

---

## Practice

# Testing and maintaining de-localized software systems in a multi-site environment using Web-based tools



Balaji.V<sup>1,\*</sup>,<sup>†</sup> and Sangeetha.B<sup>2</sup>

<sup>1</sup>*Infosys Technologies Limited, Bangalore - 561 229, India*

<sup>2</sup>*AK Aerotek Software Center Pvt. Limited, Bangalore, India*

---

## SUMMARY

Web-based approaches used during software maintenance for testing de-localized software in a multi-site environment are described. The supporting infrastructure uses the Internet for communications and project management practices for procedural direction. Processes involve specifying personnel roles and using a configuration management system for test schedules and the handling of test results. Different testing strategies are used during various stages of the maintenance lifecycle; each has benefits and weaknesses. Test results are automated using scripts, and a Web interface is provided to obtain better tracking. This achieves a 24 × 7 operational effectiveness by using automation and regression testing mechanisms, and results in documented cost savings and software quality improvements. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: multi-site maintenance; multi-site testing, automated testing, 24 × 7 maintenance; Web-based software tools

## 1. INTRODUCTION

The maintenance of software is a vitally important area of software engineering. In the past, software maintenance depended on the original programmer modifying his/her own code [1].

Advances in communications have liberated the Information Technology (IT) industry from conventional spatial constraints. As a result, co-workers no longer need to be physically together, thus making possible a multi-site software maintenance and development environment. For the application software industry engaged in providing customized solutions, the multi-site environment has led

---

\*Correspondence to: Balaji.V, Infosys Technologies Limited, Building No. 7, Electronics City, Hosur Rd, Bangalore - 561 229, India.

<sup>†</sup>E-mail: vbalaji@infy.com



to decentralization of various IT departments, so that now they may span oceans and continents, and be staffed by employees of different organizations [2]. Information flow in such a de-localized environment becomes a key factor in the success of software development and maintenance efforts [3]. The information flow impacts on:

- productivity as developers and maintainers spend time looking for information;
- quality because developers, testers and maintainers need accurate information in order to carry out their tasks effectively; and
- communication between end users and maintainers.

The influence and the power of the Internet have bridged the gap of information flow with regard to global communication [4]. Web-based tools have helped provide better traceability, increased productivity and better quality software in a de-localized environment. Software testers and maintainers use and can rely on Web-based tools to access various sources of information about the system being modified, adapted or changed [5]. These sources can range from artifacts of the development process (requirements documents, design documents, comments within code, test plans and reports, etc.) to user documentation (user manuals and configuration information) to the system itself (both the running system and the source code). Tools used for debugging the code and for analysing the results of execution from a shared simulator help in improving productivity and traceability.

In a multi-site environment, meeting the responsibility for delivering a project involves constant interactions among the multiple locations. Identification of 'primes' or 'process owners,' adopting effective testing methods and tools, setting up a configuration management system covering all sites and integrating the people, processes and technologies for delivering a quality product are the major challenges in a multi-site environment.

The roles and responsibilities are clearly defined at the beginning of the project [6]. Typically these include establishing who is to do a review of requirement specifications, of functional specifications and of design (especially database and software architecture). Other matters defined early include resolving open issues and planning the acceptance testing of the software to be delivered.

This paper first examines the network set-up, processes and practices to be established in the multi-site de-localized mode of operation. Next, various testing mechanisms used in different stages of the software lifecycle are enumerated along with their benefits and weakness. The processes followed before and after the usage of Web tools during different testing stages along with their benefits in enhancing productivity, quality and traceability are discussed. Web-based tools developed in-house, namely Application Message Decoder (AMD) for decoding messages from the Simulator, REsult Submission Tool (REST) for submission of execution results, and Web-Based Issue Tracking System (WITS) for issues tracking, are considered. Finally, the usage of Web-based applications in better project management is elaborated upon before providing some conclusions.

## 2. ENVIRONMENT

### 2.1. De-localization

In a multi-site environment, software maintenance and development work are done predominantly at offshore development centres, with a small team being stationed at the customer site. The offshore team



typically handles most of the analysis, design, programming, coding and corrective and enhanceive testing, while the on-site team at the customer location is responsible for customer interfacing, integration and adaptive testing.

## **2.2. Network set-up, process and practices**

### *2.2.1. Two preconditions*

To work successfully, the multi-site environment model needs a set of preconditions to be in place. The two main preconditions are technological infrastructure and project management.

