



# Module Based Regression Test Selection Strategy for Web Applications

Annamariale Chandran

Honeywell Technology Solutions  
Madurai-625015, Tamilnadu, India

91+ 09894316262

[Annamariale.Chandran@honeywell.com](mailto:Annamariale.Chandran@honeywell.com)

**Abstract** - For projects in the maintenance phase, updated specifications and test documents may not be maintained in a single repository. Even if they are maintained, synchronization with current changes may not be available and hence direct reference of these independent artifacts is difficult to select regression test suites. Invariably, dedicated test resource may not be allocated for a specific support project and hence the impacts on current changes with the previous specifications will not be available with the tester. Moreover, agile environment for maintenance projects adhere to deadline and hence it becomes hard to use formal Regression models.

In this paper, the approach envisaged a strategy to validate the modified software by identifying inter-related modules to perform regression testing. The concept is derived from basics of statistical theory on regression analysis. This approach mainly focuses on reducing cycle time for test design and execution by ensuring potential requirement coverage thereby reducing rework in early stages of development. Though the approach targets regression strategy for web-based applications in Maintenance/enhancement phases, it is also recommended to use for applications in different domains by identifying its relevant primary dependent variables.

**Keywords** - Regression testing, Impact, Reverse Engineering.

## I. INTRODUCTION

Regression testing is an expensive testing process but necessary in maintenance activities. It is used to validate software after some modifications are incorporated in the application. The definition for Regression testing is stated as “the process of validating modified software to provide confidence that the changed parts of the software behave as intended and that the un-changed parts of the software have not been adversely affected by the modification”. Thus regression selection aims at validating not only the changes, but also the impacted functionalities due to those changes. Most of the existing regression models attempt to find test suites for the impacted functionalities. These identified test suites

claims to provide confidence in validating the modified software and the unchanged parts.

Thus regression testing is a backward approach, where a new functionality or feature that is integrated into the system is first tested and then followed by the previously tested functionality. The correctness of any regression method is determined by the proper selection of previously tested functionalities. This selective functionality is often termed as impacted functionalities. The current techniques mostly aim at directly deriving the impacted test cases from the previous versions of test design document. In this context, there are also possibilities that the new modifications in software shall considerably change the impacted functionality. Thus there will be a change in the “expected results” of the test cases derived from earlier test artifacts. Along with the huge effort needed in drilling down the test cases, using existing models, rework in modifying the derived test cases shall also lead to poor Return on Investment (ROI). Also the drilled down test cases will not help in understanding the functionalities of the modules and its relevant high-level scenarios for a new test resource (even for the resource who have worked previously, it will be difficult to recap the functionalities with the available test cases alone).

In case of support projects, it is observed that nearly 30 to 60% (variation based on project complexity, resource capability and project management) of the project effort is spent in analyzing the impact due to the modification. The analysis effort includes both testing as well as implementation effort. This is evident from the consolidated real time data in support mode & the details are given in ‘[Results](#)’ section. Also improper identification leads to rework (from development to testing phase) and increases post release defects (from testing to production) and affects the project’s overall efficiency.

In common practice, one analyzes the impact of the application by first trying to know the high level functionalities of the application and then segregates the related functionalities into modules along with the details

of high-level scenarios. For any new changes, one tries to find in which module (functional group) it falls and then try to find the relative impacted modules and functionalities and then start writing test cases for the validation. This practical concept along with some 2 refinement in steps for better ROI is incorporated in Module based regression test selection technique. An important difference between the available regression selection technique and proposed theme is that it is not just an algorithm or guideline, but a model that extracts all impacted modules due to the insertion of new functionality.

## II. DRAWBACKS OF EXISTING TECHNIQUES IN SUPPORT PROJECTS

Although existing research has addressed many problems and put forward solutions, most regression test techniques are code-based. Code-based regression test selection is good for unit testing, but it has a scalability problem. When the size of the subject under test grows, it becomes hard to manage all the information and to create corresponding traceability matrices.

