Regression and Performance Testing of an e-learning Web Application: dotLRN

Ana Cavalli, Stephane Maag, Gerardo Morales
Institut National des Télécommunications INT - CNRS SAMOVAR,
9 rue Charles Fourrier, 91000 Evry, France

Abstract

Along with the continuous development of big and complex Web applications such as dotLRN [8], it is mandatory to execute in a scheduled way a set of test cases to assure its functional stability and to make sure that the web application still runs whatever the modifications applied to the implementation. This article highlights the importance of testing and presents the types of test that are needed to assure not just the stability but also the scalability of the dotLRN platform by testing some non-functional aspects. This article presents different methods that can be applied to test in general any Web application. Then a methodology to manually develop the test conformance in order to automatically test the regression of the implementation is proposed. Finally this paper presents our proposal to cover the performance and load tests of the dotLRN web application.

1. Introduction

Nowadays, different organizations are increasing the use of Web applications to overcome geographical barriers and gain other advantages such as flexibility in using tools for synchronous and asynchronous collaboration between users [6]. Among the different organization types, educative entities have shown a major increase in the use of e-learning platforms to support courses. In this same context, in the last few years major progress has been achieved in the design of e-learning methodologies and platforms, particularly developing tools and methods to facilitate users to access these technologies. More recently, this evolution has strengthened the idea to develop open software platforms that cover the general needs and can be customized by in-house development to answer the specific needs of each educational institution.

Along with this evolution, the sophistication and complexity of these platforms are growing. Therefore, it is not enough to perform superficial testing of usability by monitoring human-machine interaction through the user interface to cover the system functionality. To test in an exhaustive way a complex system such as an e-learning platform it is necessary to adopt well known approaches of testing, for example those used in the field of telecommunications.

In the context of the European @lis project E-LANE [1], it was needed to test the conformance of dotLRN with its model, but because of the lack of formal model, the conformance test cases where manually developed and executed. By using the conformance testing, it is possible to detect bugs, but the real problem is that dotLRN as all the open source software is constantly changing with the addition of new features. Therefore, new bugs may appear disabling some functionalities. Then it is sometimes needed to manually re-execute all the test cases, to re-test all the features in a scheduled way to guarantee the stability of the software. This testing approach is called regression testing. This paper explains how by executing manual conformance testing it is possible to re-use this test cases to perform automatic regression testing and present some of the existent tools to achieve this objective.

Among the guarantees that testing should give, is that a user cannot gain permissions over the system if he is not supposed to have them, this assures the basic security rules. Another guarantee should be that the system does what it is supposed to do, to assure its usability. It is also needed to take into account that modern Web applications are becoming increasingly complex and mission critical. This situation becomes even worse because of the lack of standardization in Web clients (browsers). Testing has to assert the system usability.

