

Building An Effective Testing Team

Suvrata Acharya
Kumud Iyer

Abstract

This case study presents the experience of a project, which follows some innovative practices to build and strengthen the testing team. The intent is to share the experiences with the industry on how to build an effective testing team for critical projects and also provide them sufficient level of challenges to keep them motivated. This paper does not narrate the standard testing process followed in the project, but will focus on all the innovative practices followed by the testing team in order to improve testing efficiency and make the testing process more interesting.

This project is executed for a leading provider of asset management and investment management solutions to financial institutions and corporations. The scope of the project includes development, maintenance and support of CICS based business components. Each of these business components performs a specific business operation or financial transaction. The run time environment for these components is Mainframe, CICS and Adabas. These components are accessed by different UI applications through SNA/TCPIP connectivity using a standard protocol. The development environment is Mainframe SASC. Since the data generated or processed are highly critical to the business users, a very comprehensive testing is done to ensure the sanctity of each component. Recently, when NIIT was being assessed as CMMI Level 5, this project was one of the key projects that went through the assessment process. This paper tries to highlight the various factors that have helped in achieving this objective, which are:

- Testing Team composition
- Setting up objectives for testing team
- Test process Standardization
- Test automation
- Creating learning opportunities
- Key success Indicators.

Testing Team

The average development life cycle of each of these components is between 3 to 4 months. Since there are multiple components being developed simultaneously, the average team size is 2, which includes 1 developer and 1 tester. Also, the business functionalities in these components are not similar in nature, as every component is developed to meet the requirement of one financial transaction. Testing of these applications involves verification of data or information generated by each component under different business scenarios. The project has an independent testing team led by one Quality Assurance Manager, who coordinates the testing activities for the entire project. Each member of the testing team is motivated to play a role more than just a Tester. Apart from testing, the tester also works on process improvements, automations and plays a role of business expert within the project. The core competency required to test this application is the understanding of the business functionality of the components under test. In order to create more scope to acquire new business knowledge the testing team is consistently working towards improving the testing productivity. This enables them to play the role of Business Analysts as well, leading to an opportunity for growth of the individuals.

One of the major challenges faced by this team is to create standardized processes and automated testing tools to test in mainframe testing environment. Since, there is no tool available in the market to test the application on the desired environment, the team has taken the initiative to build the tools and work towards consistent improvements. One of the many driving factors to these initiatives is the need for the reduction of manual work and the rework during the testing life cycle.

Setting Up the Objective

One of the main activities to set the team going is to have clearly defined objectives with goals and measures clearly specified. The primary focus of the testing team is to ensure that the components are delivered with the best of quality. The main objectives defined for the team is:

- Meeting the deadline
- Reduction of testing cycle time
- Quality of deliverables
- Self learning
- Continuous improvement through Innovation

In order to meet these objectives the major challenges that the team faces are:

- Huge data verification
- Business knowledge required to create effective scenarios
- Retention of the knowledge
- Sustaining the defined quality levels of the deliverables
- Reduction in defect leakage

In a very systematic manner, the testing team has been working on some highly effective mechanisms to meet these challenges. Some of the major initiatives taken as a part of Innovation objective, are

highlighted below:

1. Development of an automated online testing tool named as SwayamParikshak (SP).
2. Creation of processes and templates to help the tester to perform independent analysis before making the test cases.
3. Introduction of a new process, "UT-IT handshake", to ensure minimum defect leakage from previous phases.
4. Development of an automated mainframe based testing tool known as Mainframe Automated Testing Tool (MATT) that does the testing in batch.
5. Development of a Test case repository called as "TestingSuite" 6.
6. Development of an Automated Review Tool called as "TestingAid"
7. Development of an Automated Testing Tool to test the mainframebased components, which use VSAM as the backend.

Apart from the above initiatives, there is a consistent effort to take various initiatives for improvement in the existing processes, templates and tools.

