WEB-PerformCharts: A Collaborative Web-based tool for Test Case Generation from Statecharts

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

+55-12-39475301, +55-12-39456549, +55-12-39457166, +55-12-39457179

alessandro.arantes@ieav.cta.br, vijay@lac.inpe.br, valdivino@das.inpe.br, danielle.guimaraes@cea.inpe.br

ABSTRACT

Distributed development of software has turned into a natural and modern approach where teams spread over the world cooperate to develop a software product, and this has become possible due to the expansion and popularity of global networks as internet. Collaborative tools coordinate a variety of tasks of several members of a team with an objective of reaching a specific goal. One such task that plays a major role, within the software development life cycle, is testing. In particular this task becomes more and more important when considering critical software such as space applications, which is the case of Brazilian Space Institutions CTA and INPE. The work discussed in this paper has two objectives: (i) present a web-based tool, WEB-PerformCharts, that can generate black-box test cases of a space application software; (ii) show that Statecharts are an excellent option to model the software specification, from which test sequences can be generated by applying several methods well known from the published literature.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification – *Assertion checkers, Class invariants, Correctness proofs, Formal methods, Model checking, Programming by contract, Reliability, Statistical methods, Validation.*

D.2.12 [**Software Engineering**]: Interoperability – *Data mapping, Distributed objects, Interface definition languages.*

General Terms

Reliability, Verification.

iiWAS2008, November 24–26, 2008, Linz, Austria.

Keywords

Web-based tool, Collaborative Applications, Software Tests, Test Case, Verification and Validation, Statecharts.

1. INTRODUCTION

Nowadays many software development companies are using computer-supported collaborative tools overcoming geographical distances in order to reduce development costs adopting distributed settings. These collaborative tools enable the collaborative work, that is nothing more than joint efforts, coordinating their tasks to conclude a project common to all members of a team. Thus, a collaborative web-based application would be an useful tool to help different teams to cooperatively address process activities related to the software development life cycle.

Testing activity is the topic discussed in this paper, and maybe, the most important phase in a Verification and Validation process. It is the software's operation with real or simulated inputs from real situations to demonstrate that software satisfies its requirements or, if it does not, to identify the differences between the expected results and obtained results generated by software under evaluation.

Verification and Validation [1] activity is one of the key issues within software development life cycle, and in particular, for critical software this activity is more important dedicating more time and resources when compared to other phases within the cycle. Brazilian Institutions such as Aerospace Technological Center (CTA) and National Institute for Space Research (INPE) are government bodies responsible to develop the Brazilian Space Mission involving both satellites and launch vehicles. These missions involve a significant amount of financial resources besides considering security issues with respect to avoiding risks to human lives and damage to environment. Space missions definitely demand organized and manageable activities related to software development. And this software is considered as critical software due to space application's inherent complexity. Besides, the unmanned aspect of satellite launching and satellite functioning in the desired orbit turns out to be an additional factor

increasing further the complexity. Reliability and safety of critical software depends heavily on the quality of test sets applied to the product, which consequently leads to the necessity of generating proper test sequences. This brings the need to work with an entirely scientific basis in order to avoid their (test sequences) inadequacy in revealing errors. So, testing activities can involve many scattered teams working together and they are essential depending on the software's complexity.

In this paper's context, software is represented as a state-machine based model, and a path from some given state to another state that is reachable, can be defined as a test sequence. If a requirements model is available, then, it is possible to use a technique to derive test sequences in the very early phases of the software development life cycle. Therefore, if the technique to model a software specification were a Finite State Machine (FSM), some methods that can be applied to generate test sequences are: T, UIO, DS, W and Switch Cover [2], [3], [4] and [5]. However, features usually present in complex software requirements representation, such as parallel activities and encapsulation, are very hard to represent using FSMs. Therefore high level techniques must be investigated to support representation of such features. WEB-PerformCharts tool, which is being presented in this paper, is a web-based tool that incorporates two main features for software testers: obtain test sequences remotely via internet by addressing distributed development situations; and, uses Statecharts, instead of FSM, for specifying requirements in order to generate test sequences.

This paper is organized as follows: Section 2 discusses an introduction about advantages of collaborative systems and their applications. Section 3 discusses the importance of testing critical systems and also presents PerformCharts by explaining how test sequences are generated from a model represented in Statecharts. Section 4 presents the WEB-PerformCharts tool. Section 5 presents a case study with results from implemented methods. Finally Section 6 concludes the paper.

