Test Case Generation for Critical Systems through a Collaborative Web-based tool

Alessandro Oliveira Arantes[1], Nandamudi Lankalapalli Vijaykumar[2], Valdivino Alexandre de Santiago Junior[2], Danielle Guimarães[2]
[1] *Institute for Advanced Space Studies (IEAv) - Aerospace Technological Center (CTA)*
P. O. Box 6044 – 12228-970 – São José dos Campos – SP – Brazil
[2] *National Institute for Space Research (INPE)*
P. O. Box 515 – 12245-970 – São José dos Campos – SP – Brazil
alessandro.arantes@ieav.cta.br, vijay@lac.inpe.br, valdivino@das.inpe.br, danielle.guimaraes@cea.inpe.br

Abstract

Tests play a major role in validating software. In particular, the role becomes more important when considering critical software such as for space applications as is the case in the National Institute for Space Research (INPE) in Brazil. Such software uses Finite State Machines (FSM) in order to model the software specification from which test sequences are generated for a black box test approach. As the software for space applications is considered as a complex system with several components (usually in parallel), test designers seem to look for other alternatives instead of modeling via FSM. This paper addresses an experience in the modeling issue in using Statecharts to represent the specification of space application software from which test sequences can be generated. Moreover, it also describes a web-based tool in order to facilitate software testing, from models specified in Statecharts, in a distributed environment.

1. Introduction

Nowadays software is an essential element in our lives and it is embedded into a wide variety of devices from simple electronic devices as music players to scientific satellites. Especially when dealing with critical systems, testing activities are an essential phase in order to validate software, since their engineering processes demand high level and high cost technologies to perform complex tasks. Therefore, space agencies that deal with complex missions, naturally demand higher software quality in lieu of the huge investments in their missions [1].

The subject discussed in this paper is, in fact, the most important phase in a Verification and Validation process being one of the key issues within software development life cycle, and in particular, for critical software this activity gains much more importance dedicating more time and resources when compared to other phases within the cycle. INPE (National Institute for Space Research) is a government institute responsible to develop the Brazilian Space Mission involving satellites and, consequently, to develop and implement their embedded software.

Complex software requires a long and expensive development process, and therefore, costs for correcting errors during final development phases turn out to be extremely expensive. For these cases, modeling techniques are welcome to provide a formal specification of software, so that tests can be applied in order to detect errors during initial development phases. The quality of test sets applied to critical systems is heavily associated to Reliability and Safety, which consequently leads to the necessity of generating proper test sequences. And, in order to avoid their (test sequences) inadequacy in revealing errors, an effective way is to adopt an entirely scientific based modeling technique.

In this paper, Section 2 discusses about software modeling techniques, in particular Statecharts. Section 3 shows the importance of testing critical systems in a collaborative scenario and also presents PerformCharts tool by explaining how a model represented in Statecharts is converted into a FSM from which test sequences are generated. Section 4 presents the WEB-PerformCharts tool. Section 5 presents a case study with results from implemented methods. Finally, Section 6 concludes the paper.

978-0-7695-3514-2/08 $25.00 © 2008 IEEE
DOI 10.1109/CIMCA.2008.143

2. Software Modeling and Statecharts

Modeling techniques are an approach that has become common for domain-specific applications recently, in particular for software development [3]. Reactive systems, that is the focus of this paper, has a very particular characteristic that is based on reactions to stimuli or events; thus, a natural choice for representing reactive systems is FSM as it can be represented graphically by a state-transition diagram. However, features as depth and parallelism are not easily specified in a straightforward manner through FSM. So, a formal higher-level technique should be investigated. Some alternatives are Petri Nets [4], SDL [5], Statecharts [6] and others. The scope of this paper explores Statecharts alternative.

Statecharts have a graphical formalism to specify reactive systems ([7] and [8]). Originally, they have been developed to specify and simulate real time systems. They extend classic state-transition diagrams by adding features as hierarchy (depth), orthogonality (parallel activities) and interdependence (broadcast-communication). The elements that are used by Statecharts to represent reactive systems are: states, transitions, events, conditions, actions, variables and expressions [6], [7], [8] and [9].

