

STeP-IN SUMMIT 2007

International Conference On Software Testing

Effective Regression Strategy for Telecom Systems

By

Raju K S R and Haridas O K
Motorola India Electronics Ltd., Bangalore,
India.

a21679@motorola.com and a19624@motorola.com

Copyright: STeP-IN Forum and Quality Solutions for Information Technology Pvt. Ltd.

Published with permission for restricted use in STeP-IN SUMMIT 2007 in agreement with full copyrights from owner(s) / author(s) of material. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior consent of the owner(s) / author(s). This edition is manufactured in India and is authorized for distribution only during STeP-IN SUMMIT 2007 as per the applicable conditions.

Practices Experience Knowledge Automation

Produced By

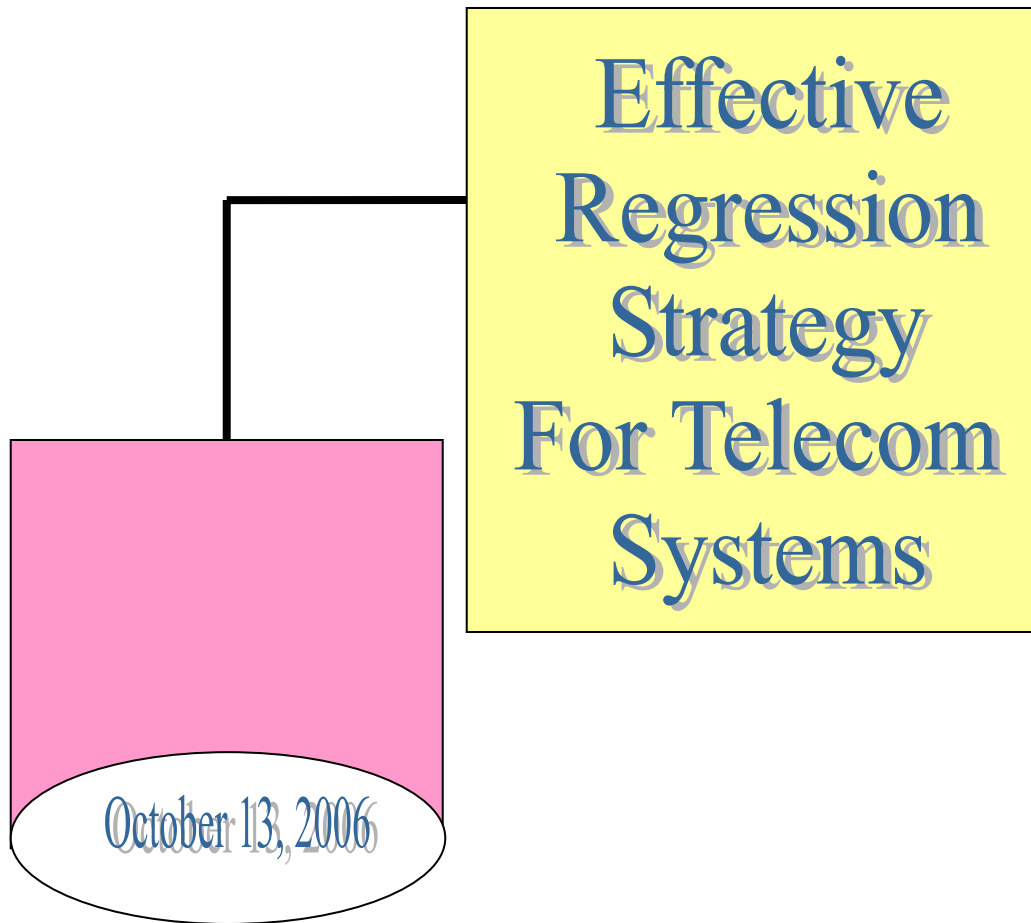
STeP-IN
F O R U M

www.stepinforum.org

Hosted By



www.gsitglobal.com



ABSTRACT

This paper proposes an effective strategy for Regression Testing for complex Telecom Systems wherein the changes going into each release/version are high.

This paper attempts to create a Test Strategy that optimizes the regression suite without losing the focus of Regression testing. We tried to come up with the guidelines for Effective regression by analyzing the key attributes that need to be considered while refining the strategy for regression. This does not mean that the Test Cases need to be removed from the regression suit in order to shorten the overall test cycle and reduce the cost, but this strategy would give a smart solution to build an optimized and effective Regression Suite.