2. COLLABORATIVE SYSTEMS

Word wide web offers resources for transmission of data at high speeds in which geographical distance is no longer a critical factor in today's world. Thus, the cooperative work among teams located in different places, geographically distant, has become a common trend and even necessary both in business and in academics [6]. This trend is further enhanced with the concept of globalization. The objective of collaborative systems is helping people involved in a common task supporting communication, coordination and cooperation. The use of these applications means accessibility for any internet user, allows a great cost saving, time saving, and increasing teamwork and efficiency since all manipulated data by one user can be immediately perceived by all other users at remote locations [6].

Web-based applications have advantages by offering a low cost solution, since in this architecture, the client can use any operating system and it requires no other proprietary software. Also, nowadays, many people have easy internet access and whenever updates are necessary, this is conducted only in the server where the applications are hosted without any necessity for the users to reinstall any kind of software. So, collaborative web-based systems (also known as E-collaboration) is a common practice adopted for many companies to develop their applications.

Collaborative tools can fall into the following categories: Group Document Handling, Real-time conferencing, Non real-time conferencing, Electronic Meeting Systems (EMS) and Electronic Workspace. In case of the tool WEB-PerformCharts, discussed in this paper, it belongs to the category of Electronic Workspace due to its main idea in offering teams a common environment for coordination and organization of their work centralizing files and documents in an on-line server [7]. Many features are commonly found in web-based applications, and those that are relevant for collaborative systems are:

(i) E-mail notifications: to communicate tasks, changes or new activities;

(ii) Project management: to control the access level of users and assign tasks to members of a group;

(iii) File and document sharing: availability of documents. Particularly in this work, software requirements specifications, software design documents and documentation related to the test process must be available to a group of people involved.

The most common feature in such tools, and at the same time most needed collaboration service, is file and document sharing [7]. Space research organizations demand high software quality, for instance, embedded into satellite or launch vehicle on-board computers. It is usual in this activity, situations where teams involved in satellite missions are not exactly in one place due to joint collaborations among space agencies to develop space applications. The use of an on-line collaborative tool would definitely aid the software testing activities in this scenario such as the one shown in Figure 1.

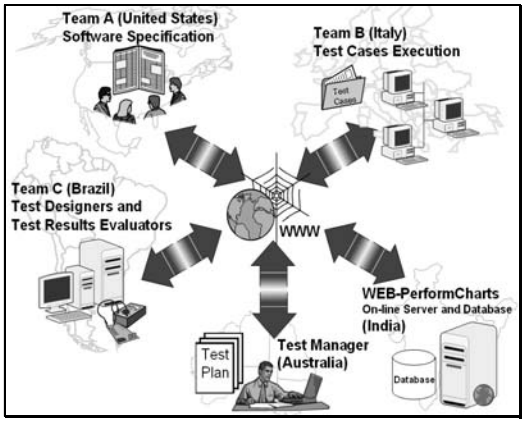

Figure 1. Example of cooperative work.

3. TESTING CRITICAL SYSTEMS

Several systems controlled by software may cause inconvenience when a fault occurs, but in most cases it does not cause serious damage. However, there is a certain category of systems, which are critical systems, where software faults can result in significant economic losses, physical damages or even threats to human lives [8]. As examples of critical systems one can mention space applications, navigation systems, banking systems or nuclear plant

monitoring. The focus of this paper is in space applications. Space missions deal with the execution of complex tasks using high cost technologies; consequently, these embedded software demands high quality and testing such systems is an essential activity to guarantee their reliability.

Tests can be applied in different phases of a system development process; even in modeling phases before implementation, it is already possible to fix errors testing a formal specification. These tests based on the software's specification without any knowledge on its internal structure are Functional Tests, also known as "black box tests". Therefore, there is a need to have a formal specification of the software to be tested, and this adds to the need of exploring techniques to represent complex reactive systems, like software embedded into satellite on-board computers. A natural choice for representing reactive systems is state-transition based techniques such as FSM, but features as depth and parallelism (usually common in modern complex reactive systems) are very hard to specify using FSM. So, a formal higher-level technique that can also be handled computationally for test case generation must be studied. There are some alternatives as Petri Nets [9], SDL [10], Statecharts and others. The scope of this paper explores Statecharts alternative.

