

# Agility in Testing – DECOP Method

K. Madan Mohan Reddy  
Tata Consultancy Services Ltd,  
Hyderabad, Andhra Pradesh, India  
m.reddy@tcs.com

## **Abstract:**

In this paper, we describe our experience in the testing industry, particularly in the area of development and testing of a Telecom product. The scope of testing involves new features, enhancements and fixes to trouble/problem reports. The purpose of our paper is to highlight the DECOP method and best practices evolved in deployment in the project.

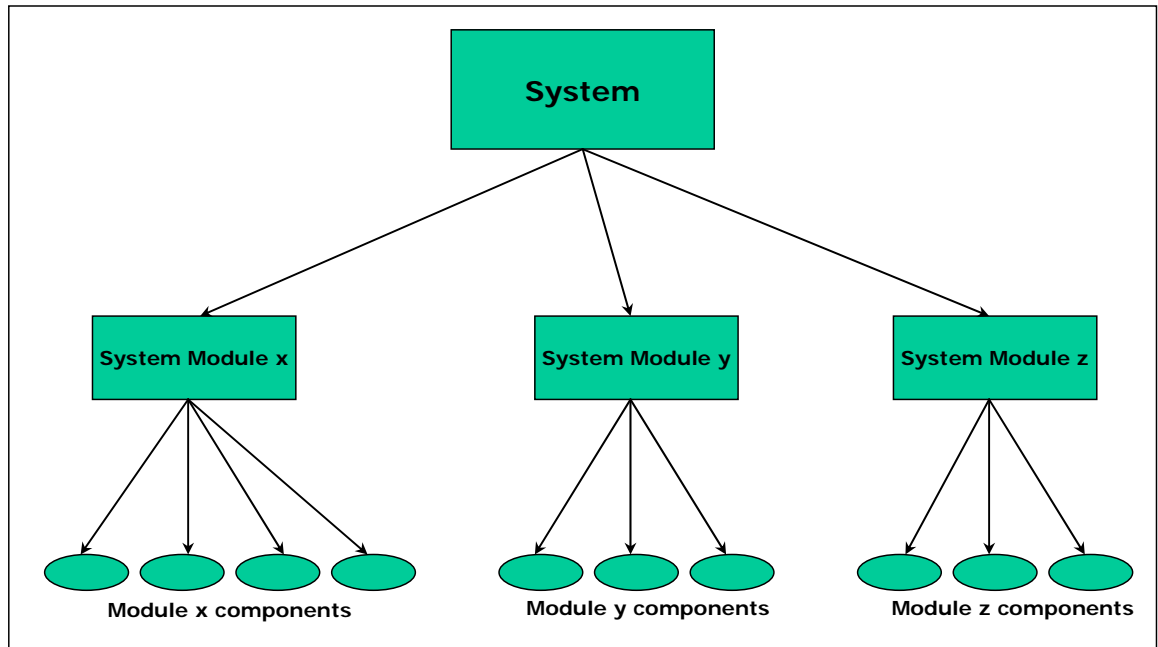
## **Keywords:**

DailyTest  
DailyBuild  
Integration  
Function Test  
System Test  
Regression Test  
System Module  
Components  
Automation

## **1. Introduction**

Today's software development organization faces challenge in getting the software to the end user more quickly than ever before and must optimize the quality of increasingly complex software applications -- in order to deliver award winning solutions that yield a high ROI and drive competitive advantage.

Many software organizations have responded to this challenge and brought in Agility in Testing. In achieving this companies of all sizes have successfully demonstrated the power of automated testing, rapidly develop and reduce the cost of delivering high-quality applications. With the high level of automation, daily build and daily test is also introduced to reduce the period of test cycle with increased efficiency and better quality.



**Figure 1: Representation of the System Hierarchy**

## **2. DECOP Method**

The DECOP method is an efficient way to test the software. It is an initiative to solve problems such as lengthy delivery cycle, late deliveries and decrease of system quality.