By using the regression test approach, it is easier to maintain the stability of the functional aspects of this e-learning platform during its constant development. However some of the important issues to guarantee the good deployment of dotLRN are non-functional aspects such as the performance of the system or the way that the system can manage big load tasks. Therefore, this article proposes also
cases of a Web application, and select just the test cases that
slicing to avoid the re-execution of all the regression test
developing speed and changeable user demands.
the open source Web applications because of its continuous
correct [9]. This testing technique is important to apply to
tested code, and provide confidence that modifications are
whether new errors have been introduced into previously
ing is the process of validating modified software to detect
the tools used to build the regression test cases, the ones that
achieve an automatic regression testing on Web applica-
tion during a certain time, or a well defined number
in which the trace of interactions (i.e the record) is written
are the most popular. Selenium [13], for example is a tool
can record the interaction between the user and the Web ap-
applications that do not have a formal specification. Among all
work in which the authors survey several analysis modeling
methods used in Website verification and testing. In [11], a
methodology of Model-Driven Testing for Web application
is presented.
Moreover, concerning the testing needed to help main-
taining the stability of Web applications, numerous works
are using the regression testing approach. Regression test-
ing is the process of validating modified software to detect
whether new errors have been introduced into previously
tested code, and provide confidence that modifications are
correct [9]. This testing technique is important to apply to
the open source Web applications because of its continuous
developing speed and changeable user demands.
The work presented in [18] proposes a method based on
Slicing to avoid the re-execution of all the regression test
cases of a Web application, and select just the test cases that
will interact with the part of the Web application that have
suffered a change with the insertion of the new code.
Several tools and methodologies have been developed
to achieve an automatic regression testing on Web applica-
tions that do not have a formal specification. Among all
the tools used to build the regression test cases, the ones that
can record the interaction between the user and the Web ap-
dication during a certain time, or a well defined number of
interactions (e.g. to follow a link or to submit a form) are the most popular. Selenium [13], for example is a tool
in which the trace of interactions (i.e the record) is written
as HTML tables. However, in the E-Lane project it was
needed to obtain a trace flexible enough to be changed and
re-used in easy and fast way, for example a trace written in
a scripting language.
Tclwebtest [17] is a tool to write tests for Web applica-
tions. It provides an API for issuing HTTP requests and pro-
cessing results. It assumes specific response values, while
taking care of the details such as redirects and cookies. It
has the basic HTML parsing functionality, to provide access
to elements of the resulting HTML page that are needed for
testing, mainly links and forms.
The execution of a test case written in tclwebtest will
simulate a user that is interacting with the Web applica-
tion through a Web browser. Using the links and forms it
is possible to add, edit or delete data of the Web applica-
tion by executing the test case script. Figure 1 illustrates
the tclwebtest code for logging in into the dotLRN platform
by requesting the register page, then filling the e-mail and
password, and submitting.

```
::twt::do_request "http://dotlrn.org/register"
tclwebtest::form find -n login
tclwebtest::field fill "user@mymail.com"
tclwebtest::field find ~n password
tclwebtest::field fill "mypassword"
tclwebtest::form submit
```

Figure 1. Example of tclwebtest code.

We can also mention some web application tools (e.g.
OpenSTA [14], SOASTA [3], WAPT [16]). The listing of
load/performance test tools and Web functional/regressio

test tools is presented in [10].

3. dotLRN Platform and OpenACS Framework

DotLRN [8] is an open source platform for supporting
e-learning. This tool was originally developed at the Mas-
sachusetts Institute of Technology as a virtual learning en-
vironment, and it evolved into a comprehensive platform
including not only e-learning support but also generic web
resources. DotLRN uses as base the OpenACS [12], a framework
for building scalable, community-oriented web applica-
tions. The framework structure is modular, in fact, dotLRN
is a set of modules that provide the additional features to
deploy an e-learning environment. The OpenACS (and
therefore dotLRN) is tightly integrated with a relational
database, both PostgreSQL and Oracle are currently sup-
ported. The Web server that handles requests at the basic
level is AOLServer, the America Online’s open source Web
server. One of its main features is the integration in its core
of a multi-threaded TCL interpreter which provides an ef-
efective solution for industrial strength type of services such
as those present in large higher educational institutions [7].

4. dotLRN Conformance Testing

As before mentioned, the regression test aims to verify if
after the insertion of new source code into the application,
all the functionalities still run correctly. Hence the first task
for testing dotLRN is to test if the behavior of this e-learning platform is conform to the blue-prints used to build it (i.e. its specification).

Conformance testing determines whether a system implementation meets some specified standards. These standards are in the description of the system that was used to build it. Further, the original specification of the system may be used in order to observe if the system meets the standards. The figure 2 illustrates the goal of the conformance testing.

Conformance testing consists in making the implementation under test (IUT) to interact with its environment. This environment is simulated by a tester (see Fig. 3) that executes the test cases and stimulates the IUT. The interfaces of the tester are called Points of Control and Observation (PCO).

Figure 2. Conformance testing scheme.

