Another experience with Test* in industry: automated localisation testing

Mireilla Martinez, Anna I. Esparcia, Urko Rueda, Tanja E.J. Vos, and Carlos Ortega

Universidad Politecnica de Valencia
Camino de vera s/n, Valencia, Spain
{urueda,mimarmu,aesparcia,tvos}@pros.upv.es
Open Universiteit
Valkerburgerweg 177, Heerlen, The Netherlands
tanja.vos@ou.nl
Indenova
Carrer Dels Traginers 14, Valencia, Spain
cortega@indenova.com
http://www.testar.org

Abstract. Test* is a testing tool that automatically and dynamically generates, executes and verifies test sequences based on a tree model that is derived from the User Interface through the Accessibility API. Test* is an academic prototype that we continuously try to transfer to companies to get feedback about its applicability. In this paper we report on one of these short experiences of using Test* in industry at the Valencian company Indenova.

Keywords: Automated testing, technology transfer, TESTAR

1 Introduction

In previous work we have presented an approach to automated GUI testing called Test*[1] (Test Automation at the user inteRface level). Test* automatically and dynamically generates, executes and verifies test sequences based on a tree model (derived from the UI through the Accessibility API). No test cases are recorded and the tree model is dynamically inferred for every state[2], this implies that tests will run even when the GUI changes. This reduces the maintenance problem that threatens other GUI testing techniques like Capture and Replay [2] or Visual testing [1].

The Test* tool has been developed in the context of the EU FITTEST project that finished in 2014. First, it was evaluated in experimental conditions using different real and complex software applications like MS Office suite (running it 48 hours we detected 14 crash sequences). Subsequently, and with the purpose of getting a better understanding about the applicability of the tool in an industrial environment, we continuously try to apply Test* in companies to get

[1] http://www.testar.org
[2] The Graphical User Interface at a particular time

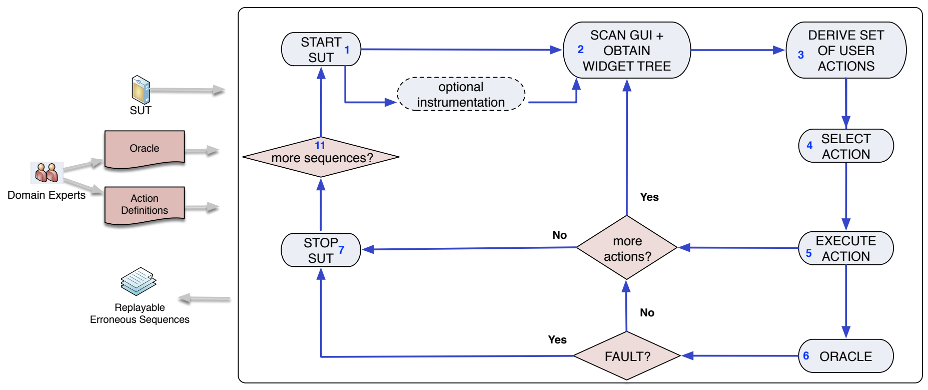

Fig. 1. *Test* testing flow*

feedback about its applicability and help companies to obtain solutions to the problems they face. In [3] results are described of transferring and evaluating the tool within 3 different companies on 2 desktop applications and one web application. In this paper we report on yet another short experience of using Test* in industry at the Valencian company Indenova[3]. This experience was done as a *micro initiative* within the SHIP project and the Software Testing Innovation Alliance[4].

2 Test*

Test* performs the steps as is shown in Fig. 1: (1) start the SUT (System Under Test); (2) obtain the GUI's *State* (a widget tree[5]); (3) derive a set of sensible actions that a user could execute in a specific SUT's state (i.e. clicks, text inputs, mouse gestures); (4) select one of these actions (random or using some search-based optimization criteria); (5) execute the selected action; (6) apply the available oracles to check (in)validness of the new UI state. If a fault is found, stop the SUT (7) and save a re-playable sequence of the test that found the fault. If not, keep on testing if more actions are desired within the test sequence.

Using TESTAR, you can start testing immediately and you do not need to specify test cases in advance. TESTAR automatically generates and executes test sequences based on a structure that is automatically derived from the UI through the accessibility API. Without specifying anything, TESTAR can detect the violation of general-purpose system requirements through implicit oracles like those stating that the SUT should not crash, the SUT should not find itself in an unresponsive state (freeze) and the GUI state should not contain any widget with suspicious words like *error*, *problem*, *exception*, etc.