*Technological infrastructure.* The most crucial resource in the technological infrastructure is a dedicated, high-speed data link via satellite or fiber. Such a link, with an appreciable data transfer bandwidth, acts as a powerful communication medium and facilitates on-line interaction and real-time communication. The various amenities, which can be implemented across such a link, include file transfer, Telnet or remote login to the customer site by the offshore team, telephone hot lines, e-mail and video conferencing. Leveraging these technologies affords immense potential as a means of bridging the physical gap between co-workers.

*Project management.* Project management is considerably complicated by the geographical separation of the parts of the multi-site team. In order to ensure that a project meets its quality and delivery objectives, procedures and processes need to be defined and implemented for various project management activities such as monitoring and control, communication, change management and testing.

## **3. TESTING MECHANISMS**

### **3.1. Types of testing**

Testing in the software maintenance lifecycle plays a critical role in the delivery of a quality product. Testing is essentially verifying and validating the user's change requirements and also ensuring that the maintenance work does result in a change of the existing software to meet the change requirements.

Testing can be classified into corrective testing, adaptive testing and enhanceive testing:

- corrective testing helps in identifying errors introduced during maintenance into the existing software;
- adaptive testing helps in identifying the changes in the environment in which the software must operate;
- enhanceive testing is used in testing the expanding needs of the user (such as new features and enhancements).

Corrective, adaptive and enhanceive testing can be performed during different lifecycle stages by using various testing methods. The different testing mechanisms for doing this are described below.



### 3.2. Manual testing

The manual method is a conventional method of testing, wherein the user develops test plans and test cases for execution for a certain set of requirements. The test cases are run for a set of requirements manually by personnel to determine the pass/fail criteria for a given scenario. Multiple components of software can be simultaneously tested. This manual type of testing can be done independently at different locations. The advantages of using this method of testing are that unit test runs on the computer can be effectively performed, with a reduction of time spent on unit testing. Unit testing can also then be made specific to a certain set of requirements, rather than testing the entire functionality.

Two of the drawbacks of manual testing are that the personnel effort needed for testing is very high, and that results in slower delivery of maintained software to the user. Secondly, manual testing results in monotony and does not provide any new challenges to the personnel involved in the testing. Manual testing is feasible though difficult in a multi-site environment, as the personnel resources in this type of testing cannot be shared easily across different locations. Test results (pass/fail) have to be updated manually and then exchanged between multiple locations, for example by using file transfer protocol (FTP).

### 3.3. Automation testing

Automation testing is an automated method of testing wherein test cases are written in a specific test case language and executed using a simulated environment. The Simulator simulates the functionality of the system under test (SUT) or system under maintenance (SUM). In a multi-site mode of operation, the dedicated network as described earlier provides access to information for all the users on the network. Figure 1 shows an overview of a typical network set-up used in automation testing.

In Figure 1, W3 corresponds to a workstation at the customer site and W1 corresponds to a workstation at the offshore development centre, located say in India. The user logs on to a remote workstation W3 located say in North America or Europe using the Telnet protocol. This enables the personnel in India at W1 to access information located at the remote workstation W3 and *vice versa*. It also creates a virtual workspace and helps in better utilization of available resources.

Automated test cases for execution can be written on the local workstation W1 using a specific test case coding language. These can be transferred to the remote workstation W3 using FTP, for execution. A platform for executing automated test scenarios can be invoked from the remote workstation W3. This platform is able to compile, link and generate an executable file from the automated test case. The executable file can be executed through the platform, which will be connected to the Simulator through a X.25 link. The Simulator simulates the software to be tested, by providing a virtual environment. The results of the simulation are returned through the existing link to the platform invoking the test cases and stored on workstation W3. The results of execution can be retrieved from W3 for further analysis on workstation W1, again using FTP.

Figure 2 specifies the directory structure to be followed to ensure better traceability. The directory structure specifies the procedure for storing and accessing automated test scenarios and related documents. This structure is maintained both at the local (W1) and remote (W3) locations. The server is organized into different directories.

