Plavis/FSM: an Environment to Integrate FSM-based Testing Tools

Adenilso da Silva Simão[1] , Ana Maria Ambrósio[2] , Sandra C. P. F. Fabbri[3] , Ana Silva Martins Serra do Amaral[2] , Eliane Martins[4] , José Carlos Maldonado[1]

[1] {adenilso,jcmaldon}@icmc.usp.br
Departamento de Ciência da Computação e Estatística – Universidade de São Paulo

[2]ana@dss.inpe.br, anasil@lac.inpe.br
Engenharia e Tecnologia Espaciais – Instituto Nacional de Pesquisas Espaciais

[3]sfabbri@dc.ufscar.br
Departamento de Computação – Universidade Federal de São Carlos

[4]eliane@ic.unicamp.br
Instituto de Computação – Universidade de Campinas

Abstract. *PLAVIS project aims at establishing the basis for a platform (an integrated set of tools) to support the VV&T activities, providing an environment that improves the validation and verification process applied to space software systems. This paper introduces an environment, named* Plavis/FSM*, which integrates the tools Condado, MGASet and Proteum/FSM. These tools were developed in previous research projects by different groups for supporting the generation of test cases from FSM-based specification.*

1. Introduction

Creating tests manually has several disadvantages, as pointed by different authors (Bertolino et al., 2005; Robinson, 1999). First, tests come late in the development cycle; when bugs are revealed, the cost of rework and additional retesting is high and impacts the product release. Second, the increased complexity of software systems nowadays implies an indefinite number of input combinations that result in distinct system outputs, that a test analyst cannot achieve an adequate coverage of these combinations by manual testing. Third, handcrafted tests are difficult to maintain and reuse. Finally, the oracle, a mechanism used for test results analysis, is inaccurate. The term model-based testing refers to test case derivation from a model representing software behavior (Bertolino et al., 2005).

Model-based testing presents several advantages. Behavior models are constructed early in the development cycle, which allows testing activities to start before the coding phase. In this way, tests can be based on what the software should do, and not on what the software do (Apfelbaum and Doyle, 1997).

In this context, the development of supporting tools is mandatory. However, the effort of developing a supporting tool is fairly high. In the Academy, the aim of producing a full-featured tool can hardly be afforded and must often be abandoned in favor of more modest goals. In general, proof-of-concepts tools are developed. Besides the cost of development, one of the main setbacks is the exploratory aspect of a proof-of-concept tool. Sometimes, the kind of empirical insight provided by the tool is what is necessary to make fine-tuning adjustments in the theory. In order to foster the transfer of some new technology to industry, it is necessary to provide evidences that the new technology brings substantial, significant benefits over the

current practices, if any. In this context, proof-of-concept tools are very important, aiding the conduction of case studies and experiments.

PLAVIS (PLAVIS, 2005) (PLAtform for Software Validation & Integration on Space Systems) is a joint cooperation project among Brazilian research institutes and universities. There is also a cooperation with France supported by Capes-Cofecub. The basis of this project is to establish a strong cooperation among universities and research institutes and centers that possess the domain of the technology in research, use and evaluation of best software engineering practices and state of the art VV&T tools. The project aims at establishing the basis for a platform (an integrated set of tools developed by different groups) to support the VV&T activities in order to provide an environment to improve the validation and verification process applied to space software systems. A prototype has been designed and implemented and it consists of software testing tools that will enable to experiment and evaluate practical case studies. Case studies have been focused on space applications from the National Institute for Space Research (INPE).

The set of available tools provides an environment not only to carry out an uniform, systematic and high-quality VV&T activities, but also to carry out empirical studies to evaluate the cost and benefits of new technologies. In the current stage, only the Finite State Machine (FSM)-based tools were integrated in PLAVIS, so this version is named Plavis/FSM.

Specifically, the Plavis/FSM integrates Proteum/FSM (Fabbri et al., 1999), Condado (Martins et al., 1999) and MGASet (Candolo et al., 2001). These three tools support distinct approaches and were developed independently by different researchers along different periods. With the integration of these tools, we expect to be able to conduct experiments that compare their ability in revealing faults and that aid the definition of a strategy to synergistically use the strength of each.

The paper is organized as follows. In Section 2, we describe Plavis/FSM and the tools that is already integrated in it. We briefly describe the tools integrated in Plavis/FSM, namely, Proteum/FSM (Section 2.1), MGASet (Section 2.2) and Condado (Section 2.3). Finally, in Section 3 we make some concluding remarks and point to future directions in this research.