Because of the lack of formal specification, it was used the documentation of dotLRN and our expertise in OpenACS/dotLRN to build an informal specification (i.e. the list of requirements and standards that dotLRN must meet). By using this specification it is possible to interact with the platform and observe if it meets the standards or not. Now, to achieve the stability of the platform during its continuous development we have the same problematic presented in [9] the simplest testing strategy, re-test all, however, this requires an unacceptable amount of time and resources. It is needed then to automatize the process to re-test the platform to improve the testing efficiency.

In the following section, the methodology to solve this problem and the used tools to test the dotLRN platform in the context of the E-Lane project are introduced.

5. Methodology for the Development of the Regression Test Cases

In order to test a Web application that does not have a formal specification, test cases may be manually developed. In this section the methodology followed to develop these test cases for dotLRN using the tclwebtest recorder tool is illustrated. It will also introduce the acs-automated-testing, an OpenACS [12] package for the management of test cases execution and the verdict storage of each test case.

Although the phase of the conformance testing was a hand made process, by using the TwtR plugin (see section 7.2) it is possible to obtain a record or static trace of the interaction of the user with the Web application, this static trace is the base for the development of the regression test cases. The process of obtaining a static trace is illustrated in the figure 5.

The TwtR [4] is a recent tool based in the interaction recording as Selenium, by using this tool it is possible to obtain the trace in a language based in TCL script language, more specifically in tclwebtest code. TwtR will produce a static trace that does not contain all the interactions between the user and the Web application, but only the stimulations of the user to the Web application. This static trace is the basis for the part of the functional testing process presented in this article.

Figure 5. Generation of the static traces used as a base for further regression testing.

To build the tests for the dotLRN features, it is first extracted information from the OpenACS documentation and from dotLRN platform usability to build its specification. After this, we interact with dotLRN to test its conformance while recording the static trace using TwtR. If the result of this conformance test execution verdict is positive, the static trace can be re-used to serve as a basis to build the regression test case. This static trace must be modified by inserting variables instead of hard coded values (e.g. the id’s and the URL’s). Each test case will be finally described in tclwebtest and composed by a set of concatenated scripts. The generation of each test case using tclwebtest was done by following these steps:

1. Create the script of the test preamble, i.e. a sequence of operations that will lead the system to a state where the test case can be executed. E.g. for testing the edition of the user’s name, first it is needed to corroborate the
The load test consists in constantly increasing the load on the system via automated tools. For a Web application it consists in increasing the number of HTTP requests and/or the size of the database. The goal of this test is to expose bugs that do not surface in cursory testing, e.g. memory management bugs, memory leaks, buffer overflows, etc. The performance test is a combination between the load test and the collection of information about the resources. Hence, monitoring server hardware and resources is crucial. To obtain the results of this test, a carefully controlled process of measurement and analysis is required. The main goals are to detect bottlenecks at the application, database, operating system and network level.

### 6.1 Performance test tools

Nowadays many efforts in the research field are proposing new methodologies to generate exhaustive performance testing by using model-based testing such as the work presented in [15]. However, the tools to facilitate the development of test cases are processed in a semi-automated way. Among these tools, the OpenSTA [14] has been selected in the E-lane project to generate the load test cases for dotLRN.

### 6.2 Methodology

To find performance bottlenecks it is needed to put the server into a big load and in parallel to supervise the different levels behavior. In a Web application these different levels are: database, OS, application and network level.

The methodology consists in:

1. Generate traces for each task that will be executed during the performance test. These traces can be considered as static scenarios, because the variable values are already assigned;
2. Add to this static scenario variables and procedures to obtain a dynamic script able to fully interact with dotLRN;
3. Create agents that will supervise the server hardware while the execution of the test. At least it is needed to create agents to do monitoring over (1) the CPU and (2) the used memory of the server;
4. Fix the number of virtual users that will simultaneously use the server;
5. Fix the tasks that every virtual user will execute on dotLRN;
6. Run the test and in parallel (1) observe the behavior of the hardware and (2) obtain the time of response of the HTTP requests;
7. Analyze the crossed data of the number of virtual users vs. the behavior of the server to obtain the area of the instance where the bottleneck is located.

We illustrate the deployment and application of this methodology to a real case study in the following.