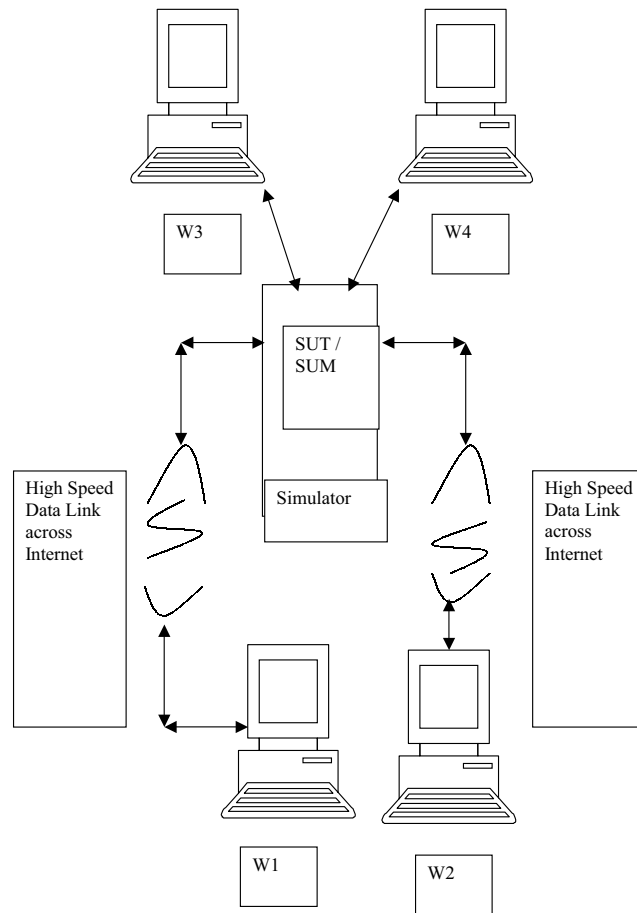


Figure 1. Overview of a typical network set-up used in automation testing.

- Common Area—This area is accessible by all users. Read only access is provided and write access is available only for the concerned point of contact. This directory contains reference documents supporting the source code.
- User Area—Each user is given a dedicated directory for running tests and doing other work. The user stores the test scenarios developed by him/her there. Each test scenario is identified by a unique test case identification (ID). All the executable files are stored for execution through the Simulator. Results from the Simulator are stored in report files for further analysis. Failed test scenarios are debugged immediately and re-executed until test cases run successfully.