The DECOP method has five principles:

- **D**ailyBuild and **D**ailyTest
- **E**arly quality check
- **C**ontinuous integration planning
- **O**ne development track – few branches
- **P**arallel test and verification on all levels

## **2.1. DailyBuild (DB) and DailyTest (DT)**

Connecting DB and DT allows the organization to build a new system every day based on the latest deliveries, create upgrade packages, builds and verifying the quality of the SW on a daily basis. This is done automatically and performed every night.

DB and DT can be used by system module integration to create a new upgrade package containing only the latest SW from that system module. System integration creates new upgrade package contains the latest delivered system modules.

## **2.2. Early Quality checks**

There are many approaches to software testing; we use different techniques to find different faults in each test phase.

### ***2.2.1. Document Inspection***

As the first static test phase, Document Inspection has the goal to verify Design specifications against the requirements.

### ***2.2.2. Code Review***

As a second static test phase, Code Review has two goals:

- Verify the code implementation against the Design Specifications and various Design Rules
- Sanity check the code implementation using peer review technique

### ***2.2.3. Component test***

Component Test tests the minimal software unit, or component. The goal is to verify that the design of the component has been correctly implemented.

Static analysis tools like Coverity, Purify, PureCoverage, Lint, etc are used in this phase. Component test includes both regression test and test of new functionality.

### ***2.2.4. Integration test***

Integration Test exposes defects in the interfaces and the interaction between components. Progressively larger groups of tested software

components are integrated and tested until the software works as a system. Integration Test is done in DailyTest, System Module Integration (SMI) and System Integration (SI).

The test cases used in integration test are from regression test.

#### **2.2.5. *Function test***

Function test tests at system level for proper functionality as defined in the system function specification. Functional test cases correspond to functional specifications. Function test is done with system module focus and is used to verify new functionality. Also Trouble Report verification is also carried by the function testers, this will ensure that the bug is fixed before we deliver to the customer.

Many Function test tools are available in open source to create test frame work. Tools like Abbot Java GUI Test Framework, DejaGnu, Expect, WebInject, etc.

#### **2.2.6. *System verification***

System verification tests a complete integrated system to verify that it meets the requirements. This includes Performance, load, stress, volume and usability testing.

Many tools are available in open source to create test frame work. Tools like Apache JMeter, WebLOAD, TestMaker, stress\_driver, etc.

#### **2.2.7. *Regression test***

Regression Test uncovers regression faults. Regression failures occur whenever functionality that previously worked as desired, stops working or no longer works in the same way.

Function test cases are candidates to be included in regression test to secure that new functionality will continue to work in the future.

### **2.3. Continuous integration planning**

DECOP emphasizes the importance of frequent deliveries of changes from design to test. Designers should submit code as often as possible (every

day) and the integration teams should get new impacted component as often as possible (every day).

The consequence of this principle is that the integration and test teams will get new functionality and corrections at the same time. This could mean that the re-tests will be run on a system or a part of the system where the functionality has been changed at the same time.

#### **2.4. One development track – few branches**

Using DECOP, corrections are planned and delivered by development team in a frequent manner in the same branch as new functionality. It means that test needs to be able to handle delivery of corrections together with new functionality.

Working in one or few branches implies that we must use one and the same system version in test, the latest one. In SMI we may not always want to use the latest version of all system modules. If we need a stable base we can use the last released baseline version.

#### **2.5. Parallel test and verification on all levels**

One important aspect of DECOP is to get feedback as early as possible. If we can get as many as possible to lay their hands on the new system version as early as possible we would get earlier feedback. To do this we need to parallelize the test activities on all levels.

Some parallelism will be used. SI and SMI will in some aspects run in parallel. Function test will also be executed in parallel with both Regression Test and System Verification.

Finally, upgrade to the new system version must be supported efficiently to avoid older system versions being used by testers and to avoid delays.