7. Implementation and Evaluation

7.1 Configuration of the Web server

Our laboratory have configured a server of a dotLRN instance at http://e-lane.int-evry.fr to apply the methodologies presented in this paper to a real world implementation. This server was configured over a Pentium 4 with 2 gigas of RAM using Debian as an operative system. DotLRN 2.3 needs to use as base software the AOLserver 4.5, the OpenACS 5.3, and Portgresql database.

7.2 Regression test cases generation and execution

Even if tclwebtest is very intuitive, hand development of test cases for an exhaustive platform test requires a big amount of time and effort. To reduce this needed time, a new tool has been recently developed, a plug-in called Tcwtest-Recorder (Twr) [4] for Mozilla Firefox. Using the Twr a trace written in tclwebtest of all the interactions that a user had with the Web application is provided. Then, test cases may be coded by adding variables and applying some changes to the code of this trace.

Among all the test cases provided by the E-lane project, this paper presents the generation for the addition of a Question and Answer from the FAQ package of dotLRN.

The trace produced during the addition of a Question and Answer (Q&A) of the FAQ is illustrated in the figure 4. In this trace it can be observed that the elements such as the FAQ name and the Q&A is hard coded. In the first line it can be viewed that the user has followed a link named Test faq which is the name of the FAQ to be tested. After this in the third line the user followed the link to arrive to the page to create a new Q&A. Then from the lines four to seven the form was filled with the question What is your name? and the answer Harry.

By changing all the texts to be inserted in the HTML forms, and text of the links to be followed need to be replaced by variables, the static trace is transformed into a dynamic script that will be the part of the test case for testing the addition of a Q&A. We have to notice that this script will be the part of the test case that will interact with the test Adding a Q&A, but it will also be re-used and served as the preamble of other test cases, for example to the test Edit a Q&A. Remember that before editing an item, it is mandatory that this item exists.

It is important to notice that the chunk of code of the figure 6 will be just the part of the regression test case that will interact with the part of dotLRN that allows to add a new Q&A. The entire test case must include the preamble (log in of the user, creation of the FAQ, assign a value to the variables to be used, etc.) and the part of the test case that will analyze if the Q&A was correctly created and assign the verdict. The entire test case is available at the annex of this article.

Figure 6. Chunk of the test case that was extracted from the Twtr trace illustrated in the figure 4.

7.3 Regression test execution

The OpenACS framework has among its packages the acs-automated-testing, this package allows to execute the test cases and to store the value of the verdict.

For each tested package, say package-xx, it has been created a sub directory test in the path /package-xx/tcl/. Within such test subdirectory there are two files:

- tclwebtest-procs.tcl: This file contains the definition of the functions used in the test case.
- package-xx-test-procs.tcl: This file contains the test cases, where the logical architecture of the test is defined. It is in this file where the functions defined in the tclwebtest-procs.tcl are called.

7.4 Performance test cases generation and execution

The generation of the test cases is possible by following the methodology described in the subsection 6.2, using
the OpenSTA tool and by establishing a communication between the tester (OpenSTA) and the IUT (dotLRN) by using the SNMP protocol.

1. Generation of the static scenarios. OpenSTA allows recording the interactions between a user and a Web application. As a result it gives a trace describing the communications between the server and the user written in SCL script language. For each task (e.g. following a link) OpenSTA can generate a trace which can be considered as a static scenario to execute this task.

2. Adaptation of the scenarios to dotLRN. The re-execution of a static scenario should not have problems if its task is just following some links to arrive to a certain page. On the other hand, if the task uploads a new document to a Web application, the scenario cannot be exactly re-executed with success. Since the file-id of the document uploaded during the recording of the trace must be unique, it is impossible to use the same file-id while uploading during a re-execution. Because of this, it is mandatory to replace the hard-coded id numbers by variables in the SCL code.

3. Creation of the collectors. The information that will be used to measure the system performance is gathered during the test cases execution. There are two types of information: the first one is the time of response of the server to an HTTP request, and the second one is the information about the hardware behaviour of the server. The first one is collected using the same protocol (HTTP) used to load the system and the second one is collected by the collectors i.e. requests of the tester through the port 161 using the SNMP protocol.