Statecharts have a graphical language to specify reactive systems in a formal manner ([11] and [12]). They have been originally developed to represent and simulate real time systems. They have an added value of being formal and their visual appeal along with the potential features enable considering complex logic to represent the behavior of reactive systems. They originated from state-transition diagrams and these diagrams have been extended with notions of hierarchy (depth), orthogonality (parallel activities) and interdependence (broadcast-communication). Statecharts depend on the following elements in order to represent a reactive system: states, events, conditions, actions and transitions [11], [12], [13] and [14]. It is also possible to define variables and expressions.

States are clustered to represent depth, thus enabling to combine a set of states with common transitions into a super-state. Super-states are usually refined into further sub-states in a top down approach. State refinement can be achieved by XOR and AND decompositions. The former decomposition is employed whenever an encapsulation is a must to improve the clarity of the visualization. When an XOR super-state is active, one (and only one) of its sub-states is indeed active. The latter approach, AND decomposition, is used to represent concurrency and when active, all of its sub-states are active. The state that contains no more further refinements is known as BASIC.

In Statecharts the global state of a given model is referred to as a configuration that is the active basic states of each orthogonal component. In Statecharts, by definition, when modeling a given system, there must always be an initial state also known as default state. This is the entry point of the system. Another way to enter a system is through its history, i.e. when a system is entered the state most recently visited is activated, thus bypassing the initial state. In order to indicate that history is to be used instead of entry by default, the symbol H is provided. It is also possible to use the history all the way down to the lowest level as defined in the Statecharts formalism by applying the symbol H*. Transitions are generally represented by this notation event[condition]/action, which means that, once event is enable and condition satisfied, transition is executed. So, when transition is done, a reaction continues by executing the action, which is another reaction.

PerformCharts tool was initially created to be used to evaluate performance of reactive systems by associating them to Markov Chains, also represented as FSM [15]. In PerformCharts, events are divided into two categories: external and internal or immediate. External are those that have to be explicitly stimulated whereas internal are those that are automatically sensed by Statecharts dynamics and reaction takes place [15]. This is the same as defined in Statecharts. Actions, are considered as internal events that affect other orthogonal components. It is important to remember that a Markov chain is generated for performance evaluation, as long as the external events follow an exponential distribution. Also, it is possible to associate probability to transitions in situations where a same event takes a source state to more than one destination resulting in a conflict or non-determinism. Therefore, the original notation for transitions was changed in PerformCharts to: event[condition]{probability}/action [16]. This tool was written in C++ language.

In order to reach the objective of generating test sequences, PerformCharts was adapted to convert Statecharts representation of reactive systems into an FSM. This FSM had to be prepared and preprocessed so that it could be converted into an appropriate input to be used by another tool Condata [3] from which test sequences used to be generated. Condata implemented Switch Cover method [5]. As Condata was written in Prolog language, it takes the input of a state machine as a base of facts.

In PerformCharts, calls to methods for specifying the model in Statecharts as well as calls to methods to generate the corresponding FSM has to be written in C++. In order to provide a better interface, PerformCharts Markup Language (PcML) [17], an XML based interface has been developed in order to support Statecharts specifications.

PcML code is edited by any text editor and parsed by a Perl script that converts it to a main program in C++ of PerformCharts. Thus, this main program is linked and compiled with other classes and obtains performance measures or FSM. This FSM is the basis to generate test sequences.

4. WEB-PERFORMCHARTS

In order to enable different teams, distributed geographically in different locations, working in software testing sharing projects through Internet access, PerformCharts was modified to become a new tool named WEB-PerformCharts [18]. It is a web-based tool idealized to help software testers working in different places for cooperating in common projects, and approaching their expertise and know-how in order to benefit software's quality [18].

In order to reach this objective, PerformCharts tool has been modified to run remotely through a web-based interface and to be hosted in a web server using database access. This on-line database has been implemented in order to promote testers to load and save projects from anywhere to the server, instead of manipulating just local files spread in many computers.

Internet development technologies were required for implementation besides the traditional HTML, and the preference was for technologies free of costs as PHP, MySQL for databases

and Apache web server software. Thus, except for hardware costs, the system is entirely free of software packages costs. At the moment, WEB-PerformCharts is installed in Windows based platform servers; however, a Linux version is under development and will be available very soon.

Once logged into the system, testers are able to create, edit or delete projects and their associated PcML specifications. Each user can manipulate just one project at a time, and when a project is selected (from a list of all available projects) it can be modified and run the test case generation method as many times as required. It is an interesting feature since the software can be incorrectly modeled in Statecharts and may require changes in its specification. These changes can be perceived by anyone who can access the same project.