3. Testing Critical Systems within a Collaborative Scenario

There are different types of tests and, depending on these types, they can be applied in different phases within a software development process. Even in modeling phases, before implementation, it is already possible to fix errors as long as a formal specification is available. Without any knowledge on its internal structure, testers have a perception that software is a "black box". Consequently, these tests based just on the software's specification are Functional Tests, also known as "black box tests".

Situations, where teams involved in satellite missions are not exactly in one place, are usual in space research activities, since joint collaborations are a very common trend among space agencies. Therefore, the use of an on-line collaborative tool would definitely aid the software testing activities developing space applications in this scenario such as the one shown in Figure 1.

Due to popularization of internet access, web-based applications become popular providing advantages by offering a lower cost solution than conventional applications. In this architecture, a client does not require proprietary software, and can use any operating system as well as any internet browsers available free of costs. Also, whenever necessary, updates may be conducted only in the web-server where the application is hosted, without any necessity for the users to reinstall any kind of software. So, collaborative web-based applications (or E-collaboration) can be seen as a natural approach adopted for many companies of varied segments; and in this work a collaborative application was developed in order to generate "black box tests" for software specifications uploaded via internet.

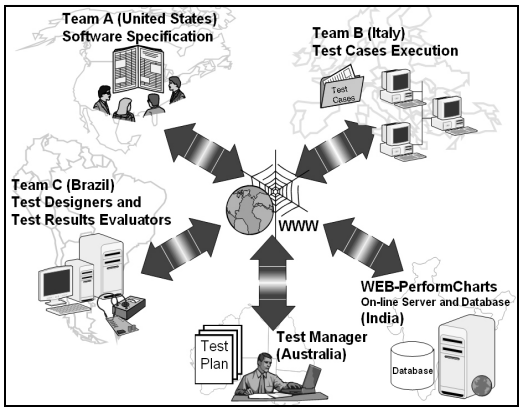

Figure 1. Example of cooperative work

PerformCharts is a tool used to generate test sequences from Statecharts specifications. It was initially developed to be used to evaluate performance of reactive systems by associating them to Markov Chains [10]. However, since Markov Chains can be represented graphically by a state-transition diagram, PerformCharts tool has also been used to generate test sequences. Or, in other words, PerformCharts converts a Statecharts model (software specification) in a FSM from which test sequences can be derived.

So, the specification of a reactive system in Statecharts and generation of FSM (or Markov chain) have to be coded as calls to methods in C++ language as a main module. In order to avoid this tedious coding, an XML-based language PcML (PerformCharts Markup Language) [11] has been developed

PcML code is edited by a text editor and parsed by a Perl script that converts it to the main program in C++. Thus, this program in C++ (that is a main program) is linked and compiled with PerformCharts classes and when executed, the corresponding FSM is generated. Test sequences are generated based on this FSM, as mentioned before.

4. WEB-PerformCharts

With the objective of enabling different teams, distributed geographically in different locations, working in software testing sharing projects through Internet access, PerformCharts was modified to become WEB-PerformCharts. It is a web-based tool to help software testers working in different places cooperating in common projects, and using their expertise and know-how in order to benefit software's quality. PerformCharts tool has been modified and adapted to execute remotely through a web-based interface and to be hosted in a web server. Also, instead of manipulating local files spread in several computers, an on-line database has been implemented in order to able testers to load and save projects from anywhere to the web server. Besides the traditional HTML, other resources for implementation were required, and the preference was for cost-free technologies such as PHP, MySQL for databases and Apache web server software. At the moment, WEB-PerformCharts uses a platform based on Windows servers; but a Linux version is in its final stage and about to be released.

Statecharts specifications (written in PcML) are distributed in projects that can be created by any user and can be shared among users. Once logged in the system, testers are able to create, edit or delete projects and their associated specifications. Tester can modify or run the test case generation method as many times as required. This is an important feature especially when software is incorrectly modeled or has to undergo changes in its specification. These changes can be perceived by anyone who can access the same project.