The major categories that contribute to the regression are:

- Maintaining the regression suit effectively
 - Guidelines for adding Test Cases to Regression suite
 - Test Coverage
 - Controlling the growth of Regression Suite
 - Categorizing Test Cases inline with ODC (Orthogonal Defect Classification)
- Test Selection strategy
 - Effective Test Case Selection Criteria using available tools
- Effort and Cycle time reduction
 - Automation of Test Setup
 - Automation of Test Case Execution
 - Verification of Test Results using Tools
 - Batch Mode Execution using the tools

Introduction:

The complexity of the Telecom Systems is growing day by day and whenever there is a new feature or enhancement going into the product, there is a need to ensure that the legacy functionality was not broken. This is being achieved through Regression Testing. With increase in the number of versions of the product, the Regression Testing becomes tedious and more costly as there is significant increase in the number of Test Cases in to Regression Suite. There is a need to define an effective strategy for Regression Testing for complex Telecom Systems wherein the changes going into each release are high.

The need for the Effective regression Strategy:

With the increase in the complexity of Telecom Systems, the growth of the regression suit is a major concern for all the organizations. With the rapid growth of the test suit, every organization faces the challenge to reduce the regression effort and cycle time. At the same the purpose of regression testing must be met by ensuring that no legacy functionality was broken by the new Feature/enhancements. Going forward the maintenance cost of the regression suit increases with the number of test cases added into the regression suit and with the changes in the legacy functionality.

With the use of proposed Effective regression Strategy, we can overcome most of the difficulties that were mentioned above and there by an organization can save more than 60% of Regression effort and more than 50% of Cycle time.

How to Implement:

❖ Maintaining the regression suit effectively

Each organization must have well defined guidelines to select the right test cases from the feature test suit into regression. These cases should cover the full functionality of the feature. Below are points to be considered before adding into the regression suit.

➤ Guidelines for adding Test Cases to Regression suite

A clear set of guidelines for adding Test Cases into the regression suite is the most important for any organization. There are different categories of Test Cases that get created for each feature testing. Regression suite should be a controlled repository and not all of the Test Cases arising from Feature Testing should be dumped into the regression suite.

For example if there are 500 new Test Cases resulted from a new Feature Testing, then not all 500 Test Cases should be dumped into the regression suite. Instead categorize these 500 Test Cases based on functionality and from each category select the bare minimum number of Test Cases to be rolled into regression suite. This number could be anything between 30 to 40% of the total new feature testing Test Cases, i.e. 150 to 200 Test Cases in this case.

➤ All Integration test cases

Integration test cases cover multiple functionalities and therefore by choosing such Test Cases, the simple functionality Test Cases can be eliminated without impacting the coverage. For e.g. the test case which involves both call processing and O&M can eliminate a test case with call processing alone.

- All Complex test cases
Complex test cases are the right choice over simple test cases by the fact that one complex Test Case can cover functionalities spread across many simple Test Cases. For e.g. Complex test case covering the Feature interaction can eliminate simple test case from both the features.
- Boundary value test cases
Boundary value test cases mainly check the system robustness against errors injected by external subsystems and/or systems. So they need to be very much part of the regression suite.
- A sample of successful test cases
Apart from the above categories, from the remaining set of Test Cases, an optimum number of unique successful Test Cases can be chosen. This will eliminate the same functionality being verified under different configurations.
- A sample of Failure test cases
Failure scenarios are intended to test the system robustness. So an optimum number of unique failure test cases can be chosen.

➤ Test Coverage

One important factor to be considered is the coverage of the Feature level requirements in the regression suite. Any new functionality is defined by a set of requirements. The Feature Test Team writes the test cases covering all these requirements. The following cases may arise while coming up with the test cases:

One requirement may map to multiple Test Cases

One Test Case may map to multiple requirements

Under such situations, one should select an optimum number of Test Cases in such a way that 100% of the Feature requirements are covered by these Test Cases. One technique that can be used here is to eliminate the redundant Test Cases. Redundancy can be defined the following way:

Consider the following cases:

Test Case1 -> covers a particular requirement1 of the Feature functionality

Test Case2 -> Covers requirement1 + requirement2

In such cases Test Case1 should be considered redundant and Test Case2 should be chosen for roll over into the regression suite.

A typical example in Telecom Testing in the call processing area would be:

Test Case1 -> Intended to test the call origination functionality

Test Case2 -> Intended to test the handoff (Mobile calls)

If Test Case2 is analyzed then it can be noticed that it first makes a call origination and then tries to do a handoff. Thus it clearly indicates that Test Case1 is not required to be rolled into regression suite if Test Case2 is rolled over.