Process Standardization

This section will explain how the testing process is standardized in the project. The Testing process in the project comprises of four major steps:

Tester Analysis:

The objective is to maximize the understanding of the requirements by the tester. This analysis is independent of the analysis already done by the developer, hence helps the tester to create unbiased testing scenarios. As a part of this process, the tester breaks the requirements into various sub-components and fills up a template that is specially designed to organize the requirement gathering process. Moreover this process helps in the reduction of effort for scenarios identification and data preparation. This template contains separate sections for validation of input fields, possible dependency of input/output fields and database dependency. (Ref. Appendix E, for the template of Tester Analysis Document).

Test Cases Identification:

The input to this step is the tester analysis document. In this step, the tester identifies the required scenarios and makes a test plan. A test plan is prepared using an excel spreadsheet template which has different sections, one each for testing strategy and different types of testing done in the project. A test case in the test plan contains the information regarding the description of the testing scenario, the traceability to the requirement, test data, data required for environment setup, expected results etc. Once the test plan is prepared, it is reviewed against that design document prepared by the tester in order to validate both design and test plan. This test plan forms input to the automated testing tool that executes the

test cases directly from the test plan.

Testing Prep:

To understand the need for this particular step, one has to understand the test execution process first. The test execution process consists of 3 steps:

a. **Preparing the environment:** This step comprises of setting data in the test database so that the desired environment to execute a particular test case is setup. This activity is required to be done during the run time as the test cases are independent of each other and each requires a different type of data in the database. These set ups are pre-recorded QAHyperstation scripts which get executed automatically by the testing tools.

b. **Execute the test case:** Once the data is setup, the testing tool, as per the user's instructions, which are provided as a sequence file to the tool before it's execution, executes the test cases. The tool can execute any number of test cases without requiring any user intervention during the execution period.

c. **Verification of results:** Once the execution of a particular case is over, the testing tool verifies the output with the expected output. Expected output can be in the form of output field values or messages generated by the component, or it can be the data created/updated by the component in the database.

So, as part of the testing prep, the tester basically prepares all the inputs for the testing tool. QAHyperstation tool available on the Mainframe is used to record the setup scripts for each test case. Also, the tester prepares baseline dumps (database snapshots) for each test case with the help of the testing tools. Tester fills up the expected output field values in the casebook.

Testing:

Testing can be the most time consuming phase depending on the number of rounds of testing. The testing cycle time not only includes the execution time but also the time taken by the developer to fix the problem once identified. Although the testing execution time has reduced substantially after the introduction of automated tools, but the tester's involvement is also required to ensure that developer has delivered a good quality code before the testing starts. One of the key testing processes defined in the project is the tester's involvement in code review, unit test plan review and unit test results review. This ensures the minimization of defect leakage before the testing starts. Moreover, before the testing begins, there is a defined handshake process, which checks for process adherence in all the phases before testing phase. Once the UT-IT handshake is complete, tester starts the test cases execution using the testing tool on the basis of the testing prep. Any defect found during the testing is logged and tracked till closure. The stop criteria for testing is successful execution of all the cases in the casebook. Open issues, if any, are documented in the test plan with proper explanation for future reference. All the

test results and documentations are delivered to the client for reference during the acceptance testing.

Test Automation

This section will explain briefly about the testing tools that were developed within the project, to automate the testing execution process. All these tools are intended to test in a mainframe environment. The following sections explain the features of each of these tools.

SwayamParikshak (SP):

SP is an integrated testing tool that automates the complete process of testing of mainframe based middleware components online. It interacts with the mainframe using VB Automation scripts and SNA connectivity, thereby eliminates user intervention for performing testing tasks. Minimal intervention is required in terms of setting up automation script information files that contain data required to carry out the specific tasks. This is an in-house testing tool developed to take input from the test case book, run setup scripts & test cases and then take database dumps on the tester's local machine. It also compares the new test results with the corresponding existing/ previous test results and the Database Dumps for the positive test cases with their corresponding baseline dumps respectively (Detailed features of the tool in appendix-A)

Mainframe Automated Testing Tool (MATT):