In theory the number of users who can access WEB-PerformCharts is not limited, since it depends directly on the server capacity to support a heavy on-line workload as well as on the storage capacity. The interface was developed keeping in mind the facilities to provide the user features to manage her or his projects creating a new one, deleting or modifying an existing project in order to obtain new test cases running the test case generator. Test cases are stored in database, and can be accessed anytime by those who have the proper authorization. A text file with PcML specification is uploaded to web server (WEB-PerformCharts) when user selects it using the provided interface implemented in HTML and PHP. When uploaded, the PcML contents are automatically parsed by a PHP script which extracts any Statecharts specification data and insert them into a MySQL database. This data inform the database is read and used to invoke proper methods holding the encapsulation, states, events, conditions, parallel components and transitions. It calls appropriate instances from PerformCharts and generates the FSM from its specification. If performance evaluation is required, a Markov chain is the result instead of FSM; but in either case (Markov chain or FSM) the output is inserted in database and can be extracted in XML format for any other use.

However, once FSM is available, test sequences can be generated from it by application of methods. A previous version of WEB-PerformCharts just generates FSM through a collaborative methodology, and it depended on another tool Condado [10] to generate test cases. However, CONDADO that implemented Switch Cover method cannot generate the sequences depending on the complexity of the FSM. The reason why some FSMs cannot be dealt with is not clear. Therefore, the present version of WEB-PerformCharts incorporated two test sequence generation methods: Transition Tour [12] and Switch Cover [13]. These methods have been developed as "cartridges" to be applied on the resulting FSM from Statecharts representation. This "cartridge" approach enables developing and implementing other methods that can generate test sequences based on FSM modeling. In fact, other methods DS (Distinguishing Sequence) [12] and UIO (Unique Input Output) [14] are being implemented at the moment. Figure 2 describes all basic steps to generate test sequences using WEB-PerformCharts. The generation using Transition Tour is indicated as "Path A", and Switch Cover method as "Path B". One major advantage in implementing Switch Cover within the web tool is that those complex FSMs that CONDADO couldn't deal with, are now being handled.

5. Case study

In order to show the use of WEB-PerformCharts for test sequence generation, a following case study is considered. This is a piece of APEX [15] software, an astrophysical experiment aboard on a Brazilian scientific satellite; more precisely, this is a command recognition component. Command messages are sent in a format composed of six fields: SYNC (EB9 synchronization value), EID (experiment identification), TYPE (specifies accepted commands), SIZE (amount of bytes in the DATA field), DATA and CKSUM (8-bit checksum). SIZE and DATA fields are optional and depend on the type of command. The behavior of command recognition component software is shown in Figure 3 and it is a low-level modeling since it is possible to see all the specified values of the protocol frame fields.