### **3. Test tools**

#### **3.1. DailyBuild**

DailyBuild is an automated way to build binaries and external libraries. It can be run on a daily basis or even several times per day. The result from DB like logs from build and binaries are presented graphically on the web and the binaries can be stored in a version control system (like Clearcase). DB works according to the build-when-needed principle. It only rebuilds binaries when there is new code that shall be included in those binaries.

#### **3.2. DailyTest**

DailyTest is a tool that starts automated test suites and presents the output on the web. DT also provides functionality to build upgrade packages and perform upgrade of multiple systems and run specific test suites on each system. DT can be manually started, but the normal use is to let DB trigger it automatically as soon as new components, binaries, and the Upgrade Package (UP) are ready.

#### **3.3. Test cases Automation**

All Test cases (includes System, Integration, Function and Regression Tests) are automated. It can be run on a daily basis or even several times every day. The result from automation logs are presented graphically on the web and the results are stored in the Tool. We can use these scripts for reproduction of faults.

### **4. Benefits**

With the introduction of the DECOP method we have demonstrated that it takes less time than the traditional method and also important aspect is to get feedback as early as possible, this will minimize the risks of the problems.

#### **4.1. The cost of introducing faults**

One of the main contributors for unnecessary work in the projects is that we find too many costly faults late in the project. It is commonly believed that the earlier a defect is found the cheaper it is to fix it. The ultimate

solution to this problem would of course be to stop introducing faults in the first place. But, we can never be sure that there will be no faults so we also need to optimize the use of test resources.

#### **4.2. The quality of the next system version**

One goal in One Track is to make each version of the system better than the previous. But, what makes a version better than the previous version? Would it be better if we find a less number of bugs during system verification? Would the next system version be better if we increased the number of test cases?

In the end, the only person who can tell us this is the customer or the end user. The user perception of the functionality and the system quality must be the only true measure of this.

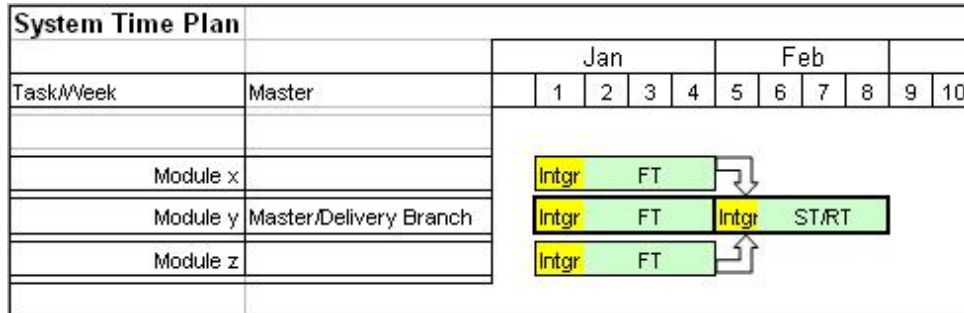
Getting end-users to test each system version can be very hard and costly so a cost efficient way to do this is to create a customer-like environment in system verification.

#### **4.3. Short iterations**

One Track emphasizes the importance of frequent deliveries of changes from development to test. Developers should submit code as often as possible (every day) and the integration teams should get new impacted component as often as possible (every day).

The consequence of this principle is that the integration and test teams will get new functionality and corrections at the same time. This could mean that the re-tests will be run on a system or a part of the system where the functionality has been changed at the same time. As a tester you need to be aware of this.

The best way to handle this issue is to be very clear about what has changed and communicate this in a proper way.