Specification based technique aims at providing a methodology to determine the test cases path by an activity diagram that considerably increases analysis time to extract the test suites. When the project enters maintenance mode, the confidence level in getting updated specification and test case documents are only around 60- 70% and this percentage decreases as the changes of the project in maintenance phase increases. The nature of work in case of maintenance task should also be taken into consideration while applying any regression models. Support projects demand on-time delivery with more accurate results. So every time to follow some strategy (from any available regression technique) to extract regression test suites will consume more effort. In some cases, this process will consume effort that shall overtake the entire cycle time of completion of the implemented changes. As stated previously, in case of support projects the specification will not be an updated one maintained in a single repository. A demand of this requirement cannot be expected from the project, since the resource availability and environment factors (like production fix) will be the constraints and any control mechanisms shall be less effective. For changes in the application, to find the corresponding requirement and test cases from the available traceability matrix will be time consuming. In case of any new requirement, to fit the same in the existing traceability document shall be a tedious process. Thus specification based technique may not be a correct approach to apply in support projects.

One other approach is to re-use all the previously developed test cases and executing them on the modified program. Even for a small portion of modification in the system, this retest-all approach is not feasible and also not recommended in business perspective. This technique also assumes that the individual who are involved in Regression selection has a thorough understanding of the application. But this fails in most of the cases for the projects in maintenance phase and to some extent in the development projects. As stated earlier, all the available regression selection methods aim at finding the test suites and the extracted test suites shall not be feasible in applying directly for validating the modified changes for the below mentioned reasons.

- Changes on projects under maintenance are normally validated with a separate test case document and not updated on the earlier version of the project.
- In most cases, the original test cases may be entirely changed from the current functionality, though the module remains same and also this cannot be traced due to the specific nature of support projects.
- The skill level of resources may not be equal in deriving the test cases. In some instances, the new resource may find it difficult to understand the existing test cases.

The following expectations also serve as a block to adopt the available regression techniques

- Instant understanding of high level functionalities of the derived test cases
- The module in which it falls
- Understanding the existing scenarios from the extracted test suites
- The priority & severity of those test cases

The above expectations along with the specified drawbacks lead to new Regression selection criteria for projects in maintenance phase. Also the listed gaps can be narrowed to some extent when scenario level cases are provided along with the impacted modules. Ambiguity for change in functionalities to identify the impacted modules is also considered in this new regression selection approach.

## III. MODULE BASED REGRESSION SELECTION TECHNIQUE

To derive the impacts of the changes, **parametric statistical** theory is used. It is a branch of statistics that assumes expected data to come from a type of probability distribution and makes inferences about the parameters of the distribution. The derivative of this theory in system



testing phase to perform regression analysis is given below:

*Module relation with primary dependent variables*  

$$y(m) = f(x(p), \beta) \quad \text{-----(1)}$$

Where,

$y(m)$  -> denotes the modules derived in the application under test

$x(p)$ ->primary dependent variables identified for the derived modules

$\beta$  -> referred as intelligent guessing function that ensures the correctness of mapping the primary dependent variables to the identified modules

*Final parametric relation between modules*

$$Y_d(M) = y_{id}(m)f(\alpha) \quad \text{----- (2)}$$

Where

$Y_d(M)$  ->is the dependent module that will get impacted another module in the application

$y_{id}(m)$  ->is the independent module referred at that instant

$\alpha$  -> referred also as an derived function from equation (1) to differentiate  $Y_d(M)$  and  $y_{id}(m)$

#### IV. CASE STUDY

As stated above, regression analysis aims at measuring the relation of the independent variable to its dependent variable(s) so as to predict the future value of dependent variable. In the context of software testing for web-based applications, both dependent and independent variables are considered as modules in the existing application. To derive the regression curve between the modules, the intermediate step is to consider the factors that lead to find the inter linkages between the modules. These factors are the primary dependent variables. The primary dependent variables are considered as ‘fields/ sections’ and ‘functionalities’ for web-based projects.. This approach is explained by a case study for better understanding. A simple application that manages employee request and workflow process is taken for the analysis and this application be termed as Employee Request Management system (ERM).