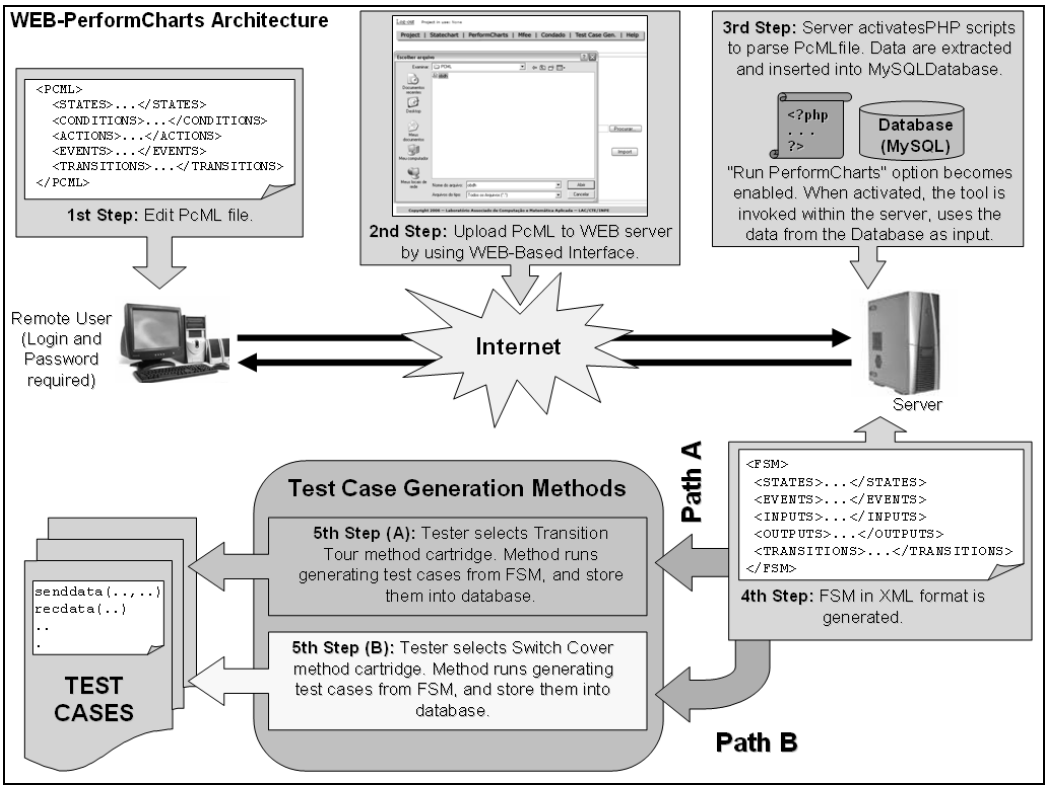

Figure 2. WEB-PerformCharts architecture

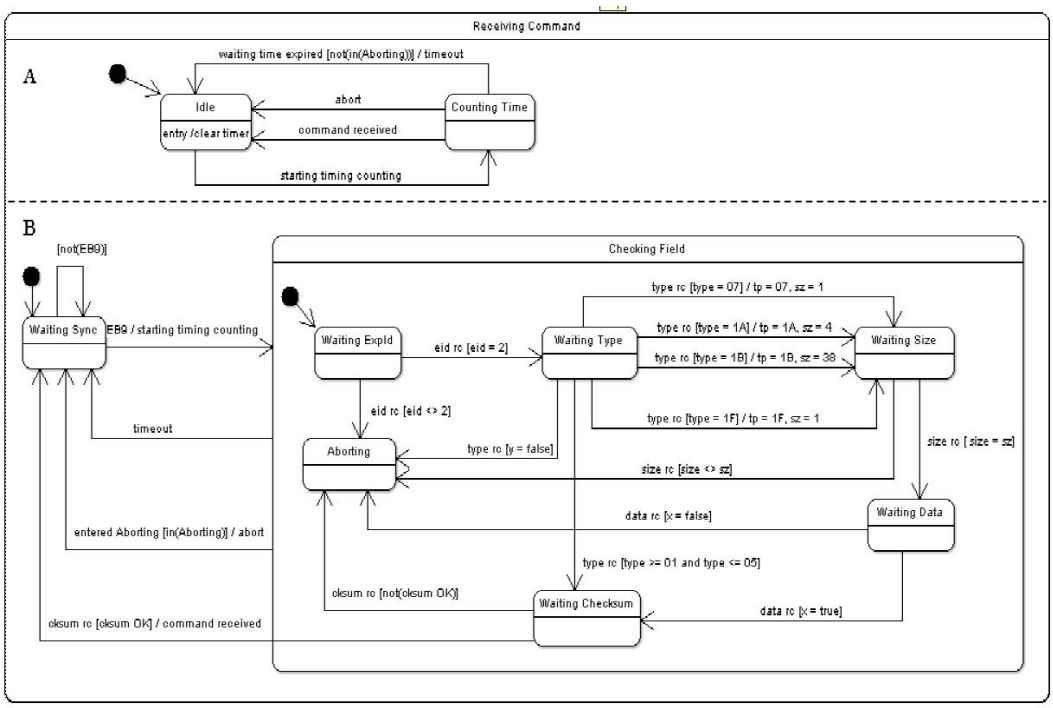

Figure 3. Statecharts representation of APEX system [15]