The objective of this tool is to provide an automated platform for testing to be done in batch mode. This tool was developed to further improve the test execution time. One limitation of SP is its dependence on the network link between user's machine and mainframe. As a result if the link goes down or gets slow then it requires user intervention, to remove the link dependency, MATT was developed which reduced the test execution time by approximately 60%. Once the tester prepares the test cases, this tool executes all the cases in a batch mode and generates the comparison results. MATT has following basic features (Details in Appendix B):

- Automated execution of cases on CICS without involving SNA
- Automated execution of QAHyperStation scripts through JCL
- It takes dumps directly on Mainframe datasets.
- Automated result comparison using SUPER

VSAM Testing Tool:

SP and MATT has the ability to take dumps from the mainframe ADABAS. There are customized query transactions written by the project to read the ADABAS files explicitly. Recently there was a requirement to write new components that could use VSAM as its back end, for this the testing team used the features of SP and Microsoft ADO to develop an automated tool in a limited time. This tool is designed explicitly to conduct automated testing for components with back end as VSAM. (Details of the tool in Appendix- D)

TestingAid – Automated Deliverable Review Tool:

The objective of this tool is to verify all the test results and test documentation before final delivery. Since, the size of the deliverables is very large, manual review is a time consuming process, TestingAid was developed to reduce the review effort substantially by automating the review process, and has been successful in achieving the same.

Creating Learning Opportunities

As stated earlier, knowledge is a key requirement to test these transactions. Automation helps the testers to get rid of the manual/mundane work, but strong business knowledge helps the individual to improve his/her own productivity. For a long term project like this it's not only important to train the new team members with a proper induction plan, but also it is required to give each individual continuous learning opportunities so that he/she can add more value to the project. Automation helps the testers to save time so that they can focus more on the knowledge; needless to say it is very critical for the project to retain that knowledge as well. The project has adopted following mechanisms to create and retain the knowledge.

Learn Plan:

For every individual in the project, a learn plan has been defined to encourage new learning. A mentor is assigned to every individual, who helps the individual to acquire new business and technical knowledge. Strong business knowledge helps the testers to identify effective scenarios for the testing and the technical knowledge helps the testers to look towards automating the work otherwise done manually. Moreover, learning is one of the key motivating factors for the testers to enjoy their work continuously.

Knowledge Sharing Sessions (KSAK):

Once a person acquires some new knowledge, it is important for that person to share the acquired knowledge with the entire team. Regular KSAK (Knowledge Sharing Among Knowledgeable) sessions are conducted to encourage the individuals to share their knowledge. This not only helps the group to be benefited from an individual's learning, but also opens up new learning directions for the others

Knowledge Portal:

The project uses a tool, known as Knowledge Portal, to store the knowledge in a structured manner. This tool, developed by NIIT, has not only a knowledge base, but also facilitates creation of knowledge dynamically. The team is encouraged to use the knowledge portal instead of e-mail for their query resolutions so that knowledge can be retained in a single place for the future reference. Individual learning plans and knowledge sharing are made using this tool, so that it can be retained for the future reference. The knowledge base is used as a one-point reference for all the activities being performed in the project. It contains all the

processes, business knowledge and technical knowledge being used in the project.

Testing Suite – A Test Case Repository:

The scope of the project is to develop new components and also maintain all the existing components. During the maintenance phase, it is ensured that proper regression testing is done before the change is delivered. For regression testing, the tester identifies the test cases to test the modules that may get impacted as a result of the change, to ensure the sanctity of the existing functionality. This process takes a lot of time, especially when the module is commonly used by multiple components. To save time during identification of test cases, the team came up with a concept of a test case repository, which is known as Testing Suite.

