Master Thesis
Software Engineering
Thesis no: MSE-2009-10
June 2009

ISTQB: Black Box testing Strategies used in Financial Industry for Functional testing

Umar Saeed
Ansur Mahmood Amjad

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Contact Information:
Author(s):
Umar Saeed
Email: chumarsaeed@yahoo.com

Ansur Mahmood Amjad
Email: ansurjajja@gmail.com

University advisor(s):
Kennet Henningsson
Department of System and Software Engineering

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

ABSTRACT

Black box testing techniques are important to test the functionality of the system without knowing its inner detail which makes sure correct, consistent, complete and accurate behavior or function of a system. Black box testing strategies are used to test logical, data or behavioral dependencies, to generate test data and quality of test cases which have potential to guess more defects. Black box testing strategies play pivotal role to detect possible defects in system and can help in successful completion of system according to functionality. The studies of five companies regarding important black box testing strategies are presented in this thesis. This study explores the black box testing techniques which are present in literature and practiced in industry as well. Interview studies are conducted in companies of Pakistan providing solutions to finance industry, which is an attempt to find the usage of these techniques. The advantages and disadvantages of identified Black box testing strategies are discussed, along with it; the comparison of different techniques with respect to most defect guessing, dependencies, sophistication, effort, and cost is presented as well.

Keywords: BBTS (black box testing strategies), BVA (boundary value analysis), EP (Equivalence Partitioning), DT (Decision Table), CT (Classification Tree), SD (State Diagram), UC (Use Case)

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

There are many products and services such as microwave ovens, cards, trains, buildings, bank services and mobile phones in our daily use. Software is major commodity of these products and services. The demand of software quality is increasing, as the software is becoming more and more important to us. With the passage of time, software products are growing larger and becoming more and more complex. It leads to more opportunities for defects to sneak in software development and its maintenance. Software takes a lots of resources of software development organization and they are interested to provide software in functional form as long time as possible. Due to changes in the environment and in the user requirements, it is very important that the software is easy to adapt and maintainable. The complex software is difficult to understand and reason about it. This offers considerable challenges for people developing and maintaining software.

The term debugging was introduced in 40s. At early stages, programming was considered that you "wrote", Turing[1] a program and then you "checked it out.", Turing[1] The terms debugging, program checkout and testing were mixed and not clearly differentiated. Debugging is an activity to get the bugs out. The term "Program Checkout", Turing [1] was introduced by Alan Turing in 1949. In his article he discusses the use of assertions for what we would today call "proof of correctness", Turing [1]. The second article of Alan in 1950 can be considered the first on testing, Reed [36]. In his article, he addressed the question "How would we know that a program exhibits intelligence?", Reed[36] If necessary to build such a program, then this question is case of "How would we know that a program satisfies its requirements", Reed [36] He introduced an operational test for intelligent behaviour by a computer program. In his test, Turing used the program and a human (reference system) to be indistinguishable to a tester (interrogator).

The differentiation between debugging and testing had been made in 1957. Charles Baker made this distinguish in a review, Hamlet et al [37] of Dan McCracken's book *Digital Computer Programming*. There were two goals of program checkout: "Make sure the program runs", Hamlet et al [37] and “make sure the program solves the problem.” Hamlet et al [37]. The focus of former was debugging and the latter as the focus of testing. The optimized term “make sure”, Hamlet et al [37] was then translated into testing whether system meets its goals or not. The terms “debugging” and “testing”, Hamlet et al [37] included the efforts to detect, locate, identify, and correct faults.

The formal definition of testing was introduced in 1979 by Myers [32] as “the process of executing a program with the intent of finding errors.” According to this definition, fault detection is primary goal. Myer’s goal was, to show that program has no faults; one might select test data which has a low probability of finding errors. If the goal is to find errors, one will select test data which have high probability of detecting errors and our testing succeeded successfully. Other people and Myers’s book; Deutsch and Miller et al [5, 6] discussed software analysis and review techniques along with testing. Deutsch and Howden [7, 8] wrote articles on fault detection approaches.