2. Plavis/FSM

A main goal of PLAVIS project is the comparison of state-of-art methods and their gradual transfer to industry environment. Several problems rise from the need of comparing different methods. The comparison should be supported by a tool, whose main task is to integrate the results of the existing tools for the underlying methods. However, the integration of the tools poses other problems.

Although the underlying tools work on FSM-based specifications (models), slight variations of the underlying models do exist. Moreover, distinct model properties are required by the tools. So, if the models upon which the tools work are not the same, the results can be inconsistent.

Another important point is the data format the tool requires as input. This problem is due to the independent development of each tool. The developers attempted to design an input format that is most appropriate to their own tool. The lack of a standard format makes each developer choose the format that best fits his/her goals. Moreover, the results of the tools can also vary a lot. This problem is also due to differences in the underlying model and in the data format. The problem with the distinct input format can be solved in a quite straightforward way, by designing converters to and from a particular format.

2.1. Proteum/FSM

Although Mutation Testing is primarily aimed at assessing the quality of an existing test set T, it can also be used to enhance the capabilities of testing by expanding T with tests targeted at undetected mutations. A key issue in mutation testing is the identification of a fault model, which states which faults should be detected. Fault-based testing can be designed according to the classes of faults a generic model encompasses.

Proteum/FSM is one component of a family of tools (Maldonado et al., 2000) for supporting the application of Mutation Testing in several contexts, such as program testing (Delamaro et al., 2001) and specification testing (Fabbri et al., 1994). All the Proteum Family tools support the minimal set of operations that must be provided by any mutation based testing tool, such as, test case handling (execution, inclusion/exclusion and disabling/enabling), mutant handling (creation, selection, execution, and analysis) and adequacy analysis (mutation score and reports).

Proteum/FSM was developed as Web-based application. Therefore, it can be used remotely and in a wider range of environments. It implements the mutant operators defined by Fabbri et al. (1994) using the $\mathcal{MuDeL}$ language (Simão and Maldonado, 2002) and used the `mudelgen` system for actually generating the mutants.

2.2. MGASet

MGASet (Candolo et al., 2001) is a tool for generation of test cases for FSM. It was designed in order to incorporate several distinct algorithms for generating test cases, as well as functionalities for editing and simulating FSM. Some useful properties of the FSM can also be verified, such as, completeness, determinism and minimality.

MGASet is primarily intended to be a tool for experiment with new method of test case generation. Currently, however, it has implemented only W method. In the forthcoming steps in the research, we will study and embody other generation method, such Wp, UIO, and DS (Fujiwara et al., 1991).

MGASet was developed in Java. It can be run either though a GUI or by command-line invocations.

2.3. Condado

Condado tool (Martins, 1999) was developed in the context of ATIFS project (ATIFS, 2005) for automatically generating specification-based test cases. It accepts a FSM or a Extended Finite State Machine (EFSM) as input and generates tests considering both aspects: control and data. The control part addresses the valid sequences of interactions, while the data part addresses values of interactions parameters.

For the data part, the domain testing technique has been integrated. The domain testing technique avoids the state-explosion problem. This problem means that the number of different states of the system is too large to be represented; it can occur if the specification is expanded for all possible values of interaction parameters and state variable values. For the control part, the algorithm for test case generation is based on depth-first search.

Condado was developed in Prolog. Its input is furnished as a Prolog fact base. The FSM model may be given in a protocol specifications language or graphically via the MME tool (ATIFS, 2005). Its output is a set of test cases written in a text file.

2.4. Plavis/FSM: Architecture

The integration promoted by Plavis/FSM is based on data. The platform converts its internal representation of the FSM to and from each of the particular data format of the integrated tools. The platform prepares the data in the suitable way and invokes the respective tool, which runs in a controlled environment. Then, the platform

collects the results the tool puts in a particular file and converts them to its internal format. During the preparation of the data, the FSM is adequately transformed in order to fulfill the requirements of the tool. The common internal representation allows us to compare the test case generated by the tools, as well as to assess the adequacy of these test cases with respect to the Mutation Testing.

Plavis/FSM architecture is scratched in Figure 1. Plavis/FSM was designed and developed as a Web application and is available for remote usage. The platform implements a password-based access control[1]. Our primary motivation was to allow that the platform be used in a wider range of environments. The kernel of the platform is the test case manager module (TCM). It not only coordinates the communication with the other module, but also processes the user inputs and deliveries his/her the output. This module embodies the main functionalities of the whole platform.