The collectors are periodically sending requests to the server and receiving back the information about the hardware status. For example a collector can be periodically sending requests to obtain the percentage of CPU used by the server.

4. Parameters configuration of the test cases and its execution. It is possible to follow the steps 4, 5 and 6 of the methodology presented in the section 6 through the user interface of OpenSTA. By performing steps 4 and 5, the load that will be applied to the Web application is fixed. In the figure 7 it is illustrated the communications during the execution of the tests (step 6), the load will be applied using HTTP requests, then the time of response will be received via HTTP, and the monitoring during the test execution via SNMP.

5. Finding the bottlenecks. By using the information of the time response for the HTTP requests, behavior of the CPU, management of the memory and the percentage of CPU used by the database; it is possible to find and establish the performance limits of the system. Using a tool like OpenSTA or The Grinder [2], the information can be displayed as in the figure 10. For example using this graphic, HTTP responses vs. Elapsed time it can be easily observed

![Figure 7. Scheme of the communication during the performance test.](image)

the picks of traffic responses, by using this information with the behavior data of the CPU and the memory management, it is possible to know if the bottleneck is in the OS level, database level or the application level. If no bottleneck was found in the past 3 levels the bottleneck will be at the network level.

7.5 Results

By applying the methodologies presented to the real use case, we have generated the regression and performance test cases. All the regression test cases are within the source code of dotLRN, so the aca-automated-testing package detects the files containing the test cases and generates a user interface to perform its execution.

In the context of the E-lane project, the main functionalities of the packages included by default in dotLRN platform are tested. In total more than 250 test cases where developed to test the packages: forum, faq, dotlrn, calendar, evaluation and file storage. As an example, the table of the figure 8 lists the necessary regression test cases for the evaluation package.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_faq</td>
<td>Testing the creation of a FAQ</td>
</tr>
<tr>
<td>delete_faq</td>
<td>Deletes an existent FAQ</td>
</tr>
<tr>
<td>edit_one_faq</td>
<td>Edits the FAQ name and description</td>
</tr>
<tr>
<td>enable_faq</td>
<td>Enables an existent FAQ</td>
</tr>
<tr>
<td>new_Q_A_faq</td>
<td>Creates a new QA of a new FAQ</td>
</tr>
<tr>
<td>delete_Q_A_faq</td>
<td>Deletes a QA of a FAQ</td>
</tr>
<tr>
<td>enable_faq</td>
<td>Enables an existent FAQ</td>
</tr>
<tr>
<td>new_Q_A_faq</td>
<td>Creates a new QA of a new FAQ</td>
</tr>
<tr>
<td>delete_Q_A_faq</td>
<td>Deletes a QA of a FAQ</td>
</tr>
<tr>
<td>insert_after_Q_A_faq</td>
<td>Creates a new QA of a QA with existent QA's</td>
</tr>
<tr>
<td>preview_Q_A_faq</td>
<td>Displays an existing question in a FAQ</td>
</tr>
<tr>
<td>swap_with_next_Q_A_faq</td>
<td>Swaps an existing QA with a QA of a FAQ</td>
</tr>
</tbody>
</table>

![Figure 8. List of test cases needed to test the basic features of the FAQ package.](image)

Since the development of the test cases they are part of the dotLRN source, they are daily executed to uncover regression bugs and maintain the stability of the main dotLRN functionalities along its continuous development. In the figure 9 it is illustrated the table displayed by the aca-automated-testing after executing a test case, it contains the
log messages that were written by the `aa_log` (the last line of the figure 6) during the execution of the test case and finally the verdict of the test.