Specifications are distributed in projects, that can be created by any user and can be shared among users. The implementation of workflow routines is under study and the communication between them can be done through integration with their e-mail. The idea of group users into workgroups seems very useful and will be studied for implementation also.

The number of users who can access WEB-PerformCharts is not limited in theory. It depends directly on the server capacity to support on-line workload as well as on the storage memory.

In case of a huge number of users accessing the same server, they could be organized hierarchically according to their functions (e.g. Administrator, User, Guest, Project Manager, General Manager, etc.) providing an easier management. In fact, in its preliminary version, WEB-PerformCharts has two access levels for users: Administrator: full access for any project, and can create another user accounts; User: access just for projects created by her or him.

The web-based interface provides 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 method again. These test cases are stored in an on-line database in the server, and can be accessed anytime by those who have the proper authorization. WEB-PerformCharts automatically uploads a PcML specification to web server when user selects it using web-based interface, which is implemented in HTML and PHP. When uploaded, the PcML contents are automatically parsed by a PHP script which extracts any specification data and store them into a MySQL database. Data inserted in this database is read and used to invoke proper structures holding the encapsulation, states, events, conditions, parallel components and transitions. It calls appropriate methods from PerformCharts and generates the FSM from its Statecharts specification. If performance evaluation is required, a Markov chain is the result instead of FSM; but in either case, they (Markov chain or FSM) are stored in the database and can be extracted in XML format for any other use.

However, once FSM is available, methods can be applied in order to generate test sequences. WEB-PerformCharts is not limited to a single method. Besides its own embedded T-Method (Transition Tour) and the integration with Switch Cover (implemented in yet another tool, Condata), WEB-PerformCharts is opened for implementing any other method as long as the method can be applied on an FSM representation. The idea is to make these methods as independent cartridges within the system, and as an experiment, Switch Cover method was also implemented following this concept enabling its use without the need of Condata tool. Methods DS (Distinguishing Sequence) and UIO (Unique Input Output) are being implemented at the moment.

Recollecting, in test sequence generation, users have two alternatives within the WEB-PerformCharts tool. They can use one of the cartridges from WEB-PerformCharts, or they can export a base of facts which are input for Condata tool run its own method. This conversion is automatically achieved by using a parser written in XSLT. Figure 2 describes all basic steps to generate test sequences using WEB-PerformCharts. The generation using any of the cartridges methods is named as “Path A”, and integration with Condata tool is “Path B”.

5. RESULTS

In order to show the use of WEB-PerformCharts for test sequence generation, consider the example in Figure 3. This software behavior model was specified in the scope of the Qualidade do Software Embarcado em Aplicações Espaciais (QSEE - Quality of Space Application Embedded Software) research project [19]. This project is an experience at INPE in outsourcing the development of satellite payload embedded software. The software, SWPDC, is in charge of collecting and formatting data from Event Pre-Processors (EPPs), receiving and executing commands from the On-Board Data Handling (OBDH) computer, transmitting telemetry data to the OBDH, generating housekeeping information, accomplishing data memory management, implementing fault tolerance mechanisms and supporting loading of new programs on the fly. EPPs are front-end processors in charge of fast data processing of X-ray cameras signals of an astrophysical scientific experiment under development at INPE and the OBDH is the satellite platform computer [19].

A project like QSEE fits a collaborative systems approach. Taking into account only the on-board computers, it is perfectly possible that different organizations might be in charge of distinct computing subsystems development. For instance, one organization may be responsible for developing the OBDH, and its related software, another for the SWPDC computer, and the SWPDC itself, and even another for the EPPs and associated software. A completely distinct organization may be in charge of Verification and Validation of these software in an Independent approach, known as Independent Verification and Validation (IVV) [19]. In such scenario, WEB-PerformCharts comes into aid IVV’s test designers to generate test cases remotely via web.

Statecharts shown in Figure 3 is just a small part of the entire SWPDC modeling. It deals only with some state management of the software. Managing State is an AND state composed of four XOR states, denoted A, B, C and D. A and B are sub-states wondering if EPPs are active and able to send data collected during their operation or if they are inactive. Sub-state C models event report generation to be included in Housekeeping data to be sent to the OBDH. Housekeeping data have status information related to the health not only of SWPDC but also of the hardware of the computing subsystem. Sub-state D is related to data acquisition from EPPs by SWPDC computer. EPPs can generate three type of data known as Scientific, Diagnosis and Test data.