[3] www.indenova.com/
[4] www.innovationalliance.eu
[5] Test* uses the Operating System's Accessibility API, which has the capability to detect and expose a GUI's widgets, and their corresponding properties like: display position, widget size, ancestor widgets, etc.

This is a very attractive feature for companies because it enables them to start testing immediately and refine the tests as we go.

3 Indenova and the SUT eSigna

Indenova is a Valencian ICT company that provides ERP (Enterprise Resource Planning) solutions for companies. Their initial clients are based in Spain. But throughout the years, Indenova has gained new clients in Latin America. Testing at Indenova is mainly done manually and basically is done at the system acceptance test level. Written requirements are used for the design of system test suites. They would like to have more tests automated, but currently in the company there is a lack of time and people with knowledge about test automation.

Becoming aware of Test* Indenova is very interested to see how they can start test automation, so they provided access to their eSigna product. This System Under Test is a web portal from which several services are available and enable users to perform specific processes inside their organisations. Thus, eSigna is a base component in which concrete services can be plugges-in as required by each particular project. Those services are independent from each other, but they are interconnected to share information in real time.

4 The industrial experience

During the investigation we have measured the following *effectiveness and efficiency* aspects of the setup, development and testing of Test*:

1. Number of failures observed after executing the Test* on eSigna
2. Time needed to set-up the test environment and get everything running
3. Lines Of Code (LOC) and time needed for error definition, oracle design, action definition and design of stopping criteria.
4. Time for running the Test* tool

The project has been carried out in a fashion that allowed us to perform iterative development of Test*. The process included the following steps which were repeated several times to yield the final setup:

1. Planning: Implementation of Test Environment, consisting of planning and implementing the technical details of the test environment for Test*, as well as the anticipating and identifying potential fault patterns in the Error Definition.
2. Implementation: Consisting of implementing the Test* protocol consisting of: Oracles to implement the detection of the errors defined in the previous step; Action Definition to define the action set from which Test* selects; and the Implementation of stopping criteria that determine when sufficient testing has been done by Test*.
3. Testing and Evaluation: Run the tests.

4.1 Planning the testing: what do we want to test

One of the immediate problems that Indenova faces with eSigna fits exactly with the Test* capabilities. As indicated the initial clients from Indenova were from Spain, but gradually they have expanded to Spanish speaking South American countries. One of the problem encountered is that there are differences between the Castilian Spanish spoken in Spain and the different Latin American Spanish. Although it is not a problem of not being able to understand what is meant, some of the clients from Columbia and Peru just have complained about the usage of Castilian words. For example:

English	Spain	Latin America
Mobile phone	Móvil	Celular
Holiday	Festivo	Feriado
Computer	Ordenador	Computadora

Since the implementation is not based on dictionaries and the Castilian Spanish is hard-coded, there is no other way than test the application to find the words that need to be changed for the other countries. This is a tedious and boring job.

4.2 Implementing the Test* protocol

Test* has the flexibility to adapt its default behaviour for specific needs. We will describe next how did we setup the tool to automatically verify localisation problems on *eSigna* product:

1. Set *eSigna* activation - it will tell Test* how to start/run the application. Being a web application, it consists of a command line *BROWSER URL* where *BROWSER* is the path and executable of an available web browser (i.e. Internet Explorer) and *URL* the entry point for the *eSigna* web application.
2. Set test algorithm - as we would like to verify any potential localisation issues around the target product' UI, we changed the default random algorithm to a more proper search based optimisation criteria that would better explore the SUT UI space.
3. Set suitable actions - from the space of candidate actions that the user could perform over the product UI we are interested in 1) actions that will enable an automatic login to *eSigna* and 2) actions which are not interesting for our localisation verifying objective (i.e. web browser actions, a logout button, an administration panel in *eSigna*, etc.)
4. Set localisation oracles - verifying the localisation correctness of *eSigna* for a target language can be straightforward performed by defining a list of taboo words that should not appear in the UI. This list can be easily defined in Test* UI through Java regular expressions (i.e. .*[mM][óo]vil.*|.*[fF]estivo.*|.*[oO]rdenador.*).
5. Set the stopping criteria - we might choose between a fixed time for execution, a fixed length for UI executed actions or use a more suitable stopping criteria.