The Testing Suite helps in identification of scenarios and data for regression testing on the basis of business functionality. A repository of full path test cases for each component is maintained for no-harm testing of component during each maintenance release. These cases / scenarios are grouped on the basis of the functionality that they test. Once, the tester selects the cases to be tested, the tool prepares the necessary files that can become input to the Automated testing Tool. Regression testing effort is saved up to 90% by using this tool. (Refer to appendix C for tool details)

Key Success Indicators

This section will share some of the major benefits that came out of these initiatives. The team has incurred both tangible as well as intangible benefits, as detailed below:

Tangible Benefits:

1.Reduction of UAT Defects: The number of UAT defects has been reduced significantly over the last few years. The following graph (Figure 1) gives a view of the UAT defects over last 4 years.

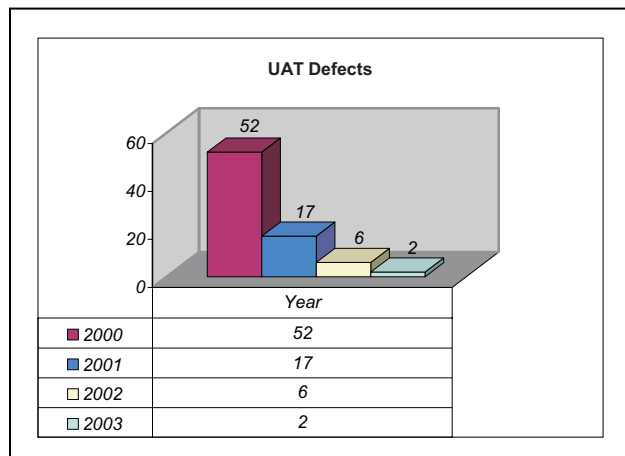


Figure 1

2.Reduction of Testing Cycle Time: Introduction of tools and processes to prevent defects has helped the project to reduce testing cycle time significantly. The following diagram shows the percentage of testing effort for major development projects over the last 4 years.

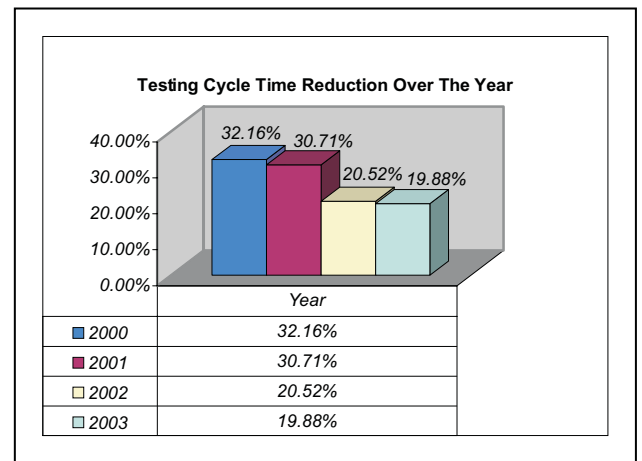


Figure 2

3.Continuous Improvement in Test Execution Time: MATT was introduced this year after successful implementation of SP. When the performance of MATT in terms of test execution time was measured against SP, it was found that MATT takes an average 68% less time than SP.

4.Improvement in Regression Testing Effort: After the introduction of Testing Suite, the regression testing effort was reduced to around 90%.

5.Reduction of IT Defects: IT defects reduced significantly because of the processes like UT-IT handshake process thus reducing the testing time further.

Intangible Benefits:

1.Tester mainly focuses on the test cases and business knowledge rather than on execution.

2.Better focus on the quality – Tester can test more test scenarios in the given time.

3.Improved team morale as less manual execution.

4.Testers got empowerment to focus on overall processes in the project, not just testing.

5.Testers demonstrate their technical and innovative skills, thus bring a lot of motivation within them.

6.More scope for testers to increase their domain knowledge helps them to see more growth aspect.