➤ Controlling the growth of Regression Suite

One factor that can influence growth of regression suite is re-usability of the Test Cases that already exist in the regression suite by the Feature Test teams. This is a very important attribute which can control the growth of regression suite to a great extent. It is a very simple technique. When you put some quantity of Test Cases into the regression suite, you also make sure to remove the obsolete test cases from the regression suite.

If 100 Test Cases from feature test team needs be added into regression suite, then browse the regression suite (using a tool through which we can get a set of test cases similar to the

new test case) for all possible Test Cases which can match the new Test Cases. Add the new 100 Test Cases and at the same time remove the old obsolete ones from the regression suite.

➤ **Categorizing Test Cases inline with ODC (Orthogonal Defect Classification)**

While designing Test Cases, classify each Test Case based on ODC triggers and/or enter the standard keywords based on functional area. A pre defined set of keywords need to be embedded into the test cases. The keywords are put within the test case to identify that the test case is covering a particular functionality. For example if a test case is written to test soft handoffs, then the keyword SHO may be put within the test case. This will benefit during the selection of the test cases for the regression.

❖ **Test Selection Strategy**

Test cases should be selected appropriately for the regression based on the functionality that was added/modified, impacted functional areas and based on the keywords entered into the test cases. Test cases can also be selected based on the past data, say high fault yielding test cases. This is possible if the organization is maintaining the execution records of previous regression cycles.

➤ **Effective Test Case Selection Criteria using available tools**

From the regression suite, using available tools, an optimized set of test cases need be picked up for execution. The Test selection can be done by the following ways:

- **Using the keywords in the test cases**

Make use of the keywords embedded in the Test Cases to identify the Test Cases that are being impacted by the new feature. If a particular feature impacts soft hand off functionality, then test cases with SHO keyword can be selected for execution. A tool need to be developed to parse all the test cases and gives the list of test cases against the given keyword.

- **Using past execution data**

Test Cases can be selected based on past historical data maintained in a database. Test Cases which have been yielding defects in the past can be selected for execution during regression. A tool can be developed to fetch the high defect yielding test cases from the database.

- **Selection based on functions modified**

Test Cases can be selected based on the functions modified/added by the development team. Get the list of all functions modified by the development team and based on those functions select the most matching Test Cases from the regression suite. This can be achieved by developing a tool that gives the methods/functions modified for a given feature and mapping these methods into functional areas and/or keywords.

❖ **Effort and Cycle Time Reduction**

Saving of Regression Testing effort and time can only be achieved if it is optimized and most effective. Timely delivery is very essential for every organization and so optimization of the regression suit is very important and there by one can reduce the regression cycle time and the regression effort. Effort and cycle time reduction is possible to great extent through

automation.

➤ **Automation of Test Setup**

Setting up the Test bed for Test Execution eats up much of the time from the entire regression cycle. This is mainly due to the following factors:

- Human errors occurring during the bring up of Test bed
- Configuration errors
- Limited knowledge of the tester

All the above factors contribute for substantial increase in the time taken for bringing up of the Test Setup. This can be eliminated by automating the Test Setup process either by using available tools or by writing scripts.

➤ **Automation of Test Case Execution**

The Test Cases written by feature test team can be executed manually which is very time consuming. The Test Cases can be automated in which case the testing does not need any manual intervention and will reduce cycle time considerably.

At the time of feature test case design, additional effort have to be spent to automate the test cases however that is only one time effort and will return benefits over a period of time. Test Cases can be automated using many ways like:

- Test Scripting
- Develop tools to generate test scripts

➤ **Automation of Test Result verification**

One of the major activities during regression testing goes in verifying the Test Execution results. At the end of each Test Case execution, the tester should carefully verify the results of execution from the logs collected. Some of the issues which can results during manual verification of results are:

- Misinterpretation of Test results
- Oversight

Such issues can be overcome if the verification can be automated. Again a bit of Test scripting may be required at the time of Test design.

➤ **Batch Mode Execution**

Executing Test Cases one by one requires continues effort from the tester. Manual test cases mostly require execution of test cases one after the other. However if test cases are automated, then by using a small script, the test execution can be made in batch mode. Whether 100, 200 or 500 Test Cases, they all can be put in batch mode execution and no manual intervention is required. The execution can even be triggered during off office hours and thus cycle time and effort can be reduced.