We made use of the Test* test protocol to define a more proper stopping criteria for *eSigna* verification (check next point). It will account for the UI space exploration and stop when no more space is being explored.

6. Advanced test protocol setup - this protocol is a Java class composed of a method for each task in the testing cycle presented in figure 1. Concretely, we implemented the automated login inside the task *START SUT*, non-interesting actions filtering inside the task *DERIVE SET OF USER ACTIONS* and the stopping criteria in the *more actions?* check point.

Once Test* was setup for automated localisation verification we just had to wait for the tool test reports. Following the testing flow of Test* it would first activate eSigna, perform an automated login and repeat a cycle of <*select and execute action*, *verify localisation problems*, *check stopping criteria*>.

4.3 Testing and Evaluation

Our context multilingual scenario consisted of one target language, *Latin American* as this was the first concern on eSigna testing with Test*. We account in table 1 (LOC = Lines Of Code; time in minutes) for metrics that measure the effort required for our solution on automated verification of localisation issues.

Setting up Test* for eSigna is an easy process that consists on providing the command line that would activate the product. Actions configuration would require some effort though as we would like Test* to perform automated tests without user intervention. Thus, we first need to analyse eSigna authentication process to provide the proper actions once the product has been activated. Additionally, we wanted to maintain our tests in relevant UI parts of eSigna, for example disabling/filtering non interesting actions like closing the browser, login out of eSigna, etc. Yet, 35 lines of code and 10 minutes were enough for Test* to perform automated tests over eSigna. We acknowledge that future enhancements on Test* would enable a more efficient configuration of actions (we used version 1.1a of the tool).

Table 1. Efficiency (time in minutes)

Setup	**Actions**		**Oracles**		**Stop criteria**		**Test run**	
time	*LOC*	*time*	*LOC*	*time*	*LOC*	*time*	*time*	*actions*
1	35	10	0	5	9	2	60	100

Oracles did not require any lines of code, but just a regular expression with the full list of unwanted localised product words (e.g. Móvil, Festivo, Ordenador). We defined a regular expression for a list of more than 30 words that were most issued by the Latin American community. The company was aware of 2 particular words that were incorrectly localised.

The stopping criteria was easily implemented looking to the UI space exploration. We forced to stop the tests when no more space was being explored by the last 100 executed actions.

Finally, we observed that revealing the 2 wrongly localised words in eSigna, initially notified by Indenova, was reported fast by Test* (in the first 5 minutes of execution). Other words were not reported, but Indenova indicated that such words were not part of the product. We also observed that new UI space was explored after an hour of execution, which could reveal additional issues in the localised product. We expect a direct relation between the UI space exploration (coverage) and the effectiveness achieved on localisation verification of software products.

5 Conclusions and further work

We have presented a short experience of transferring an academic prototype from the university, for testing software applications at the UI level, to the industry. Indenova is a Valencian ICT company that provides ERP solutions to other companies. We applied the prototype Test* for testing localisation issues in eSigna product, which targets the Latin American countries.

The automation level achieved by the prototype and its potential for testing software products made Indenova consider the integration of Test* into their testing processes. They used the prototype for performing smoke testing, which would provide early feedback of the quality of developed product versions.

As further work, we will improve localisation testing in the prototype by including dictionaries. We would also like to further investigate the effectiveness of the presented localisation testing solution.

Acknowledgement

This work was partly funded by the SHIP project (EACEA/A2/UHB/CL 554187) and the PERTEST project (TIN2013-46928-C3-1-R). Test* was funded by the EC within the context of the FITTEST project, ICT-2009.1.2 no 257574 (2012-2015).

References

1. E. Alegroth, M. Nass, and H.H. Olsson. Jautomate: A tool for system- and acceptance-test automation. In *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, pages 439–446, March 2013.
2. Bao N. Nguyen, Bryan Robbins, Ishan Banerjee, and Atif M. Memon. Guitar: an innovative tool for automated testing of gui-driven software. *Autom. Softw. Eng.*, 21(1):65–105, 2014.
3. Tanja E.J. Vos, Peter M. Kruse, Nelly Condori-Fernández, Sebastian Bauersfeld, and Joachim Wegener. Testar: Tool support for test automation at the user interface level. *Int. J. Inf. Syst. Model. Des.*, 6(3):46–83, July 2015.