##### A. Requirement Database

The conventional process provides the existing specification document, test case design and traceability documents. But the prerequisites for module based approach are to have the listed sub-step details as

requirement database. As and when the testing phase gets completed for a project in Full Life Cycle (FLC) development, the test resource should collect the requirement database information. Even if this information is not received from FLC mode, it is recommended to have some agile practices (like SCRUM meetings, document reduction techniques) to gather the updated information as one time investment and the requirement database should also be updated for modified changes.

##### Identify modules

The requirement database for Regression analysis is constructed by extracting the list of modules. Team specific convention names should be given for the extracted modules and sub-modules (M1..... Mn). For the defined set of modules, high-level test scenarios are extracted as an add-on output for the impacted modules. These scenarios in turn should have traceability maintained for the individual requirements within the specific cluster and the collection of modules is explained for ERM application. Here the module notations are meant for supporting internal logic for automatic retrieval of the impacted modules.

**TABLE 1: MODULES AND SUB-MODULES IDENTIFICATION**

| Main Modules   | Sub Modules            | Module Convention   |
|----------------|------------------------|---------------------|
| Login          |                        | ERM_M <sub>1</sub>  |
| Request Form   | User Details           | ERM_M <sub>21</sub> |
| Workflow       | Request Details        | ERM_M <sub>31</sub> |
|                | Primary Approval       | ERM_M <sub>32</sub> |
|                | Secondary Approval     | ERM_M <sub>33</sub> |
| Reports        | User Report            | ERM_M <sub>41</sub> |
|                | Request Form Report    | ERM_M <sub>42</sub> |
| Administration | User Management        | ERM_M <sub>51</sub> |
|                | Workflow Configuration | ERM_M <sub>52</sub> |
|                | Mail Notification      | ERM_M <sub>53</sub> |

For the identified modules and sub-modules capture high level scenarios and functionality and trace it to the relevant modules. The ‘Request Form’ high-level functionality is given below.

- User details should be auto-populated from LDAP for the logged in user
- Mail Id field should alone be allowed to be edited by the user
- Manager ID should be same as that of the user’s portfolio

- Data requirement validation (Refer to xxx SCM location)
- Mail should be sent to the primary approver when request is submitted and status changed to PA (Pending primary approval)

Extract related Fields/Sections

As stated, one of the dependent variable is ‘fields/sections’ for the independent variable ‘module’. For example, if the ‘First name’ is the field that belongs to Employee details section in request form and Workflow module, any change in field information (such as change in maximum length, acceptable data type or removal of that particular field) will also affect the other modules. Thus modules are related through this identified dependent 4 variable and thus confirm that Request form & workflow modules are related through this section - ‘Employee details’

**TABLE 2: EXTRACT DEPENDENT VARIABLE1-FIELDS**

| Modules                      | Fields/Sections for relation identification | Primary relation notation 1                     |
|------------------------------|---|---|
| Login                        | UserName                                    | ERM.M <sub>1</sub> .UN_ERM.M <sub>21</sub> .ED  |
| Request Form - >User Details | Employee Details section                    | ERM.M <sub>21</sub> .ED_ERM.M <sub>3</sub> .EDr |
| Workflow                     | Employee Details section as Read only       | ERM.M <sub>21</sub> .ED_ERM.M <sub>41</sub> .SF |
| Reports                      | Search section – filter fields              |   |
| Workflow                     | Workflow-Configuration                      | ERM.M <sub>52</sub> .WFc_ERM.M <sub>3</sub> .WF |

Extract related functionalities

The second dependent variable is the ‘functionality’. For example, ‘User Request form’ shall accept the Manger Employee ID, only when this ID falls in the Requestor’s portfolio/ domain. The same conditional check should also be incorporated in case of ‘Report’. When a search is performed for a user, the application should return results only for the users under his/ her portfolio. Thus, it relates in extracting only the common functionalities that affects multiple modules (independent functionalities are captured in high-level scenarios)