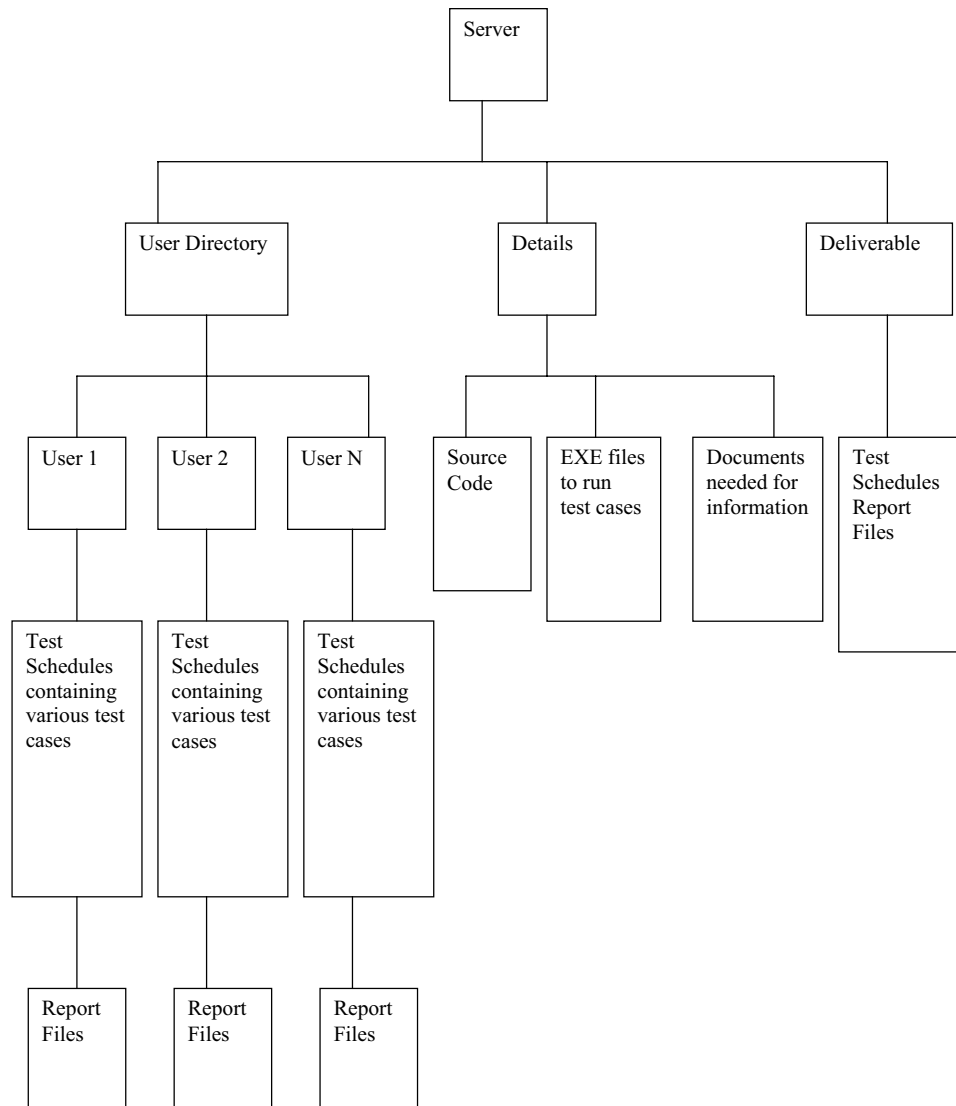


Figure 2. The directory structure used to ensure good traceability.



- Deliverable Area—Each user submits the successful test case scenarios into this area. The report of execution is also submitted. These are versioned for better tracking. A checklist is maintained in this area, which is to ensure that the process is followed correctly.
- Delivered Area—The individual points of contact have access to this area, which contains versioned test scenarios and checklists. This area is delivered to the customer for facilitating acceptance. The versioned test cases and test plans are stored in a central library system for future use.

A single point of contact at each location (such as at W1 and W3) is responsible for coordinating the testing process and ensuring better project management. Communication between the contacts is on a daily basis through e-mails and phone calls using the existing network. It is ensured that the offshore development team works only during daytime (night time at customer location) to effectively utilize the resource (SUT/SUM) available at the customer site. This becomes possible on account of differences in time zones between the customer site and offshore centre.

Automation testing in a multi-site environment provides benefits in that the single resource, viz. Simulator, is effectively used round the clock on account of time differences between the two locations. This results in  $24 \times 7$  operational effectiveness. Adaptive testing can be done efficiently. Enhance testing of the addition of new functionality can be done quickly, resulting in a near immediate availability of results. Automating test scenarios helps in identifying bugs in code during the early stages of prototyping and design of an enhancement. Bugs can be fixed immediately before the delivery of the source code or before acceptance testing.

Drawbacks of automation testing are that it is dependent on the availability of the link for execution of test cases. When the link between the onshore and offshore sites is down, test scenarios cannot be executed with reasonable promptness.

### 3.4. Regression testing

In the multi-site environment, regression testing is performed for testing the entire software product by executing the automated test cases using a batch file. The batch file used consists of test scenarios and error scenarios involving multiple functions and can be executed on the Simulator automatically over the weekend. An iteration count can be specified for the execution of the failed scenarios during regression. The test case will be executed the number of times specified by the iteration and helps in re-execution. The iteration count is usually set to two, in order to achieve better efficiency in the usage of the SUT/SUM. The results of the regression testing are made available on the completion of the software maintenance work. Keeping the results of the regression testing helps in analysing and taking preventive measures to ensure proper functioning of the software.

The directory structure followed in this method consists of a regression suite directory for the various regression packages and a report directory for storing the results of execution. A regression suite consists of test scenarios from different functional components integrated into a single package. The test scenarios are selected from different functional components to test the entire modified product being delivered. The Product test team identifies the test cases to be included in the regression package. A team partly from the customer site and partly from the offshore location then reviews this package. Schedules for execution are also prepared and distributed on a need to know basis.



Regression testing becomes beneficial when performed from the customer location, as the connectivity through the link to the Simulator will be available without any issue. This might not be the case if the execution is done from offshore, which is dependent on the connectivity link.

Regression testing is also beneficial in testing and maintaining legacy software systems, as the regression testing can be done at different checkpoints in the maintenance lifecycle to ensure the existing functionality is not affected. Appending new test scenarios to the batch file is easy and can be done as and when needed.