The guideline published in the Institute for Computer Sciences and Technology of the National Bureau of Standards in 1983, introduced a methodology which integrates analysis, review, and test activities to provide product evaluation during the software life-cycle, NBS FIPS [9]. Three evaluation techniques recommended; the basic, the comprehensive and the critical. A single testing technique cannot guarantee error-free software. A set of techniques can be helpful in development and maintenance of software.

Several type of testing techniques exist in comprehensive software testing process, many of which can occur simultaneously which can be named as unit testing, integration testing, stress testing, regression testing, system testing, quality assurance testing and user acceptance testing. Unit testing is involved combination of structural and functional testing by programmers. The individual components or units are combined with other units to make sure the necessary communications, links and data sharing. Integration testing handles testing of these components. Stress testing try to determine the failure point of a system under extreme pressure. Regression testing is concerned about conformation that implementation of changes have not adversely affected other functions. System level testing starts when enough modules are integrated to perform functions in a whole system environment. Quality assurance testing is performed by Quality Group to ensure organizations standards are being followed or not, original requirement is documented or not. User acceptance testing begins when the user's get their first crack or software.

There are two classes of software testing; Black box testing and white box testing, Williams [10]. The black box testing ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions; Williams [10].It is often used for validation of product, Williams [10]. One important point to note about black box testing is that it is done at functional and system level of testing when product design is at high level, Williams [10].

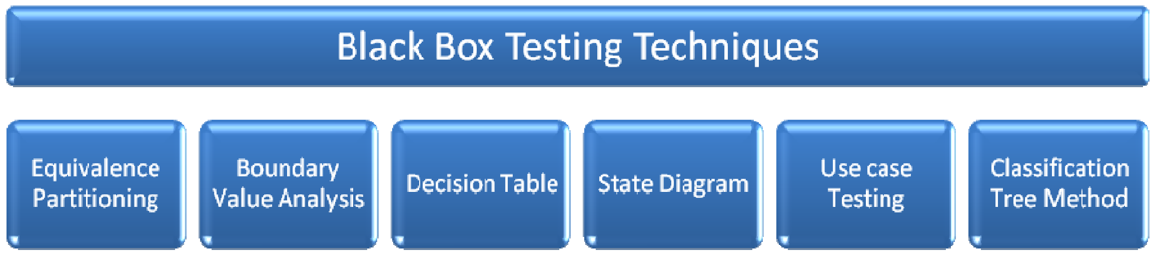


Figure 1.1 Important Black box Techniques

The black-box testing technique has grown in importance, Belli et al [13]. According to international software testing qualifications board, Williams [10], six test design techniques are most important for functional testing. They are equivalence partitioning, boundary value analysis; state diagrams, decision tables, use case testing and classification tree method. These six design techniques can also be seen as strategies of black box testing, Chen et al [11]. It is because the black-box techniques are used to select test cases up to minimum level, Chen et al [11].These six techniques also deal with defect guessing and exploratory testing, Ryber [12].

1.2 RESEARCH QUESTIONS

The main question which drives our thesis is:

Analyze and compare black box testing techniques from functional point of view?

As mentioned in Section 1.1, software is playing a vital role in different organizations. The software availability, reliability, usability and efficiency demand its increase. The continuous evolution in software is in progress due to change in requirements, standards, and so forth. It becomes difficult to maintain the quality of software. High quality software meaning according to IEEE definition of quality, IEEE [14]:

1. "The degree to which a system, component, or process meets specified requirements."
2. "The degree to which a system, component, or process meets customer or user needs or expectation."

Quality is a broader term that constitute on several aspects of the software which may all contribute to achieve high quality of software. For example, according to ISO, there are six characteristics in quality of software: functionality, reliability, usability, efficiency, maintainability and portability, ISO/IEC [15]. Thus, external (with respect to customer) and internal (with respect to project) attributes are involved. Each characteristic is further divided into sub-characteristics which show the broader concept of software quality.