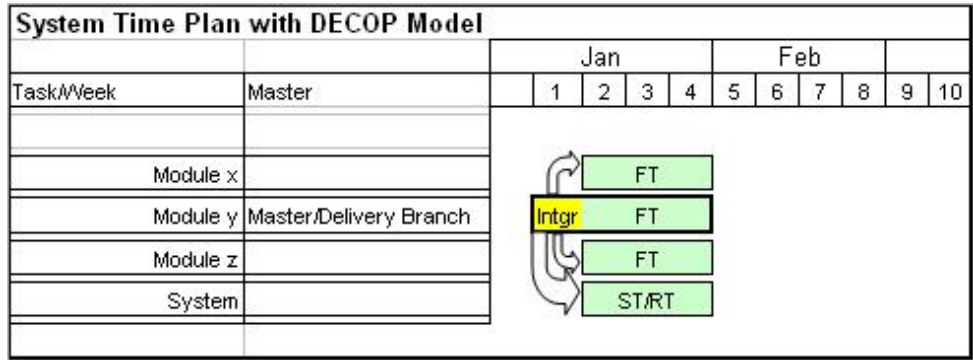
**Figure 2: Representation of the Traditional Test Model**

## 5. Case Study

The purpose of the paper is to share in detail the DECOP principles followed by a case study that highlights the success of this approach.

As shown in the Figure 2, each delivery cycle will take 8 weeks. As Function testing, System Testing and Regression testing are executed one after another. Integration of components on each system module is performed first before starting the function testing and system module integration testing at each system module level. This is to make sure that the new functionality introduced work as designed/required. Function testing and System Module regression testing will take 3 weeks to complete. After meeting the system module exit criteria, each of the system modules are integrated to perform Regression, integration and system tests. These tests will take another 3 weeks to complete.

In this model, each new requirement/s (set of functions) is carried out in its own branch. Later all these branches are merged to the System Module branch where System Module integration is performed. Finally all the other System Module branches are merged to the Master/Delivery Branch (one of the System module branch).



**Figure 3: Representation of the DECOP Test Model**

With the implementation of DECOP model we can perform the integration activity in one delivery branch and also reduce the delivery cycle to 4 weeks. One important aspect of One Track is to get feedback as early as possible. With DB and DT possibility of building a new system every day based on the latest deliveries, builds and verifying the quality of the SW on a daily basis. This is done automatically and performed every night. With this build System Modules function and integration testing can be performed in parallel to the System and Regression testing. In this model all the integration activities are done in one branch. Due to this lot of effort on the merging and other build activities can be minimized. As shown in the Figure 3 each delivery cycle takes only 4 weeks.

**6. Citations**

- **Book**  
 Andreas Spillner, Thomas Rossner, Mario Winter, Tilo Linz: 2007. *Software Testing Practice: Test Management*. Mumbai: Shroff Publishers & Distributors Pvt Ltd.
  
- Dorothy Graham, Erik Van Vennendaal, Isabel Evans, Rex Black: 2007. *Foundation of Software Testing*. Delhi: Rahul Print O Pack.
  
- **Book Chapter**

Andreas Spillner, Thomas Rossner, Mario Winter, Tilo Linz: 2007. *The Test Plan: The level test plan* (pp 55). Mumbai: Shroff Publishers & Distributors Pvt Ltd.

Dorothy Graham, Erik Van Vennendaal, Isabel Evans, Rex Black: 2007. *Testing throughout the Software life cycle: Test levels* (pp 41-50), *Static Techniques* (pp57-73). Mumbai: Shroff Publishers & Distributors Pvt Ltd.

## References

- [1] Andreas Spillner, Thomas Rossner, Mario Winter, Tilo Linz: 2007. *Software Testing Practice: Test Management*. Mumbai: Shroff Publishers & Distributors Pvt Ltd.
- [2] Dorothy Graham, Erik Van Vennendaal, Isabel Evans, Rex Black: 2007. *Foundation of Software Testing*. Delhi: Rahul Print O Pack.
- [3] Opensourcetesting.org: 2003: Open source Software Testing tools. <http://www.opensourcetesting.org>
- [4] Test2008: 2008: Conference Proceeding Template. <http://test2008.in/icstd-template.doc>