The drawbacks of regression testing are that the size of the batch file can become huge depending on the functionality to be tested. Batch file executions can take from a few hours to a few days, thereby resulting in non-availability of the Simulator for other testing purposes. Results become available only after the execution of the entire batch and typically the report is also large.

### **3.5. Porting**

Test case porting is a mechanism by which test scenarios written for one SUT are modified for testing another SUT/SUM. As an example, test cases developed for a North American customer can be modified to suit the requirements of a European customer. Porting reduces the effort needed to create new test scenarios and helps in reuse. Porting can be achieved efficiently by the use of scripting languages like PERL, which help in conversion to meet the new requirements. Ported test scenarios can be executed on the SUT/SUM specified, and the process followed will be similar to the one described for automation testing. Porting can also be beneficial in converting ageing legacy systems applications to new Web-based applications with programmable interfaces.

## **4. WEB-BASED APPLICATIONS AND TOOLS**

### **4.1. Three Web-based tools**

The advances in the Internet and in its accessibility have resulted in the use of Web-based applications and Web-based tools for better project management. Web-based applications and scripting languages help in collating for easy access the results received from the Simulator and data from other sources. Three of the Web-based tools—AMD, REST and WITS—have improved project tracking and status reporting. Since no commercially available tools could be found to fit our project needs, we developed the tools in house.

### **4.2. Application message decoding tool**

The AMD tool helps in decoding messages received from the Simulator. The messages received from the Simulator are produced in hexadecimal code. These messages are converted using a script and the output is displayed in natural language (Text) using a Web interface for easy readability and faster debugging of failed test scenarios. This helps in identifying errors quickly and helps in resolving issues with the existing software build within the stipulated time frame. The messages received are primarily Simulator outputs, received from testing an existing or a new or changed requirement. The Simulator simulates the functionality of the SUM/SUT. The tool can be customized for decoding messages arising





## **IN Codec**

Select IN Variant

09 81 03 0E 12 0B 52 22 00 12 04 16 14 18 10  
00 00 04 43 48 00 0C A9 62 81 A6 48 04 00 00  
01 00 32 01 6C 7E A1 7C 02 01 00 02 01 00 30  
74 80 01 00 83 08 84 13 16 14 21 61 30 00 85

Decode

Clear

Message:

```
SS7_IE_NO_TAG( 103)
|--SCCP_IE_VARIANT(3020) = 01
|--SCCP_IE_MSG_TYPE(3001) = SCCP_UDT(09)
|--SCCP_IE_PROTCL_CL(3007) = 81
|--SCCP_IE_CLD_ADDR      (3005) = 5222001204161418100000
|--SCCP_IE_CLG_ADDR(3006) = 4348000c
+-TCAP_BEGIN(8001)
|  |--TCAP_IE_OTID(8005) = 0000b012
|  |--TCAP_IE_DIALOGUE_PORTION(8007)
|  |  +-TCAP_IE_EXTERNAL_TAG(8028)
|  |  |  |--TCAP_IE_OBJECT_IDENTIFIER(8042) =00118605010101
|  |  |  |  +-TCAP_IE_ASN1_TYPE(8029)
|  |  |  |  |  +-TCAP_IE_DIALOGUE_REQUEST(8031)
|  |  |  |  |  |--TCAP_IE_PROTOCOL_VERSION(8034) = 0780
|  |  |  |  |  +-TCAP_IE_APPLICATION_CONTEXT_NAME(8035)
|  |  |  |  *--TCAP_IE_OBJECT_IDENTIFIER(8042) =
04000001003201
+-TCAP_IE_COMPONENT_PORTION(8008)
+--TCAP_IE_INVOKE_COMPONENT(8010)
|  |--TCAP_IE_INVOKE_ID(8041) = 00
|  |--TCAP_IE_LOCAL_OPERATION_CODE(8018) = 00
+-CAP_IE_INITIAL_DP_ARG(14005)
|  |--CAP_IE_SERVICE_KEY(14061) = 00
|  |--CAP_IE_EXT_BASIC_SERVICE_CODE(14030)
|  |  *--CAP_IE_EXT_TELESERVICE(14032) = 11
|  |--CAP_IE_CALL_REFERENCE_NUMBER(14018) = c2
|  |--CAP_IE_MSC_ADDRESS(14046) = 91161408090000
|  |--CAP_IE_CALLED_PARTY_BCD_NUMBER(14020) = a11614305100f6
*--CAP_IE_TIME_AND_TIMEZONE(14098) = 0210607010913200

TTF_OK
```

Figure 3. An example of the input to and output from the AMD tool.



out of different standards (American, European, etc.). It is also useful in building prototype message formats early in the software maintenance cycle. Shown in Figure 3 are the results of output from the Simulator in hexadecimal, and a sample decoded message in natural language (Text) form using the AMD tool. As can be seen from Figure 3, provision exists to select some variant types based on the different standards.