**TABLE 3: EXTRACT DEPENDENT VARIABLE2-COMMON FUNCTIONALITY**

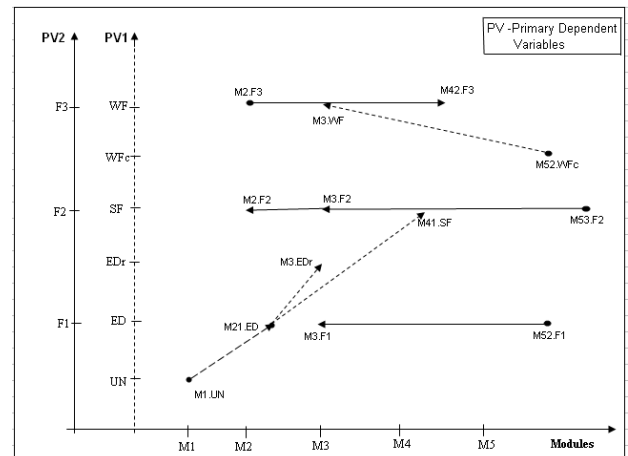
| Related modules  | Common Functionalities            | Primary relation notation 2                |
|--|-----------------------------------|--|
| * Workflow<br>* Administration - >Workflow Configuration | Changes in workflow configuration | F1_ERM.M <sub>52</sub> _ERM.M <sub>3</sub> |
| * Request Form   | Changes in Mail                   | F2_ERM.M <sub>53</sub> _ERM.M <sub>2</sub> |

|   |                              |  |
|---|------------------------------|--|
| * Workflow Administration - >Mail Notification      | Notification                 | _ERM.M <sub>3</sub>                        |
| * Request Form<br>* Reports -> Request form reports | Changes in Manager Portfolio | F3_ERM.M <sub>2</sub> _ERM.M <sub>42</sub> |

The main intention of finding related functionality between modules is that, in case of any change in the identified functionality for one module should also affect the same functionality that falls in a different module. For example, if a change in requirement states that, the user belonging to ‘XYZ’ portfolio should access Manager Employee ID belonging to both ‘XYZ’ and ‘ABC’ portfolio and the regression testing should also focus on validating whether the same requirement is also implemented in ‘Report’ module

*B. Regression logic*

Relative module linkage is identified using the primary dependent variables from the requirement database as shown in Fig 1



**Fig 1. Module linkage for the identified dependent variables**

Here ‘dot’ indicates the module that influences the other module (in this context, dependent module) to be affected because of the influence of primary dependent variables (fields and functionalities) and arrow indicates the module that is affected by the influencing variable. Thus, there are chances that same module shall act as both dependent & independent variable.

Also from the graph, it shall be noted that the same module/ sub-module can influence two or more dependent modules due to different primary dependent variables. Here M52 (Administration->Workflow Configuration) influences M3 (Workflow-> Primary and Secondary approval sub-modules) due to the change in

functionality F1 as well as the configured workflow field (WFc).

There are no restrictions in identifying the primary dependent variables for an application. It can vary, based on the nature of the project. Even in web-based applications, we can include another primary dependent variable as 'Job logic' (background job functionalities that affect the intended application functionality) and the accuracy in finding the impact between modules depend on the identified primary dependent variables.

Using the above extracted information, the final step is to find the actual impact between the modules as shown in Fig 2. All the identified modules in the application are 5 taken in both 'X' and 'Y' axis (in turn considered as both dependent and independent modules). The module

from where 'dot' starts is considered to be the dependent module & 'arrow' as independent module (From Fig 1). Regression plot is then made for the impacted modules. The final outcome of regression logic is the impact between modules, based on the influencing primary dependent variables. Thus, for any modification in the application, when the user provides inputs on the module (where there is a change) the regression logic provides output for its relevant impacted modules along with its relation on the primary dependent variable. This also provides the high-level scenarios for the identified modules. With this information, the validation of modifications is performed in the application