Appendix A – SP

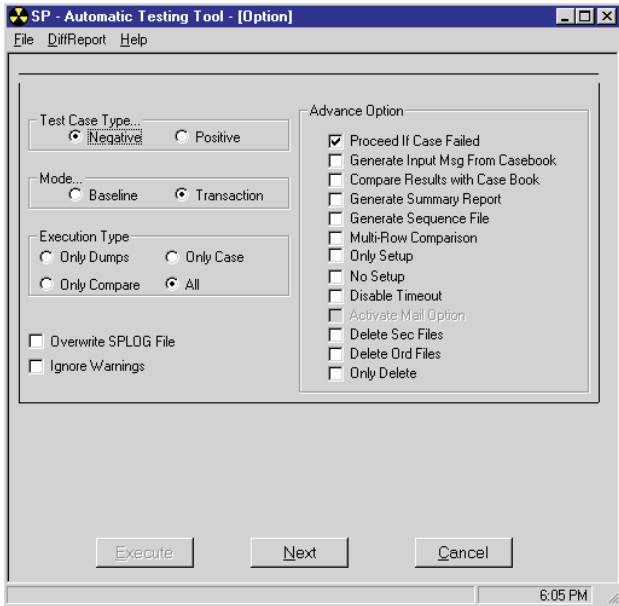


Figure 3

Features:

- This tool is installed on the user's machine. It interacts with the CICS region with SNA connectivity.
- SP is completely customized to test middleware components.
- The basic input to the tool is the IT plan and transaction specific information like name and type.
- The tool automatically generates the input message(s) from the IT plan.
- It executes QAHyperstation set up scripts using VB Automation scripts.
- It then executes all the cases in an automated manner.
- After execution, the tool takes ADABAS dumps on the local machine in a particular format for each of these cases (both baseline and actual).
- The results are then compared (i.e. output fields and error/warning messages) with the expected results in the IT plan.
- Compares ADABAS dumps to find out any problem in file update.
- Make a test report after consolidating all the identified differences.

Appendix B – MATT

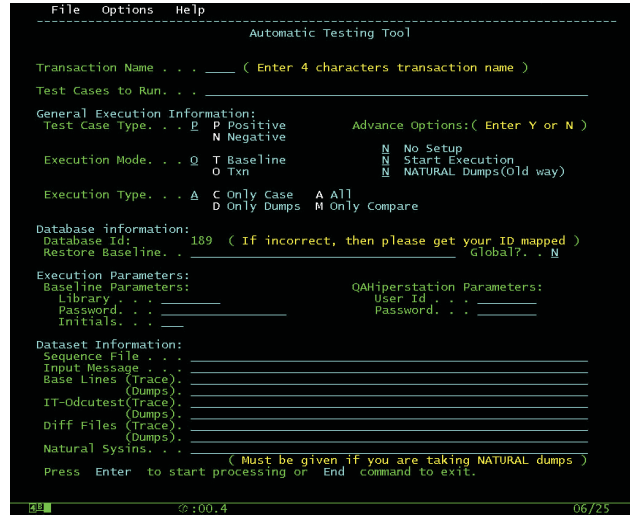


Figure 4

Features:

- Before execution of the tool, following things must be available on mainframe datasets:
 - Transaction input messages.
 - A sequence file describing the sequence of execution.
- A panel accepts all the input information from the user. The panel information can be saved for future use.
- The tool calls appropriate program to construct one or more JCLs depending on the number of input messages to be executed. By default, JCLs will be submitted automatically.
- QAHyperstation scripts are constructed dynamically on the basis of user inputs.
- Each JCL has the steps to execute the test cases, then it takes required database dumps and compares the results.
- The Dump program formulates the key values automatically on the basis of the input in the input message before taking the dump of a specific file.
- Comparison of results is done using SUPERC.

Appendix C – Testing Suite

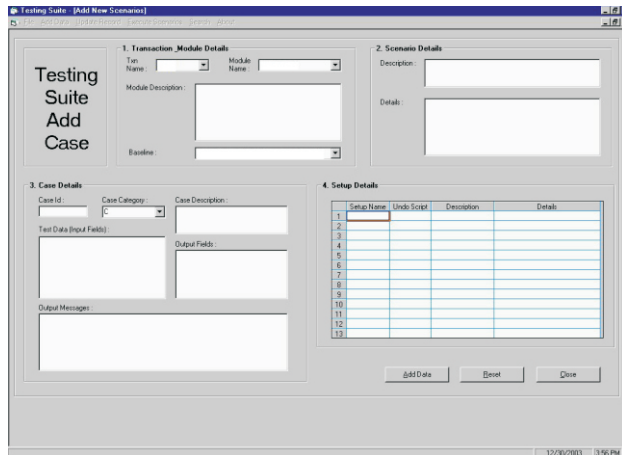


Figure 5

Features:

- Input Test Case data in database – Online and Batch Mode: Test cases and Test Data can be uploaded on the repository either online or in a batch mode using a excel spreadsheet.
- Update Data: One can update the data of a specific test case at any point of time.
- Locate test data on basis of various search criteria: Tester can search a test case using different test criteria. Some of the criteria, which are very frequently used, are module name, functionality description, key word search etc.
- Make Transaction folders with input messages and execution sequence files for SP. Testing Suite creates all the inputs that are required by SP to execute the test case.

Appendix D – VSAM Testing Tool

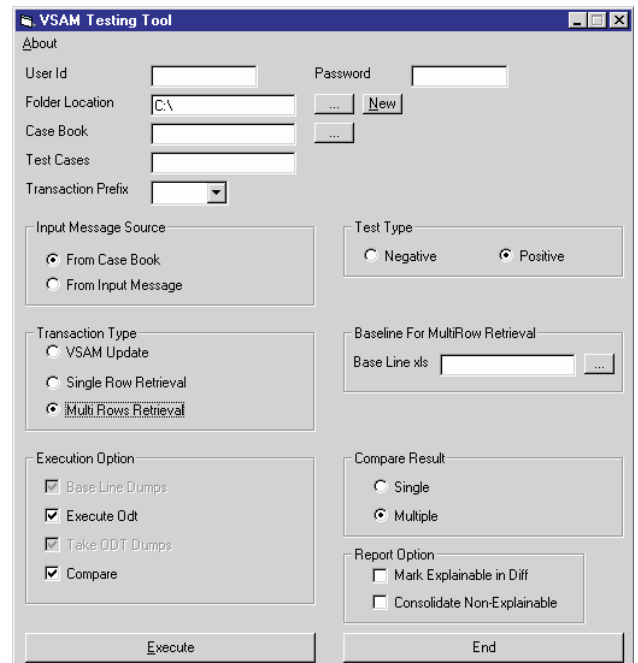


Figure 6

Features:

- The basic input to the tool is the IT plan and transaction specific information like name and type.
- The tool automatically generates the input message from the IT plan.
- It then executes all the test cases in an automated manner.
- After execution, the tool takes VSAM dumps on the local machine in a particular format using ADO (Activex Data Object) for each of these cases (both baseline and actual)
- The results are then compared (i.e. output fields and error/warning messages) with the expected results in the IT plan.
- Compares VSAM dumps to find out if the new application updates VSAM in a different way.
- It makes a report after consolidating all the identified differences.

Appendix E – Tester Analysis Sheet

Legend	Required Fields	I Fields	Optiona
Note	All input as well as output Fields must be entered here even though they may hav we make sure that we have all the fields in some positive or the other.		
Input/Output Field	Possible values	ase Dependencies	Datab
Input Fields			
ABC_Amt	1. Take care that sometimes commission is calculated. 2. Take care that amount is such that we take care of all possible TAKDOWN and PUTUP cases		-
XYZ_Id	1. XYZSUS 2. ABCSUS 3. AUTOSUM		1. XYZSUS if BNKOPTION(7th Bit) 2. ABCCRH=1 (Encroached accou

Figure 7

Features:

- This template has separate sheets to capture the functionality of Input Fields Validation, Database Dependency and Input/Output fields' population.
- Input field validation not only captures the dependency on other input fields or database fields but also the error/warning messages attached to those fields.
- Database dependency captures the list of database fields to be updated and logic attached to each field.
- Input/Output fields' possible value captures the possible values and dependency of those values with database fields.

For more information, visit at www.niit-tech.com or contact it.solutions@niit-tech.com