#### 4.3. Result submission tool

The REST is used to submit the results of execution into a common library. By using a Web interface, the results of different regression testing packages can be viewed across multiple locations. Pull down menus help in filtering data needed for specific builds and different customers. A snapshot of the status is obtained which is controlled through a user authentication. This prevents the results of an execution specific to any certain customer from being accessed by unauthorized personnel. Figure 4 shows a view of the REST tool. Mandatory fields, viz. Test Case Identifier, Week of Regression execution, Load used, System Tested (SUT/SUM) and status of execution (Pass/Fail), are pre-formatted and uploaded and inserted into the form. Pull down menus help in achieving a better collation of the data and help in providing better communication about project status between various locations.

#### 4.4. Web-based issue tracking system

WITS is used for tracking issues pertaining to the execution of test cases for systems being maintained (SUT/SUM). Each issue is uniquely identified by an issue ID. The issues are created by the single point of contact personnel (see Section 1) and assigned to the person working on the issue. The issues are tracked by different states. During the creation phase by the single point contact, the issue is at an OPEN state. When the person assigned starts working on the issue, the state is modified to WORKING. REVIEW state is reached when the issue is resolved and a review is being held. The results of the review will be entered into the tool during this phase and the state changed to working. The moderator of the review screens and assesses the changes suggested during the review and ensures that the issue is resolved before actual submission of the software into the deliverable directory. The state is changed to SOLVED to indicate that the issue has been resolved [7].

The tool also has fields to identify the complexity, target date for resolution, version history, problem description, estimated effort in hours, actual effort, defects and comments. The tool filters help in customizing the fields for extracting data from the issues and give a quick snapshot of open, working, reviewed and solved issues. An advanced search mechanism helps in querying issues assigned to a certain individual, issues raised during a specified time frame, issues based on complexity and so on. The status of these issues is identified using colour codes (Red, Yellow and Green) for better visibility and state of healthiness of the issue. Red indicates that state of the issue has exceeded the target date for resolution and needs to be reforecast with appropriate details for slippage. Yellow indicates that the forecast date is approaching and the issue is to be resolved quickly. Green indicates that the issue is in a healthy state [8].

The access to WITS is through the Internet. Internet availability of the WITS accessible data helps in leveraging the information and results in quicker and better delivery of the modified product. Figure 5 provides a short example of WITS.



**WELCOME TO  
REST**  
REsult Submission Tool

<u>S</u> UBMIT	Q <u>U</u> ERY THE
----------------	--------------------

**REST allows to track our Regression Re-Test Results.**  
***From the Main Menu you can Submit Results and Query on the Submitted Results in the Database.***

**SUBMIT A RESULT**

**Notes :**

1. ***'TCID', 'Regression Week', 'Load Name', 'Switch Name', 'Result' are compulsory fields. Form cannot be submitted without filling these fields.***










 Regression Week : <input type="text"/>	 Load Name : <input type="text" value="NA013"/>	 Switch Name : <input type="text"/>	 Package : <input type="text"/>		
 TCID : <input type="text"/>	 Result : <input type="text"/>	 Type Of SR : <input type="text"/>	 SR Number : <input type="text"/>		
 Comments If Any : <input type="text"/>					
<table border="1"><tr><td><u>S</u>ave The Result</td><td>C<u>L</u>ear The <u>R</u>esult</td></tr></table>				<u>S</u> ave The Result	C <u>L</u> ear The <u>R</u> esult
<u>S</u> ave The Result	C <u>L</u> ear The <u>R</u> esult				

Figure 4. A view of the REST tool.



Request Id	Assigned To	Request date	Details of Issue	Forecast Date
183	Sri	18.9.00	Identify correct functional requirement	30.9.00
158	Sai	22.8.00	Verify report status	28.8.00
109	Ram	15.7.00	Update test schedule	22.7.00

Figure 5. A brief sample of WITS output.