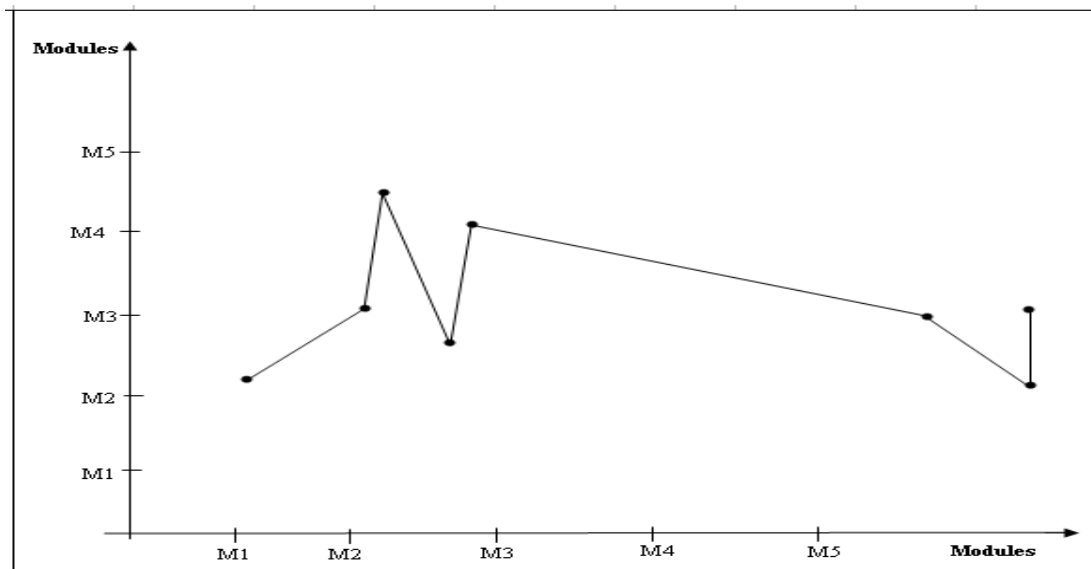


Fig 2. Regression plot for impacted modules

### C. Information Retrieval

For any changes in the application, the retrieval of regression output for creating updated test cases is given in Fig3. It uses the regression logic and in turn the requirement database as described in previous sections. The inputs from user will be the affected module based on the field under change or the functionality. For more specific inputs, the regression logic shall also return the related fields & common functionality of the module.

When the user selects the relevant primary dependent variable information, the output shall still be more specific with the most related impacted modules and high-level functionalities. With this information, the deliverables using MBRT is given in Fig 3. The requirement database also will be updated as per the new changes incorporated in the application and versions will be maintained.