This paper also proposes the guide lines for rolling over the new feature Test Cases into the Regression suite. Here are the Test Case properties that need to be considered:

✚ **Test Case independence**

A Test Case is said to be independent when that test case can be executed irrespective of any initial condition of the Test Setup. A previous test execution might have put the test setup

into a different state. But that should not impact the execution of the current test case.

Restoring the initial condition

Every Test Case should at the end of execution, restore back the initial conditions of the test setup. This is very important for the smooth execution of subsequent test case. So a cleanup block should be the last block that gets executed for each test case.

In order to verify the above properties for the test cases, the test cases can be executed in the following order and make sure everything works fine.

The new test cases must be executed in batch mode in the following orders:

- Top to bottom
If there are 100 Test Cases, then execute them in the order 1, 2, 3...99,100. Make sure every Test Case works fine in that order. This ensures that execution of Test Case1 does not impact the execution of Test Case2 and so on.
- Bottom to top
In a similar way execute the Test Cases in the reverse order like 100, 99 ...3, 2,1.
- Odd numbered Test Cases followed by even numbered Test Cases.
Test Cases need to be executed in the order of odd numbered ones followed by even numbered ones. Like 1, 3, 5.....99, 2, 4, 6....100.

Executing the Test Cases in such orders ensures the robustness of the Test Cases and thus contributes for a robust regression suite.

Case Study

The case study below explains how the organization xxx has benefited by employing the effective regression strategy. The data shows a comparison between the effort and time taken for a regression cycle before implementing effective regression strategy and after implementing it.

Before implementing the effective regression strategy in release n

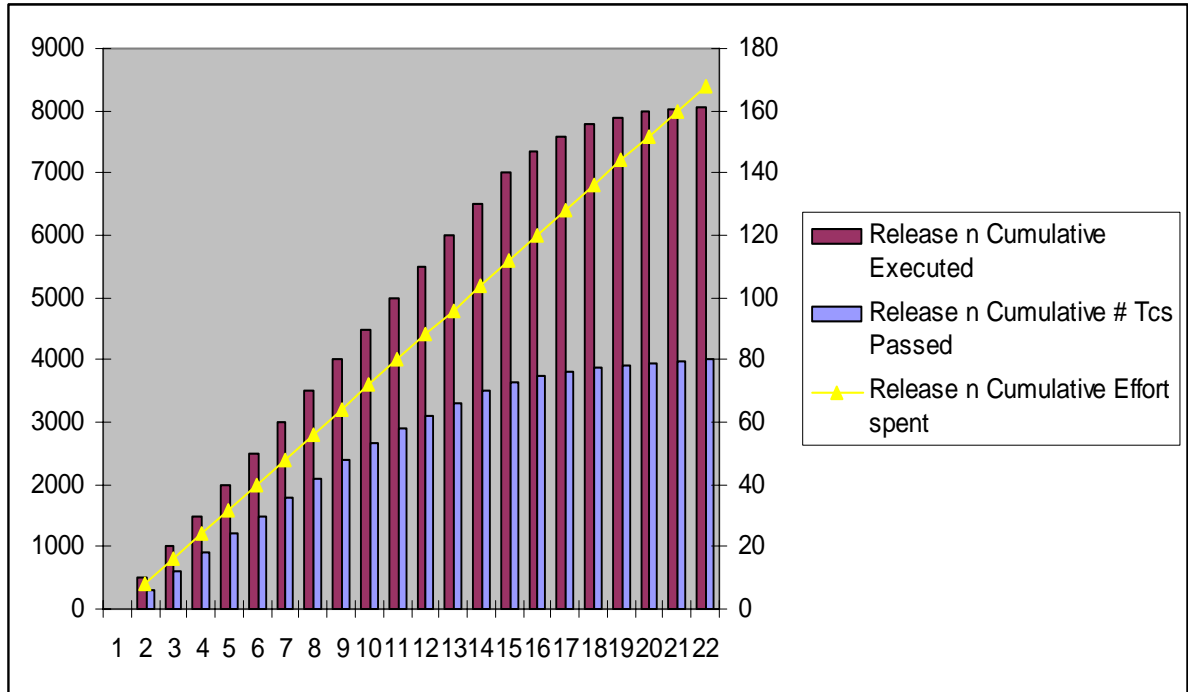
Total number of Test Cases present in regression suite = 4000

Total number of Test Cases selected for regression cycle = 4000

(All Test cases were selected and were automated)

Total effort spent for complete execution of one cycle of regression = 1 staff month