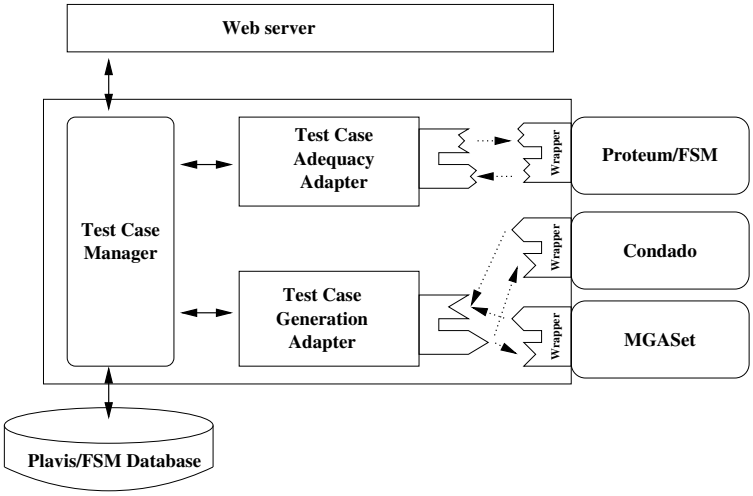

Figure 1: Plavis/FSM Architecture.

Whenever it is necessary to interact with one of the integrated tools, the TCM delegates the task to either of the adapters. There are two kinds of adapters. One adapter interacts with tools for test case generation (TCGA) and the other interacts with tools for test case adequacy analysis (TCAA). In order to make the adapters more reusable, it only provides a public interface the tools can interact with. However, in the case where the tool is legacy code and can not be easily modified to use either the adapters, a wrapper can be provided. That is the situation with the tools we have integrated so far. The wrapper as also responsible for fulfilling the requirements of input and for converting the data in and out the legacy tools. Moreover, the wrapper must get rid of any user interface.

To integrate another tool to platform, it is necessary to provide a suitable wrapper for the tool. Provided that the wrapper can convert the input and output of the tool to the format the adapters accepts, the integration could be carried out easily.

In Figure 2 is shown one of the Plavis/FSM Web pages. In this page, a summary of the session is shown on the left, including the score mutation provided by Proteum/FSM and the number of test cases generated by either tool. On the top, it is displayed the available options, allowing to include/remove test cases, include/remove mutants and access reports of the execution.

2.5. Plavis/FSM: Operational Aspects

The usage of Plavis/FSM imposes a workflow that is inspired in the way the tools of Proteum family work (Delamaro et al., 2001). In this way, the platform has some

[1]The tool can be accessed by the address: `http://143.107.183.159/~plavisFSM/main.php`. For a trial, the interested reader can request a guest account in the Register link.

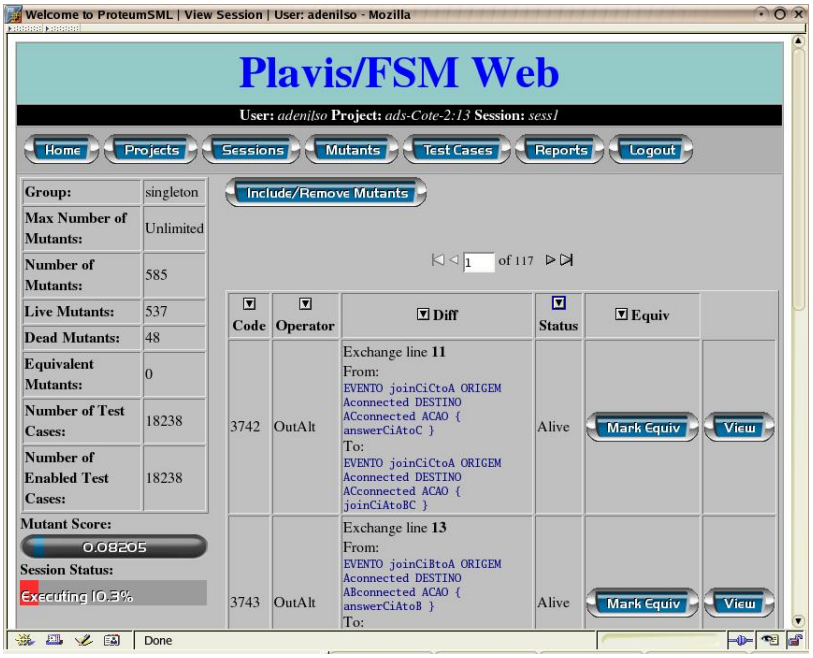

Figure 2: Plavis/FSM Mutant Viewer Page.

feature to allow the collaborative work among a set of testers. First of all, a project should be created, in which the FSM is selected, and some high-level configurations are made. In particular, the set of mutant operators that will be applied are chosen. Then, several sessions can be created within a given project. For instance, a set of sessions can be run with test cases generated by different methods.