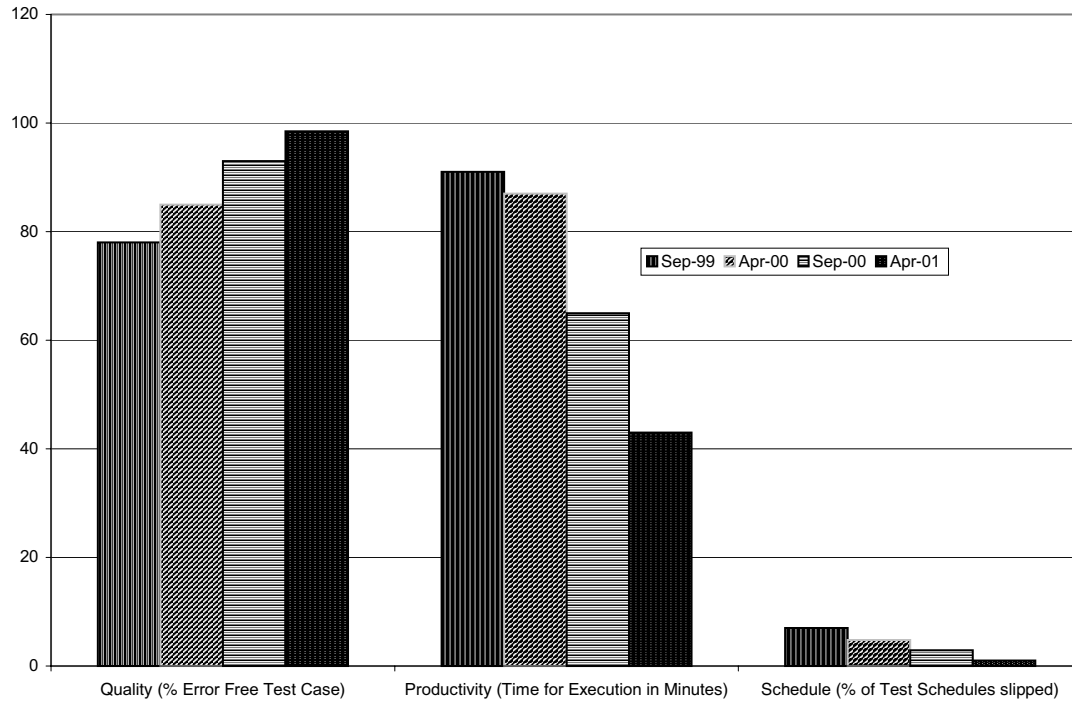


Figure 6. Improvement in test quality, productivity and reduction in schedule slippages.

#### 4.5. Experience with the tools

A comparative study was conducted on a project on the usage of the three above-mentioned Web tools. The study phase lasted over four consecutive maintenance releases of the same system, each of which included product testing. Data collected included the impact on quality (error-free test cases delivered), productivity as a factor of time (time taken for test execution in minutes) and percentage of schedule slippages. Data collected also included the impact on communications and project management.

Pre-tool use data were collected in September 1999. These data showed the quality (78% of error-free test cases delivered), productivity (91 minutes for execution) and schedule slippages with respect to forecast delivery dates (7% of test schedules slipped) before using the Web-based tools. The tools were introduced at the end of 1999 and the data were again collected during April 2000 during the next maintenance cycle. A marginal increase in quality (85% of error-free test cases delivered), productivity (87 minutes for execution) and schedule slippages with respect to forecast delivery dates (5.8% of test schedules slipped) was observed. Data were collected during the third maintenance cycle in September of 2000. Finally data were collected in April 2001, at the end of the fourth cycle. Those data revealed drastic benefits in quality (98.5% of error-free test cases delivered), productivity (87 minutes for execution) and schedule slippages with respect to forecast delivery dates (1% of test



schedules slipped). The benefits seen over the four-release cycle are 26.28% improvement in quality (error-free test cases delivered), 52.7% reduction in execution time, thereby increasing productivity, and an 85.7% reduction in schedule slippages. Figure 6 shows the improvement in quality, productivity and reduction in schedule slippages.

Data collected on the impact on project management showed that the project management effort in the testing phases of maintenance was reduced by 33%. Time spent in communication between different change agents was drastically reduced. Also, dramatic reductions in paper work resulted in cost savings to the organization in carrying out the testing during maintenance projects.