Cycle time taken = 1 month



After implementing effective regression strategy in release n+1

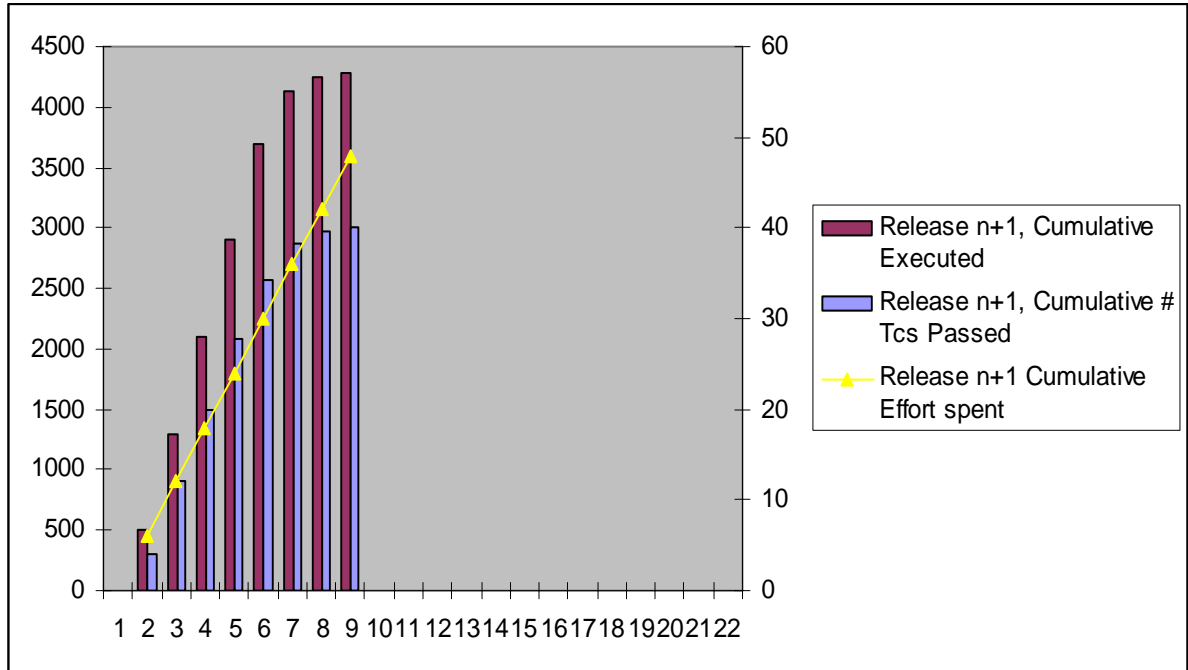
Total number of Test Cases present in regression suite = 4500

Total number of Test Cases selected for regression cycle = 3000

(Used tools for effective selection of Test Cases from regression suite using keywords and all test cases were made independent through cleanup)

Total effort spent for complete execution of one cycle of regression = 0.38 staff month

Cycle time taken = 8 days



Observations:

- Execution effort has been reduced to 0.8 minutes per test case from 2.5 minutes per test case (**68% SAVINGS**)
- Cycle time has been reduced to 8 days from 21 days (**62% SAVINGS**)
- Re executions were reduced to 43% from 101%. This is achieved due to the robustness brought into the regression test cases.
- Note: The savings may vary based on the complexity involved in the system.

Conclusion

By practicing Effective Regression Strategy, Telecom organizations benefits the cost savings and reduces the regression cycle time. Three major categories to contributing to the new regression strategy were discussed, i.e. maintaining the regression suit effectively, Test Selection Strategy and Effort and cycle time reduction. New guidelines for the test cases before rolling over into the regression suit were proposed. A case study on this effective regression strategy has been presented with a data of 62% reduction in cycle time and 68% reduction in regression effort.

About Authors:**Raju K S R**

Having nine years experience in the Telecom Industry which includes seven years of Telecom Testing experience. Currently working with Motorola India Electronics Ltd and into the Sub System Testing of CDMA Mobility Manager. Previously worked on the Telecom systems like Ericsson's CDMA BSC, Ericsson's PABX –MD110, AT&T-GTE's GTD switch and ITI's Cable Saver.

Haridas O.K

Having 11 years experience in the Telecom Testing Industry. Currently working with Motorola India Electronics Ltd and into the Sub System Testing of CDMA Mobility Manager. Worked on testing of CDoT's Digital Switching Systems, VoIP systems.