It allows three ways of test case inclusion, either interactively or by the generation tools. (i) In the interactive inclusion, the user of the tool can choose which event should occur in the currently displayed state. (ii) To use the generation tools, the user can choose which tool to use (currently, either Condado or MGASet), and how many test cases he/she wants to include. (iii) Alternatively, a set of test cases can be furnished by submitting a appropriately formatted file. After having included a set of test cases, the user can require the mutant score of the test cases. In this case, Proteum/FSM is invoked with these test cases and the score is updated in the page. Plavis/FSM also allows to view the mutants, select the live ones and mark them as equivalent, if necessary.

The set of test cases can be exported. A XML configuration file, that should be provided by the user, defines the format of the test cases. In this way, the test cases can be converted in a format suitable for other testing tools.

3. Concluding Remarks

In this paper we introduced Plavis/FSM, a integrated platform for supporting the validation of FSM-based systems. The platform combines previous effort dedicated to the development of tools in academic researches. In this vein, it provides a unified mechanism to employ the tools and compare their results.

During the integration, we had to solve two distinct problems. First, we had to choose a model each tool can deal with. Although this seems to be a simple problem, it is hindered by the slightly different definition of FSM. Actually, the main problems are the properties that each tool demands the FSM to have in order to be applicable. For instance, W method requires that the FSM be a Mealy machine which is completely specified, minimal and that every state is reachable from the initial state. On the other hand, Switch-cover method does not require that the

FSM be completely specified. In this version of Plavis/FSM only the control part of Condado is available.

Forthcoming steps of this work, includes to incorporate other generation methods, such as Wp method (Fujiwara et al., 1991). The greater the number of methods integrated in PLAVIS/FSM, the wider the comparison results and the better the insights on how to complementarily use them. Additionally, the definition and conduction of experiments will provide evidences of the benefits of employing them. Currently, PLAVIS/FSM is already been used in the context of graduate and undergraduate case studies. New case studies with real applications are also planned.

References

Apfelbaum, L. and Doyle, J. (1997). Model based testing. In *Software Quality Week Conference*.

ATIFS (2005). Ambiente de teste com injeção de falhas por software. Online in: `http://www.inpe.br/atifs`, last access: 26/04/2005.

Bertolino, A., Marchetti, E., and Muccini, H. (2005). Introducing a reasonably complete and coherent approach for model-based testing. *Electronic Notes of Theoretical Computer Science*, (16):85–97.

Candolo, M. A. P., Simão, A. S., and Maldonado, J. C. (2001). MGASet — uma ferramenta para apoiar o teste e validação de especificações baseadas em máquinas de estado finito. In *Anais do XV Simpósio Brasileiro de Engenharia de Software*, pages 386–391, Rio de Janeiro, RJ.

Delamaro, M. E., Maldonado, J. C., and Mathur, A. P. (2001). Interface mutation: An approach for integration testing. *IEEE Transactions on Software Engineering*, 27(3):228–247.

Fabbri, S. C. P. F., Maldonado, J. C., Delamaro, M. E., and Masiero, P. C. (1999). Proteum/FSM: A tool to support finite state machine validation based on mutation testing. In *XIX SCCC - International Conference of the Chilean Computer Science Society*, pages 96–104, Talca, Chile.

Fabbri, S. C. P. F., Maldonado, J. C., Masiero, P. C., and Delamaro, M. E. (1994). Mutation analysis testing for finite state machines. In *Fifth International Symposium on Software Reliability Engineering*, pages 220–229, Monterey, California, USA.

Fujiwara, S., Bochmann, G. V., Khendek, F., Amalou, M., and Ghedamsi, A. (1991). Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17(6):591–603.

Maldonado, J. C., Delamaro, M. E., Fabbri, S. C. P. F., Simão, A. S., Sugeta, T., Vincenzi, A. M. R., and Masiero, P. . C. (2000). Proteum: A family of tools to support specification and program testing based on mutation. In *Mutation 2000*, pages 146–149, San Jose, California.

Martins, E., M., S. B. S. A., and Ambrosio (1999). Condata: a tool for automating specification-based test case generation for communication systems. *Software Quality Journal*, 8(4):303–319.

PLAVIS (2005). Plavis - platform for software validation & integration on space systems. Online in: `http://www.labes.icmc.usp.br/plavis/index.html`, last access: 29/04/2005.

Robinson, H. (1999). Graph theory techniques in model-based testing. In *Proceedings of the 16th Conference on Testing Computer Software*.

Simão, A. S. and Maldonado, J. C. (2002). MuDeL: A language and a system for describing and generating mutants. *Journal of the Brazilian Computer Society*, 8(1):73–86.