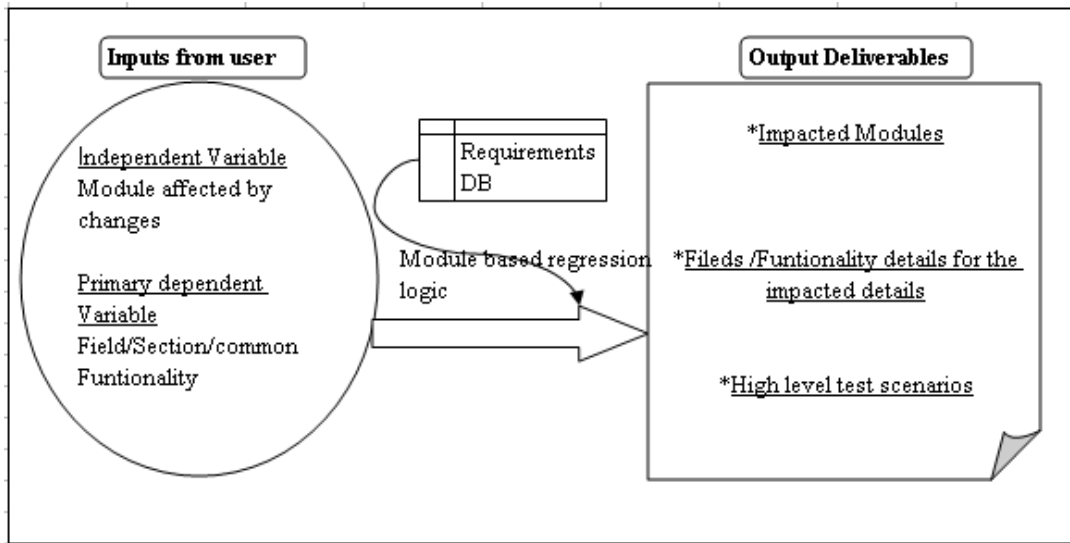


Fig 3. Retrieval of Information using MBRT

## V. RESULTS

The technique is incorporated for some support projects. The overall testing effort that includes requirement understanding, analysis, test design and execution for a single project is given below

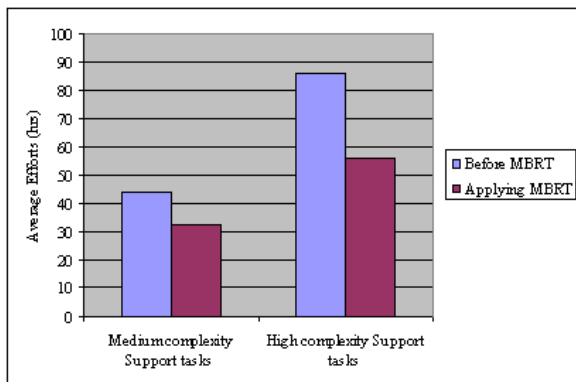


Fig. 4. Results Comparison using MBRT

From the graph it is clear that efforts decreases for both type of complexity tasks for a project under maintenance. This reduction mainly influences requirement understanding and test design phase efforts. It is also observed that post delivery defects on all of these support tasks are zero.

## VI. ADVANTAGES

Modules based regression selection technique offers the listed advantages but are not limited to the following

- Test coverage is high within the limited time frame and does not demand the resource to have complete understanding of the system
- Analyze the related impacts and come up the priority wise modules and scenarios (in turn test cases) to be executed. Thus estimation to validate the changes becomes more accurate
- Knowledge transfer time is less
- Provides input to the code-level impacts (Reverse engineering)

## VII. CONCLUSION

The Module based regression selection technique uses requirement database to derive the impacted modules and relevant scenarios to execute the identified test cases. The requirement database shall be incorporated as a phase in lifecycle of the project for an application that is developed from the scratch. For those projects that do not have any requirement database, it is suggested to use some agile practices to get a quick turnaround in developing the same. In this case, ROI need to be taken care (i.e. how long project will be in support mode and what is the frequency of testing the 'requirement changes'). With this information, received results have good degree of accuracy for quality deliverable in support projects.

## VIII. GLOSSARY

|      |                                   |
|------|-----------------------------------|
| MBRT | Module Based Regression Technique |
| DB   | Database                          |
| TS   | Test Scenarios                    |



ROI Return on Investment  
TCD Test Case Design  
FLC Full Life Cycle

#### ACKNOWLEDGMENTS

| Name            | Description   |
|-----------------|---|
| Venkatachalam V | Practice Head-Quality & Process, Honeywell Technology Solutions |
| Ravikumar B N   | Project Leader-Testing, Honeywell Technology Solutions          |

#### REFERENCES

- [1] Yanping Chen, Robert L. Probert, University of Ottawa, Ontario, Canada, "A Risk-based Regression Test Selection Strategy"
- [2] Yanping Chen, Robert L. Probert, University of Ottawa, Ontario, Canada, D. Paul Sims, IBM Canada Ltd., Canada, "Specification-based Regression Test Selection with Risk Analysis"
- [3] GREGG ROTHERMEL and SEBASTIAN ELBAUM, University of Nebraska—Lincoln, "On Test Suite Composition and Cost-Effective Regression Testing"
- [4] GREGG ROTHERMEL, Oregon State University and MARY JEAN HARROLD, Ohio State University, "A Safe, Efficient Regression Test Selection Technique"