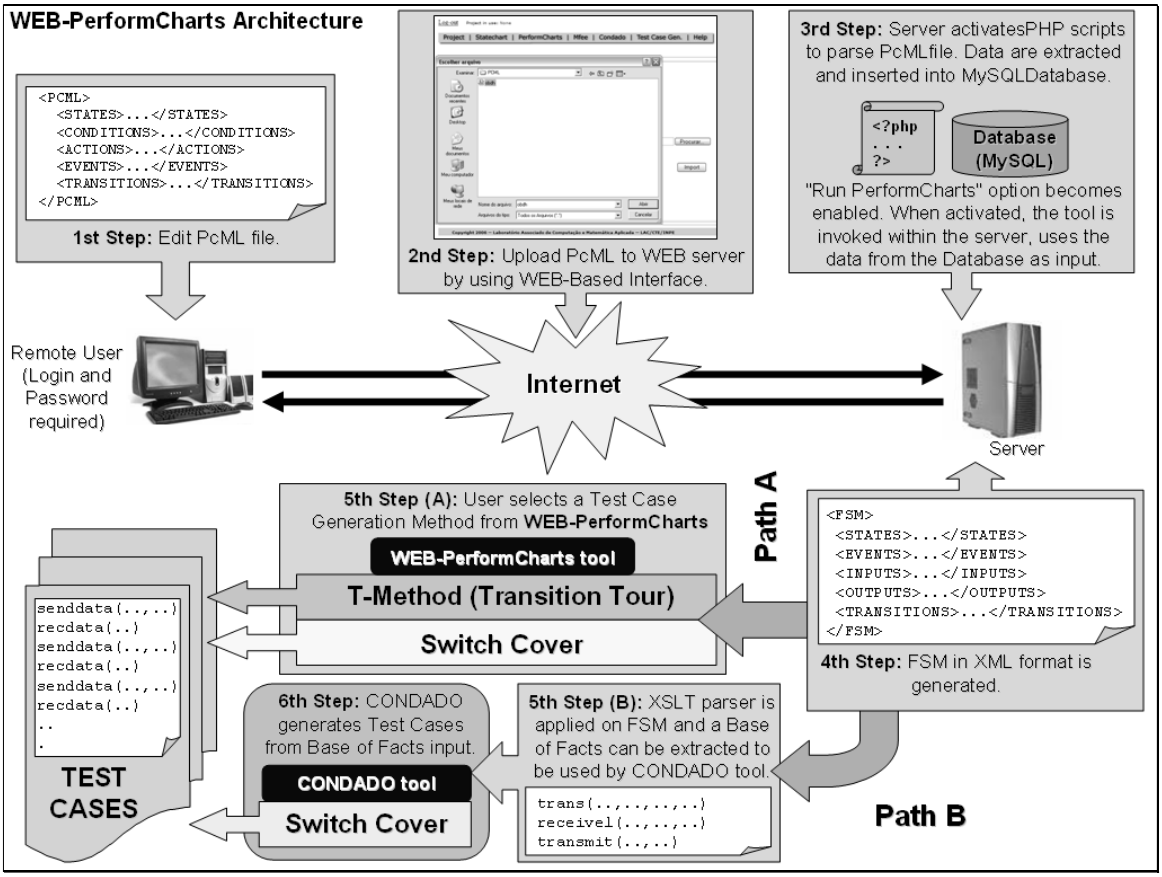

Figure 2. WEB-PerformCharts architecture.

So, SWPDC shall be able to interact with EPPs in order to request these data. In Figure 3, DD stands for Data – Diagnosis type, DT means Data – Test type, HK means Data – Housekeeping type and DM means Data – Dump type. For instance, prepare_DT event instructs SWPDC to acquire Test data from EPPs to be transmitted later to the OBDH. The methodology to generate test cases follows [20]:

1st step: The system specification is written in PcML. A part of such specification is shown in Figure 4.

2nd step: WEB-PerformCharts is accessed and PcML file is uploaded to Web server through the user interface shown in Figure 5.

3rd step: PcML file is automatically parsed by PHP and data are inserted into a MySQL database. "Run PerformCharts" option is enabled and generates a FSM from Statecharts specification.