The main question is concerned about analysis and comparison of aforementioned testing techniques from functional point of view. The functionality of system is specified in functional specification document. It is abstract level document. Difficulties exist to generate test case for piece of code before design. It is thus, need arises to generate functional test cases to check whether proposed system meets its functional specifications. Black box techniques help to design functional test cases.

This thesis addresses the above main research question. Research is conducted by performing different studies, case studies and empirical. These studies direct toward the main research question by addressing the following questions:

Q1: What are the most important black box testing strategies?

Q2: What are the advantages and disadvantages of important black box testing strategies?

Q3: What is the primary purpose of each of the black box testing strategy? Which serve more as exploratory testing and which serve more as defect guessing?

Q4: How complementary are various combinations of black box testing techniques? Is there any best possible combination?

Q1 is concerned to find out black box testing techniques which conforms the dimensions given in figure 1.2. The word ‘Important’ is centre of gravity for other terms. The black box testing techniques which are recommended by International Software Testing Qualifications Board (ISTQB) [71], and mentioned in its syllabus, ISTQB V 0.2 [72] are discussed in terms of defect finding, sophistication (Concerns about test cases which are useful to discover maximum defects and can be quantified in terms of that one), defect guessing, quality of test cases and effort. The ISTQB recommended techniques are established by experienced people related to testing field and have great exposure in respected field that’s why these techniques has been selected and will help to motivate authors research in respected country in which software field is growing. ISTQB helps to narrow down the focus of research instead of lemmatize it.

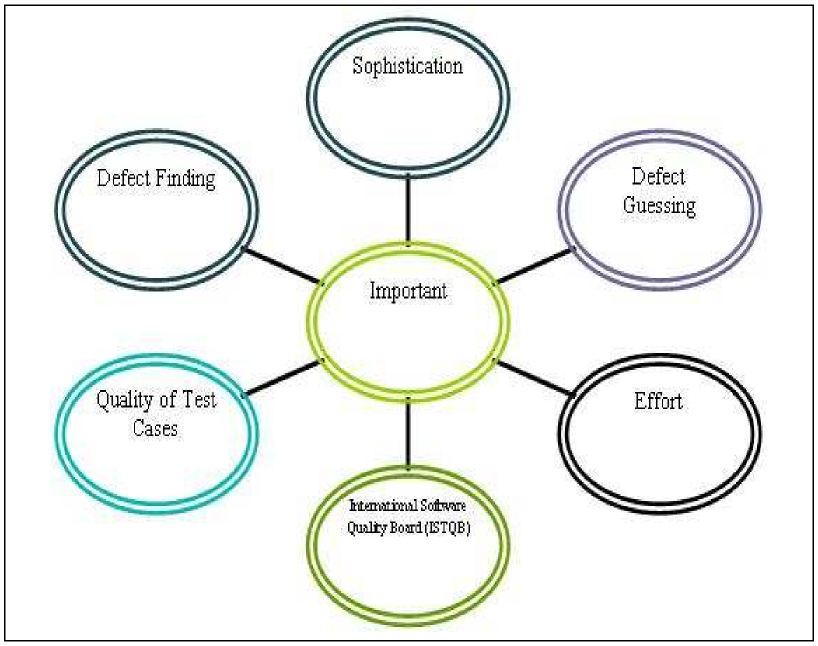


Figure 1.2 Dimensions of Important Black box Techniques

As there are number of black box testing techniques, objective is to filter these techniques based on functional testing perspective and dimension given in figure 1.2. The black box test case preparation started as the functional specification completed. But it is difficult to prepare all possible combinations of test cases and check the system against every test case. That’s why, a set of techniques exist, to find out classes of errors. E.g. Equivalence Partitioning is black box testing technique which divides the input domain of program into classes of data from which test cases are derived.

Q2 involves the understanding of black box testing techniques and to find out advantages and de-advantages of each testing technique. The black box testing occurs throughout the software development testing life cycle i.e. in Integration, Unit, Acceptance, System and regression testing stages. All types of errors cannot be found using only one black box testing strategy. Majority of bugs needs to be discovered. E.g. Equivalence Partitioning cannot find bugs on boundary. Boundary Value Analysis (BVA) choose the extreme boundary values where boundary values are maximum, minimum, typical values, error values and just inside/outside boundaries.