![Figure 9. Results of the execution of a test case using the acs-automated-testing.](image)

As the `acs-automated-testing` is used as the tester and gives the user interface for the regression test cases, the OpenSTA take these roles for the execution of the performance test cases. It is important to notice that the obtained performance test results is the information about the behavior of the different server elements (memory, CPU, etc.) and time of response while it is submitted to pressure. By performing the tests is possible to discover the load limit of the server. By analyzing the results it is also possible to track if the performance limit is set by the database, application, OS or network level. But all these details can not be generalized to all dotLRN instances because it depends from factors such as the speed of the machine, the quality of the network, etc.

As an example, it can be observed in the figure 10 the graphic describing the time of response vs. the elapsed time while the execution of a load test case. This test case was simulating the login of users to dotLRN, the number of parallel users performing this action was incremental from the minute 0 to the minute 3 from 1 to 400. This figure illustrates how from in the minute 2:00 to the minute 3:28 the system slows down in responding to the HTTP requests of the incremental number of users.

![Figure 10. Resultant graph of the elapsed time vs the response time of the server.](image)

8. Conclusion

This article has presented the methodologies and results of a dotLRN platform testing based on testing theory for Web applications. It was presented a new approach to test the functionality of an e-learning, web based system. These tests are based on use cases and cover all relevant aspects of the system behavior.

Also it was illustrated a methodology to manually write test cases in `tclwebtest`. These test cases cover the main functionalities of dotLRN platform and are now fully integrated within the OpenACS code. They can be executed using the `acs-automated-testing` package to assure the conformance of the platform with the respect of its requirements and can be used as basis for future regression testing.

The methodology to generate conformance testing for the dotLRN platform where applied and the resultant test cases are now part of the official dotLRN code. This contribution helps to maintain the platform stability and to avoid breaking existent dotLRN features when adding new source code.

Finally a methodology to develop performance testing to evaluate the system scalability is illustrated. The methodology has been applied using a well adapted tool called OpenSTA on different servers and obtained information to discover the performance limits of the system.

We can highlight that the presented methods and the software tools proposed have been applied to dotLRN platform but they are generic enough and can be applied to other e-learning and Web based systems.

References

APPENDIX

The following is the entire test case for testing the creation of a new Question and Answer in the dotLRN FAQ.

```
aa_register_case -cats {web} -libraries tclwebtest//
tclwebtest_new_Q_A_faq 

A simple test case to faq package : Create a new Q&A.

} {
  aa_run_with_teardown -test_code {

    tclwebtest::cookies clear
    # Login user
    array set user_info [twt::user::create //
      -admin]
    twt::user::login $user_info(email) //
    $user_info(password)

    # Create a new Faq
    set faq_name [ad_generate_random_string]
    faq::twt::new $faq_name

    # Create a new Question_Answer
    set question [ad_generate_random_string]
    set answer [ad_generate_random_string]
    set response [faq::twt::new_Q_A $faq_name //
      $question $answer]
    aa_display_result -response $response //
    -explanation {Webtest for creating a New //
      Question in a Faq}

    twt::user::logout
  }

  set response 0
  db_lrow faq_id "select faq_id from faqs //
    where faq_name=:faq_name"

  # The Faq page url
  set faq_page_url [aa_get_first_url //
    -package_key faq]
  :twt::do_request $faq_page_url
  tclwebtest::link follow "administer"
  tclwebtest::link follow $faq_name
  tclwebtest::link follow "Create //
    New Q&A"

  tclwebtest::form find -n //
    "new_quest_answ"
  tclwebtest::field find -n "question"
  tclwebtest::field find -n "answer"
  tclwebtest::field fill "$question"
  tclwebtest::field fill "$answer"
  tclwebtest::form submit
  aa_log " Faq Question Form submitted"

  set_response_url [tclwebtest::response //
    url]
  if { [string match "* admin/one-faq * " //
      $set_response_url] } {
    if { [catch {tclwebtest::assert text //
        "$question"} errmsg] } {
      aa_error "faq::twt::new_Q_A : //
        failed $errmsg : Don't Create New //
        Question"}
    } else {
      aa_log "New Faq Question Created"
      set response 1
    } else {
      aa_error "faq::twt::new_Q_A failed. //
        Bad response url : $set_response_url"
    }
  }

  return $response
```