4th step: FSM data is included into database and can be extracted as an XML file. Part of this file can be seen in Figure 6. Once FSM is obtained, tester can generate test sequences using Transition Tour or Switch Cover which are methods available within WEB-PerformCharts (Path A), or export a file suitable as input for Condata tool to run independently (Path B) from the WEB-PerformCharts tool. Both paths have been tested.

5th step (A): Transition Tour method was applied to the generated FSM and, this graph consisting of 40 states and 304 transition arcs was entirely covered using 1046 steps. The set of test sequences (with first and last steps) is shown in Table 1.

5th step (B): WEB-PerformCharts has an option "Get base of facts" that must be accessed in order to call an integrated XSLT parser. This parser is responsible in converting the XML data of FSM into the required input for Condata tool to generate test sequences. A part of this input is in Figure 7. Condata tool is implemented in Prolog and hence it requires the input as a base of facts.

6th step: Condata tool runs the input and applies Switch Cover method to obtain test sequences. But, for this case study the tool executed for some minutes without being able to finish due to the excessive number of combinations to be generated by this method. However, test sequences are still possible to be obtained by pruning a given transition

Except for Condata test cases that are generated locally, all other information presented are shared by any logged user in WEB-PerformCharts since they are totally stored into an on-line database and can be accessed in real-time conditions.