Q3 concerns about understanding of important black box testing strategies. It involves different studies to find out which technique is exploratory or just guessing technique. E.g. Equivalence Partitioning techniques just find out the classes of errors, on the other hand boundary value analysis increase testing scope to find out errors on boundary conditions with minimum, maximum and typical values etc.

Q4 involves analysis of black box testing techniques found in Q1. It concerns about how various testing techniques are complementary, what is their scope etc. E.g. The Boundary Value Analysis (BVA) and Equivalence Partitioning (EP) are complement to each other. BVA tries to extend the scope of EP. BVA forced on exception handling. It is type of robustness testing.

2 RESEARCH METHODOLOGY

Research is concerned about examining critically the various aspects of our day-to-day professional work, way of thinking, understanding and formulating guiding principles that govern a particular procedure , testing and developing new theories for the enhancement of practice. Research methodology is like finding a path in right direction. It helps to find answers of research questions and to achieve objectives. Following figure 2.1 shows a path of research.

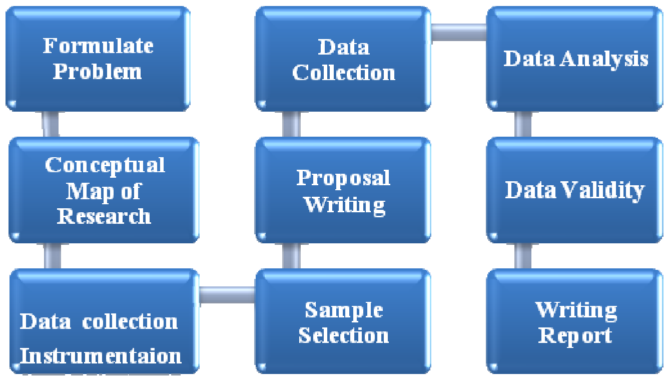


Figure 2.1 Research Process.

By following path of research given in figure1, authors devised research methodology. The Section 2.1 gives brief introduction of research strategies exist in literature, Section 2.2 describes research design, the purpose, aims and objectives and research questions are given in Section 2.3, 2.4 and 2.5 respectively.

2.1 RESEARCH STRATEGIES

Explorative study is conducted to understand a poorly investigated phenomenon, Robson [16]. Explorative studies, with the help of qualitative approach generate new ideas and ask more and new questions regarding the poorly investigated phenomenon at hand. Through qualitative research, real properties of real objects are investigated in their natural settings and environments. The findings are expressed in words instead of numbers, Robson [16]. The research is not fixed to predefined specification of how the investigation should progress during the enquiry in a flexible design but it may adapt to new insights gained during the study, Robson [16]. When complete vision has been gained and new questions and ideas arise, then more focused enquiry is undertaken, drilled down on a specific issue found promising in the explorative study. At this stage, quantitative research is taken; quantitative data is collected that is compared statistically. The quantitative approach helps to understand the relationships in the phenomenon quantitatively, e.g. to identify cause-and-effect relationships, Wohlin et al [17].

The qualitative and quantitative approaches come under umbrella of empirical research strategies. In empirical research a specific study may be categorized further with respect to its purpose. A study may be explorative, descriptive or explanatory, Creswell [21]:

- *Explorative Study.* It is conducted to gain insights and new views of the phenomenon. E.g. the open-ended questionnaire helps to get more information before a more thorough investigation is performed.
- *Explanatory Study.* Its purpose is to explain a relationship or a concept in the studied population. E.g. to explain why combination of black box testing techniques is preferred instead of single one.
- *Descriptive.* It classifies observations and presents the distributions of attributes about the phenomenon.

An empirical research may be explorative, explanatory or descriptive and qualitative or quantitative. On base of purposes and type of results expected, explorative research can be said to be more qualitative in its nature, while explanatory and descriptive research are more related to quantitative approaches.