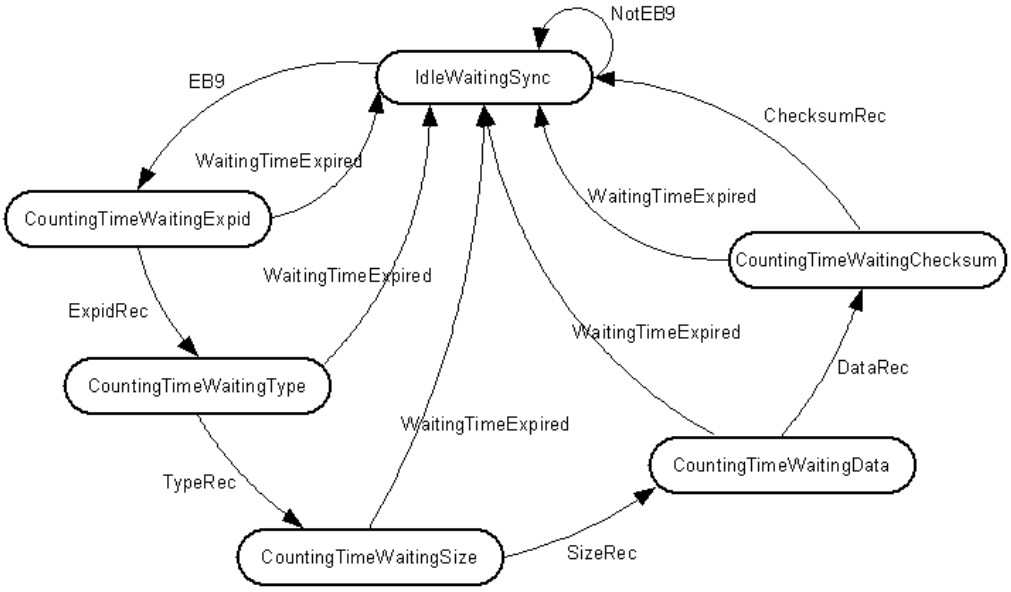

Figure 4. FSM generated from Statecharts specification in Figure 3

Figure 4 shows the FSM generated from this specification in Statecharts, and Table 1 shows results obtained by WEB-PerformCharts generating test cases for this application. In Table 1, row "time" means the amount of time spent to generate test cases, rows "events" and "cases" mean, respectively, the number of events and the number of cases generated by both methods.

Table 1. Results from test case generation

Result/Method	Transition Tour	Switch Cover
Time	328ms	7s
Events	70 events	27 events
Cases	1 case	7 cases

As can be seen in Table 1, Transition Tour method, as expected, is faster in terms of performance, but less precise by stimulating 70 events in order to cover the full graph with its unique test case sequence. Switch Cover method covered all possible 7 paths from FSM by stimulating just 27 events. However, it spent much more time. Both methods may start and finish test sequences from an initial state (IdleWaitingSync, in this case), as can be seen in Figure 4.

6. Conclusions

The adoption of a collaborative tool enables decentralized work, that is a common practice for widely dispersed companies resulting in time and cost savings in modern days, since it decreases travel and infrastructure requirements. In space research scenario this type of system is applicable and useful since professionals may be located in different institutes and the union of their efforts, wherever they are, contributes a lot towards the development of critical systems and, consequently, for the success of space missions.

The implementation of an on-line database allows test designers to share their projects in real-time conditions, and facilitates control of versions since its management is easier than copying multiple local files from several computers. Also, another advantage of WEB-PerformCharts is the access from any place in the world at anytime, and its few requirements composed just by a computer or laptop, an internet connection and a web browser.

The integration of test sequence generation methods enables the comparison of different methods when using WEB-PerformCharts. Besides Transition Tour and Switch Cover, other methods are under development and may be available as cartridges of the system soon. The main contribution of this paper is to provide a web-based tool, in a distributed environment, that supports test processes for space software engineering, providing a time comparison of two

methods for generation of test sequences within a real space application. In addition, XML was used to format documents and it contributes in standardization of test data.

With respect to complex software modeling, it could be concluded that this requires features as explicit representation of hierarchy and parallel activities. Hence, Statecharts come into picture being used as a higher-level technique based on state-transitions diagrams. However, dealing with higher-level techniques increases complexity in developing an automated environment demanding much more computational effort, and one must be prepared to pay the price for this. A proof of this is that two main discrepancies were observed among tested methods: the execution time in favor of Transition Tour and the effectiveness in favor of Switch Cover.