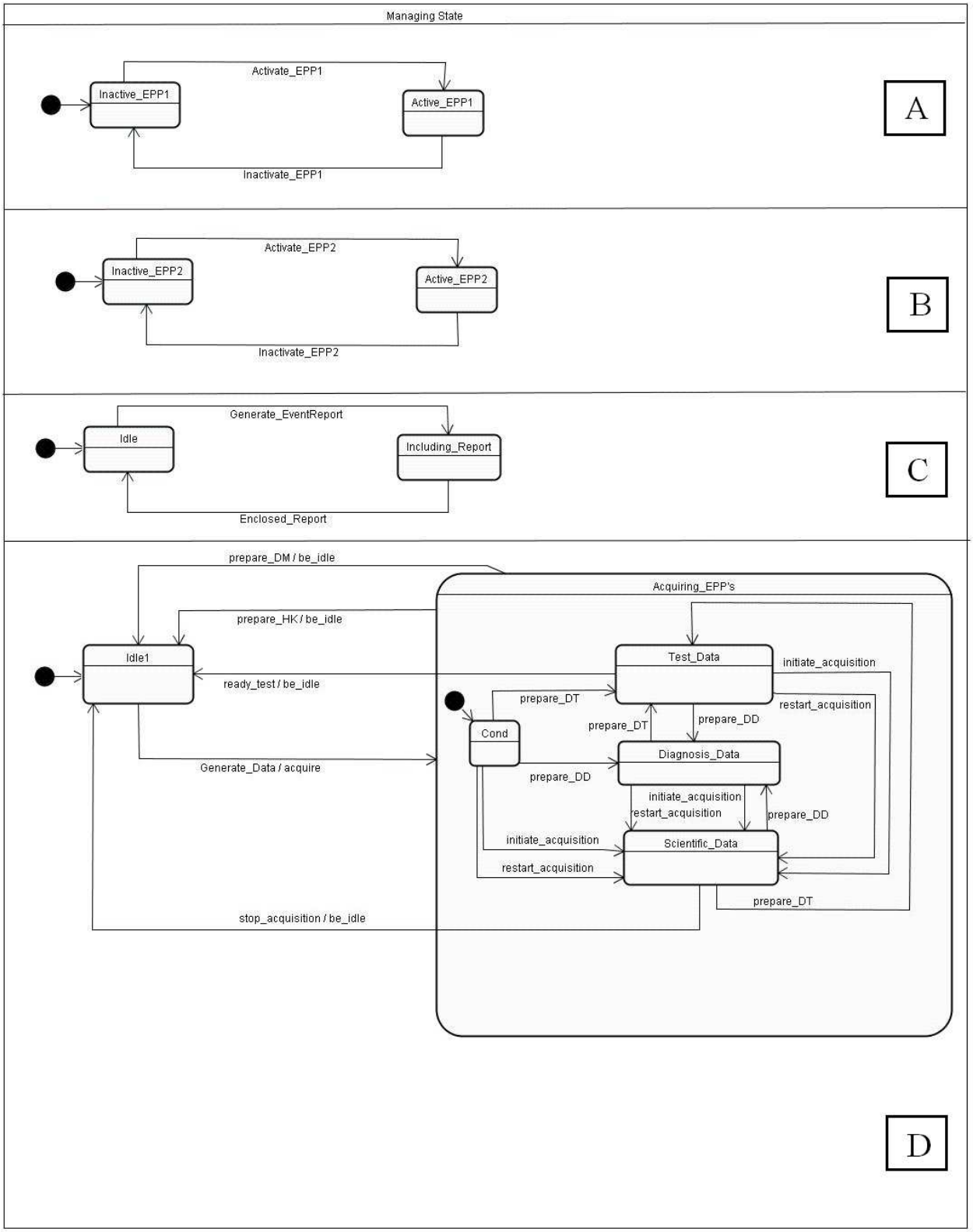

Figure 3. A small piece of a satellite computer embedded software modeling.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Pcml Title="Managing states" Date="2008-02-22" ...>
 <Info>
   <Author>
     <Name>Danielle S. Guimarães</Name>
     <Email>danielle.guimaraes@cea.inpe.br</Email>
   </Author>
   <Description>Managing Atates</Description>
 </Info>
 <States>
   <Root Name="Managing_states" Type="AND">
     <State Name="MS1" Type="XOR" Default="Inactive_EPP1">
     . . .
   </Root>
 </States>
 <Actions>
   <EventTriggerAction Name="eta1" Event="be_idle"/>
   . . .
 </Actions>
 <Events>
   <Stochastic Name="Activate_EPP1" Value="1.0"/>
   . . .
 </Events>
  <Transitions>
   <Transition Source="Inactive_EPP1"
               Event="Activate_EPP1"
               Destination="Active_EPP1"/>
   . . .
   </Transitions>
</Pcml>
```

Figure 4. PcML specification of modeling in Figure 3.

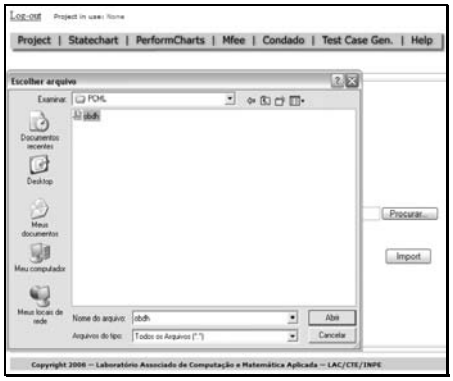

Figure 5. Web server upload interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="mfeeTesteX.xsl"?>
<MFEE>
  <STATES>
    <STATE NAME="inactiveEPP1InactiveEPP2IdleIdle1"
           TYPE="inicial"/>
    . . .
  </STATES>
  <EVENTS>
    <EVENT VALUE="1" NAME="Activate_EPP1"/>
    . . .
  </EVENTS>
  <INPUTS>
    <INPUT EVENT="Activate_EPP1"/>
    . . .
  </INPUTS>
  <OUTPUTS>
  </OUTPUTS>
  <TRANSITIONS>
    <TRANSITION SOURCE="inactiveEPP1InactiveEPP2IdleIdle1"
                DESTINATION="activeEPP1InactiveEPP2IdleIdle1">
      <INPUT INTERFACE="L">Activate_EPP1</INPUT>
      <OUTPUT></OUTPUT>
    </TRANSITION>
    . . .
  </TRANSITIONS>
</MFEE>
```

Figure 6. FSM specified in XML.

Table 1. Results obtained by T-Method

STEPS	EVENTS
1	Generate_Data
2	Initiate_Acquisition
3	Activate_EPP1
4	Deactivate_EPP1
5	Activate_EPP2
. . .	. . .
. .	. .
.	.
1043	Prepare_DM
1045	Deactivate_EPP1
1046	Enclosed_Report

```
trans(InactiveEPP1InactiveEPP2IdleIdle1,t1,
ActiveEPP1InactiveEPP2IdleIdle1,L0,Ln)
receivel('Activate_EPP1',L0,L1)
transmit(L1,Ln)

trans(InactiveEPP1InactiveEPP2IdleIdle1,t2,
InactiveEPP1ActiveEPP2IdleIdle1,L0,Ln)
receivel('Activate_EPP2',L0,L1)
transmit(L1,Ln)

. . .
. .
.
```

Figure 7. Base of facts for Condata tool.

6. CONCLUSIONS

Decentralized work is a very common trend for widely dispersed companies in modern days, since it can result in time and cost savings decreasing travel and infrastructure requirements, instead of maintaining huge, centralized and expensive buildings.