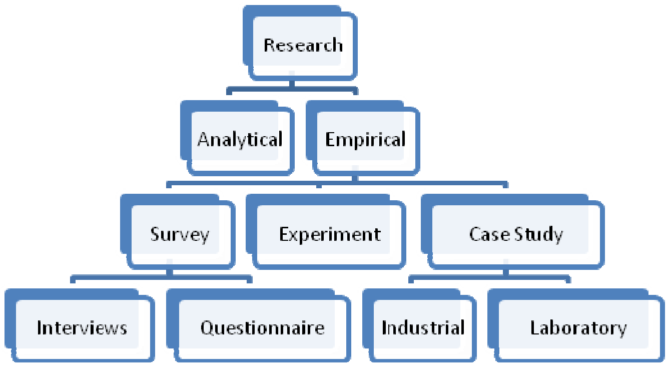


Figure 2.2 Research Methodologies [Derived from 17 and 21]

Software engineering involves the research methods, by applying these methods it is possible to make use of previously collected knowledge, techniques and experience to answer the research questions effectively. It depends on research questions which methods are more or less applicable. Empirical research is conducted by both qualitative and quantitative, experiments, surveys and case studies.

Survey:" It is a comprehensive system for collecting information to describe, compare or explains knowledge, attitudes and behaviour", Pfleeger et al [19]. Survey is conducted as interview or with questionnaires. It is conducted on a sample which is drawn from the population being studies and results are analyzed and conclusions are drawn.

Experiment: It is another empirical research method. Its concern is formal test to study a phenomenon, preferably in an actual environment. A formal experiment gives more control over the studied objects compared to a case study performed in a real-world organization, Juristo et al [20]. Description of an experiment is as follows: the objects are randomly drawn from the population of interests. Then individual objects are assigned randomly into one of two groups. After that the value of the independent factor variable is varied between the two groups while other variables are fixed or controlled. The measurement of the effects on the dependent variable in the two groups helps to draw a conclusion about the degree of cause-and-effect relationship between the dependent and the independent variables. An experiment is *randomized controlled experiment* when the sample is random and the assignment to groups is also random, Robson [16].

Case study: When the objective is to study a phenomenon in its real setting and environment then a case study is conducted. It is useful research method when there is no isolation of phenomenon from its environment or environment interacts with the phenomenon, Creswell [21]. Cases of small or larger type are collected in a case study and data is collected. Data can be quantitative or qualitative; it depends on purpose of the research. Analysis of quantitative data may be treated statistically and with qualitative data it may be treated with protocol analysis or other analysis techniques, Owen et al [22]. It may be conducted in a laboratory or in the industry as shown in above figure. In case of replicated case study environment, a *laboratory case study* is conducted, e.g. to study the distribution of defects in a research projects where project is installed and executed in the researcher's own laboratory.

2.2 SAMPLING STRATEGIES

A population subset is called a sample which is used to study participants. The selection of a portion of the population in research area which represents the whole population is called Sampling. The plan is the strategy set forth to be sure that the sample used in research study represents the population from which a sample is drawn, Denszin [73]. The following terms are associated with sampling: sample, sampling frame, population, eligibility criteria, inclusion criteria, exclusion criteria, representative ness, sampling designs, effect size etc. There are two sampling design strategies, Denszin [73]:

Probability Sampling: The elements are selected randomly where greater confidence is placed in the representative ness of probability samples. The selection process gives an equal chance to each element in the population to be selected. Main methods are: simple random, stratified random, cluster and systematic.

Non-Probability Sampling: The elements are selected using non-random way. These sampling produce less likely representative samples than probability samples but researchers can and o used non probability samples and these samples can not be generalized statistically. Main methods are: convenience, quota and purposive. Convenience or hap haphazard sample is selected at the convenience of the researcher. It is useful for formulative studies, pilot surveys, exploratory studies, testing questionnaire, pre-test phase. E.g. The selection of sample from a basked of apples The results collected after Non-Probability sampling can not be generalized statistically but conclusions can be drawn on base of assumptions, some parameters, observations, situations, domain etc after that results can be generalized theoretically. These results can match with any company related to studied domain but meet aforementioned variables.