Also, depending on the number of states and arcs of the generated FSM, the problem may be intractable, i.e., it may not be possible to generate test sequences in a reasonable amount of time. Therefore, small graphs can be easily processed by any method quickly; but for complex graphs, a method must be carefully chosen. Tester has to opt between waiting more time for a complete set of sequences from Switch Cover or obtain a fast unique, and not so precise, set of sequences from Transition Tour. Through the observation of all sequences generated it can be deduced that Switch Cover sequences are more reliable, but it is impossible to make such a measure without evaluating this issue in more details and formally.

In future, studies will be made for integration between WEB-PerformCharts and tools that perform automatic test execution in order to improve the automation of test process activities.

7. References

[1] M.F. Matiello, V.A. Santiago, A.M. Ambrósio, R. Costa, and L. Jogaib, "Verificação e Validação na terceirização de software embarcado em aplicações espaciais", *SBQS2006*, Vila Velha, ES, Brasil, 2006.

[2] Pressman, R.S., *Software engineering - a practitioner's approach*, 5th edition, McGraw-Hill International Editions, 2000.

[3] H. Xiao, M. Zhiyi, S. Weizhong, and G. Shao, "A metamodel for the notation of graphical modeling languages", *In: Computer Software and Applications Conference (COMPSAC)*, Vol. 1. 31st Annual International, Volume 1, Issue, July 2007, Page(s): 219 – 224, 24-27.

[4] D. Harel, A. Pnueli, J. Schmidt, and R. Sherman, "On the formal semantics of Statecharts", *IEEE Symposium on Logic in Computer Science*, Ithaca, USA, 1987.

[5] D. Harel, and M. Politi, *Modeling Reactive Systems with Statecharts: the Statemate Approach*, McGraw-Hill, USA, 1998.

[6] D. Harel, and A. Naamad, "The STATEMATE Semantics of Statecharts", *ACM Transactions on Software Engineering*, 1996, 5(4), pp. 293-333.

[7] I. Sommerville, *Software Engineering*, Addison Wesley, 2003.

[8] G.Y. Tian, and D. Taylor, "Design and Implementation of a Web-based Distributed Collaborative Design Environment", *IEEE Fifth International Conference on Information Visualisation*, London, UK, 2001, pp. 703-707.

[9] N.L. Vijaykumar, S.V. Carvalho, and V. Abdurahiman, "On proposing Statecharts to specify Performance Models", *International Transactions in Operational Research*, 2002, 9(3), pp. 321-336.

[10] V.A. Santiago, A.S.M. Amaral, N.L. Vijaykumar, M.F. Mattiello-Francisco, E. Martins, and O.C. Lopes, "A Practical Approach for Automated Test Case Generation using Statecharts", *In: 2nd International Workshop on Testing and Quality Assurance for Component-Based Systems*, IEEE COMPSAC Conference, Chicago, EUA, 2006, v. 2, p. 183-188.

[11] S.C.P.F. Fabbri, J.C. Maldonado, T. Sugeta, and P.C. Masiero, "Mutation Testing Applied to Validate Specifications Based on Statecharts", *Proceedings of the 10th International Symposium on Software Reliability Engineering*, November 01-04, 1999, p.210.

[12] D. Sidhu, and T. Leung, "Formal methods for protocol testing: a detailed study", *IEEE Transactions on Software Engineering*, 1989, 15(4), 413-426.

[13] S. Pimont, and J.C. Rault, "An approach towards reliable Software", *Proceedings of the 4th International Conference on Software Engineering*, Munich, Germany, 1979, pp.220-230.

[14] K. Derderian, R.M. Hierons, M. Harman, and Q. Guo, "Automated unique input output sequence generation for conformance testing of FSMs", *The Computer Journal*, v.49 n.3, 2006, p.331-344.

[15] V.A. Santiago, N.L. Vijaykumar, D. Guimarães, A.S. Amaral, and E. Ferreira, "An environment for automated test case generation from statechart-based and finite state machine-based behavioral models", *In: 4th A-MOST 2008*, First IEEE International Conference on Software Testing Verification and Validation (ICST 2008), Lillehammer - Noruega, 2008.