The decision for using an on-line database as storage method allow test designers to share their projects, and facilitates control of versions since its management is easier than copying multiple local files from multiple computers. Also, WEB-PerformCharts has other advantages when compared to conventional local systems, since it can be accessed from any place in the world at anytime with a computer or laptop, an internet connection and a web browser.

Complex software modeling requires features as explicit representation of hierarchy and parallel activities. Therefore, a higher-level technique based on state-transitions diagrams is recommended. In this respect, Statecharts come into picture. However, dealing with higher-level techniques increases complexity in developing an automated environment and demands more computational effort.

Depending on the number of states and arcs of the generated FSM, common sense says that the problem can be unfeasible. This could be seen by Switch Cover method from Condata tool that resulted in state explosion with this relatively complex example. Now, once the modeling issue is decided, a test sequence generation method has to be selected. In the example shown, Switch Cover method by using Condata tool resulted in state explosion and was unable to generate test sequence. For such cases, a simple pruning of a selected arc right in the Statecharts specification has been implemented and this enables to generate a partial FSM. On one hand, this has a drawback that a complete machine cannot be tested. On the other hand, it enables test case generation methods to deal with the machine. However, Transition Tour method reached much better results covering full graph.

The task of incorporating test sequence generation methods in WEB-PerformCharts allows its use without depending on another tool besides enabling efficiency comparison of different methods. Beyond Transition Tour and Switch Cover, other methods are under development and should be available as cartridges of the system. The main contribution of this paper is to enable a tool to support test process in a distributed environment through development of a web-based tool. Also, the use of XML formatted documents represents an important step bringing another major contribution in standardization of test data. 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. ACKNOWLEDGMENTS

The authors would like to acknowledge the financial grant provided by CNPq/Edital Universal (Project N. 474031/2006-3).

8. REFERENCES

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

[2] Lee, D., and Yannakakis, M. 1996. Principles and Methods of Testing Finite State Machines – A Survey, In: Proceedings of the IEEE, 84(8).

[3] Martins, E., Sabião, S. B., and Ambrósio, A. M. 2000. Condata: a tool for automating specification-based test case generation for communication systems, 33rd Hawaii International Conference on System Sciences.

[4] Myers, G. 1979. The art of software testing, John Wiley & Sons, 1979.

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

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

[7] Bafoutsou, G., and Mentzas, G. 2001. A Comparative Analysis of Web-based Collaborative Systems, Database and Expert Systems Applications, Proceedings of the 12th International Workshop on Database and Expert Systems Applications, pp. 496-500.

[8] Sommerville, I. 2003. Software Engineering, Addison Wesley.

[9] Peterson, J. L. 1981. Petri net theory and modeling of systems, Prentice-Hall International, London.

[10] Ellsberger, I., Hogrefe, D. and Sarma, A. 1997. SDL: Formal Object-oriented Language for Communicating Systems, Prentice Hall Europe.

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

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

[13] Harel, D. 1987. Statecharts: a visual formalism for complex systems, Science of Computer Programming, Vol.8., pp. 237-274.

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

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

[16] Vijaykumar, N. L., Carvalho, S. V., and Abdurahiman, V. 2006. Introducing probabilities in Statecharts to specify reactive systems for Performance Analysis, Computers & Operation Research, 33(8), pp. 2369-2386.

[17] Amaral, A. S. M. S., Veloso, R. R., Vijaykumar, N. L., Francês, C. R. L., and Oliveira, E. 2004. On proposing a Markup Language for Statecharts to be used in Performance Evaluation, International Journal of Computational Intelligence, 1(3), pp. 260-265.

[18] Arantes, A., Vijaykumar, N., Santiago, V., and Carvalho, A. 2008. Automatic Test Case Generation Through a Collaborative Web Application, Accepted for presentation at The IASTED International Conference on Internet and Multimedia Systems and Applications (EuroIMSA 2008), Innsbruck, Austria, 17 – 19 March.

[19] Santiago, V., Mattiello-Francisco, M. F., Costa, R., Silva, W. P., and Ambrosio, A. M. 2007. QSEE Project: An Experience in Outsourcing Software Development for Space Applications, In: The 19th International Conference on Software Engineering & Knowledge Engineering (SEKE), Boston, EUA, p. 51-56.

[20] Santiago, V., Amaral, A. S. M., Vijaykumar, N. L., Mattiello-Francisco, M. F., Martins, E., and Lopes., O. C. 2006. 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, v. 2, p. 183-188.