## 5. CONCLUSION

In the multi-site environment for carrying out software maintenance work, the usage of Web-based tools for decoding the Simulator test outputs, for results submission and for issue tracking has resulted in better monitoring and control, communications and improved project management across multiple locations. Streamlining the work process and setting up a well-defined set of procedures has resulted in increased productivity and quality, and helped in reducing schedule slippages as verified by a before-and-after study. The concept of 'flexi-place' or 'telework' has thus been made possible, and is contributing to providing  $24 \times 7$  operational effectiveness for testing during maintenance.

The lessons learned in using Web-based tools in the maintenance of the software in a multi-site environment are:

- Creating a well-defined process for communication, and identifying roles and responsibilities between multiple locations have resulted in improved project management skills and better service delivery to the customers.
- Using appropriate tools and providing an access mechanism over the Internet have reduced the gaps in information flow between locations.
- Archiving and retrieving of data by using scripting techniques, tracking issues from initial stages to the closure state, using color coding techniques to identify the healthiness state of project issues, have all helped in getting better traceability of and quality in the software delivered.
- Sharing of hardware resources between multiple sites has resulted in cost saving and also provided a mechanism to achieve  $24 \times 7$  operational effectiveness.

## ACKNOWLEDGEMENTS

We thank the organizations involved for providing an opportunity to present our experiences in testing during software maintenance. We also thank the *Journal's* reviewers for their valuable comments. Above all we thank the ALMIGHTY for having given us the strength to finish this paper in a short duration.

## REFERENCES

1. Hayes JH, Offutt AJ. Product and process: Key areas worthy of software maintainability empirical study. *International Workshop on Empirical Studies of Software Maintenance (WESS 2000)*. IEEE Computer Society Press: Los Alamitos CA, 2000; 3 pp. <http://hometown.aol.com/geshome/wess2000/janeHwess.pdf> [27 March 2002].



2. Gopalakrishnan S, Kochikar VP, Yegneshwar S. The offshore model for software development: The Infosys experience. *Proceedings of the 1996 ACM CPR/SIGMIS Conference*. ACM Press: New York, 1996; 392–393.
3. Seaman CB. Unexpected benefits of an experience repository for maintenance researchers. *International Workshop on Empirical Studies of Software Maintenance (WESS 2000)*. IEEE Computer Society Press: Los Alamitos CA, 2000; 4 pp. <http://hometown.aol.com/geshome/wess2000/Cseamanwess2000.pdf> [27 March 2002].
4. Cartwright M. Empirical perspectives on maintaining web systems: A short review. *International Workshop on Empirical Studies of Software Maintenance (WESS 2000)*. IEEE Computer Society Press: Los Alamitos CA, 2000; 5 pp. <http://hometown.aol.com/geshome/wess2000/Cartwright.pdf> [27 March 2002].
5. Kit E, Finzi S. *Software Testing in the Real World: Improving the Process*. ACM Press: New York, 1995; 252 pp.
6. Kumar MP, Das VSR, Netaji N. Offshore software maintenance methodology. *Journal of Software Maintenance* 1996; 8(3):179–197.
7. Kajko-Mattsson M, Forssander S, Andersson G. Software problem reporting and resolution process at ABB Robotics AB: State of practice. *Journal of Software Maintenance* 2000; 12(5):255–285.
8. Balaji.V. Overcoming multi site development challenges in effective technology management. *Presentations at the InDOORS Europe 2000 Conference on Enterprise-Wide Requirements Management*. Telelogic AB: Malmö, Sweden, 2000; 7 pp. <http://www.telelogic.com/news/usergroup/eu2000/presentations.cfm> [27 March 2002].

#### AUTHORS' BIOGRAPHIES



**Balaji.V** is a Project Manager with Infosys Technologies Ltd, an SEI CMM Level 5 software services and consulting firm with headquarters in Bangalore, India. He has been involved in all phases of a system's lifecycle and has contributed to the development, maintenance, testing and knowledge management of Advanced Intelligent Network features for a leading North American Telecommunications manufacturer over the last 6 years. He holds a Bachelor Degree in Electronics Engineering from the Regional Engineering College (RIT) Jamshedpur.



**Sangeetha.B** is a Satellite Team Leader with AK Aerotek Ltd, India. She has been involved in testing and configuration management of real-time embedded systems for a leading European aeronautical firm. She holds a Bachelor Degree in Computer Science Engineering from the University of Madras.