.org/ [4] http://cucumber.io/ [5] http://concordion.org/
[6] http://www.bluegriffon.org/

Fig. 1. From the Requirements Analysis to the Web Application Development


START
Requirements Analysis
Eliciting requirements
Analysing requirements
Specifying requirements
Mockups Development
Developing HTML Mockups with a WYSIWYG tool (e.g., BlueGriffon)
Acceptance Test Suite Development
Developing Test Scripts by interacting with the HTML Mockups using a C&R tool (e.g., Selenium IDE)
Requirements Specification
Use Cases
OR
User Stories
Screen Mockups
Acceptance Test Suite
NO
END
YES
Stakeholders Approve
YES: problem in Mockups
Web Application Development
YES: problem in Requirements
YES: problem in Test Suite
Run Test Suite
NO
Test Suite Fails?
Develop / Refine Web Application
YES: problem in Implementation
Web Application

in terms of effort required for creating the test scripts [4]; for this reason we avoid to adopt both visual or programmable [5] web testing tools, such as Sikuli[7] or Selenium WebDriver[8] respectively.

Acceptance Test Suite Development. Once the mockups are available it is possible to record the test suite by interacting with them. To make this task easier we suggest to implement the links among the web pages and the submission buttons. Concerning submission buttons, it is possible to hard-code the alternative links using JavaScript; for instance, when dealing with a login form we can reach two mockups, "homePage.html" and "wrongPage.html", depending on the inserted values. In this way it is possible to record the test suite as if it were a real web application. More in detail, it is necessary to: (1) open the first produced HTML mockup with the browser and activate the Selenium IDE recording functionality, (2) follow the steps described in the use cases / user stories and replicate them on the HTML mockups (e.g., insert values in the input fields, click links), and finally (3) manually insert the assertions in the generated test scripts. Selenium IDE generates the locators for the web elements to interact with using different strategies [6] and, when possible, relies on id, linkText or name values that can be easily specified, for each web element, using BlueGriffon.

Web Application Development is based on a test-first approach using the previously produced test scripts. The functionalities are implemented/refined following the test suite as a guidance until all tests pass successfully. Finally, stakeholders evaluate the resulting web application and decide whether approving it or moving through a further refinement of artifacts. Test scripts are ordered starting from the ones concerning the simpler functionalities (e.g., login) and grouped by use case/user story. It is important to notice that the web application development can be conducted with any technology – e.g.,

[7] http://www.sikuli.org/ [8] http://www.seleniumhq.org/projects/webdriver/

Ajax, Flash, etc.– and any development process – e.g., traditional, model-driven (e.g., using WebRatio) or by means of mashups. The only constraint is to use the same text (e.g., for linkText locators) or id/name attribute values used in the produced mockups. Moreover, developing the mockups and defining the links among them allows also to produce a preliminary but "working" prototype of the web application that can be shown to the stakeholders. This is very useful for detecting, as soon as possible, problems and misunderstandings in the requirements [10].

4 Conclusion and Future Work

In this work, we have proposed a novel approach for developing web applications adopting the ATDD practice. The novelty regards the usage of screen mockups for generating, with a limited effort, acceptance test scripts able to drive the development of the target web application. The approach has been successfully applied on a sample case study (weBlog application) to show its feasibility. As future work, we intend to validate and refine our approach by means of experiments and with real industrial case studies. In this way, we will be able to measure the additional costs required for generating the mockups and gather feedbacks on the effectiveness and usefulness of our approach.

References

1. F. M. Besson, D. M. Beder, and M. L. Chaim. An automated approach for acceptance web test case modeling and executing. In *Proceedings of 11th International Conference on Agile Software Development (XP 2010)*, volume 48 of *LNBIP*, pages 160–165. Springer, 2010.
2. G. Downs. Lean-agile acceptance test-driven development: Better software through collaboration by Ken Pugh. *ACM SIGSOFT Software Engineering Notes*, 36(4):34–34, 2011.
3. H. R. Hartson and E. C. Smith. Rapid prototyping in human-computer interface development. *Interacting with Computers*, 3(1):51–91, 1991.
4. M. Leotta, D. Clerissi, F. Ricca, and P. Tonella. Capture-replay vs. programmable web testing: An empirical assessment during test case evolution. In *Proceedings of 20th Working Conference on Reverse Engineering (WCRE 2013)*, pages 272–281. IEEE, 2013.
5. M. Leotta, D. Clerissi, F. Ricca, and P. Tonella. Approaches and tools for automated end-to-end web testing. *Advances in Computers*, 101:193–237, 2016.
6. M. Leotta, A. Stocco, F. Ricca, and P. Tonella. ROBULA+: An algorithm for generating robust XPath locators for web testing. *Journal of Software: Evolution and Process*, 28(3):177–204, 2016.
7. M. O'Docherty. *Object-Oriented Analysis and Design: Understanding System Development with UML 2.0*. Wiley, 1 edition, June 2005.
8. G. Reggio, M. Leotta, and F. Ricca. A method for requirements capture and specification based on disciplined use cases and screen mockups. In *Proceedings of 16th International Conference on Product-Focused Software Process Improvement (PROFES 2015)*, volume 9459 of *LNCS*, pages 105–113. Springer, 2015.
9. G. Reggio, F. Ricca, and M. Leotta. Improving the quality and the comprehension of requirements: Disciplined use cases and mockups. In *Proceedings of 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2014)*, pages 262–266. IEEE, 2014.
10. F. Ricca, G. Scanniello, M. Torchiano, G. Reggio, and E. Astesiano. Assessing the effect of screen mockups on the comprehension of functional requirements. *ACM Transactions on Software Engineering and Methodology*, 24(1):1–38, 2014.