2.3 RESEARCH DESIGN

In this thesis, empirical studies have been conducted. Authors choose interview as research strategy because authors can achieve their objectives at best level that's why it is best candidate for our study. As authors focused on organizations of Pakistan, it is not possible for us to do experiment and case study because they require actual environment with real settings. This Section describes the research approaches, data collection techniques and research methods used for the studies presented in this thesis as shown in following table 2.1

Table 2.1.Formulated Research Methodology and Approaches

Phase	**Strategy**	**Approach**	**Purpose**	**Method**	**Comment**
1	Empirical	Qualitative	Descriptive	Testing Analysis	Software Testing Process
2	Empirical	Qualitative	Explanatory	Literature Review	Software testing Techniques
3	Empirical	Qualitative	Exploratory	Interview Design	Non-Probabilistic Sampling, Questionnaire, Telephone etc
4	Empirical	Qualitative	Explanatory	Interview Findings	
5	Empirical	Qualitative	Explanatory	Data Validity	Standard Method

Phase 1 Software testing, its importance, objectives and principles are analyzed. How it is related to verification and validation. Software testing process like white box, glass box or black box and its different phase is analyzed. The analysis, related to software testing mentioned above is present in Chapter no 3, named as "Software Testing".

Phase 2 It involves exploration of different black box testing techniques in context of functional testing which is in depth analysis, comparison of testing techniques like Boundary Value Analysis is complementary to Equivalence Partitioning and what are their pros and cons. It is discussed in Chapter 4 under heading "Black Box Testing Techniques for Functional Testing". To gather all stuff discussed above which is involved in this phase, authors used different databases and resources available free of cost and provided by Blekinge Institute Library for study and research purposes of students. Mostly used resources are: ACM digital library, Engineering Village, e-bray, Google search Engine, Google Scholar, Google books and IEEE Explore.

Phase 3 It encompass on Chapter no 5 under heading "Interview Design" which explain interview design process and answer the question "How it is conducted?". Interview design consists on series of steps required to follow for conducting interview. Non probabilistic sampling strategy which is convenient sample, Harison [70] has been used to select companies and sample for conducting interview. The companies which have been selected for study are related to financial domain. Those companies provide solutions in various financial domains like e-banking, transactions management, accounting etc which shows that these companies are representative of authors study.

Phase 4 The findings as a result of study 3 are presented in Chapter 6 under heading "Interview Findings". Data and information gathered by interview is presented in tabular and graphical form.

Phase 5 It entails the validity and analysis of data gathered in study 4 which is present in Chapter no 7 under heading "Data Validity". The advantages and disadvantages, and difference of each black box testing technique with respect to renowned Pakistani software companies are given in this Chapter.

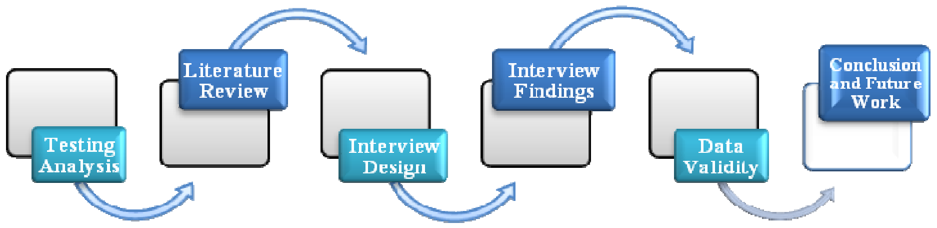


Figure 2.3 Research Process for our study.

2.4 RESEARCH PURPOSE

The research purpose is to analyze testing process and to conduct exploratory and explanatory studies to explain different black box testing techniques for functional testing which produce minimum test cases and produce more defects. It investigates the pros and cons of black box testing strategies, how much complementary are they each in terms of effort, time and sophistication. To be more practical research, interviews conducted from software architects, Quality Assurance managers, Product / Project managers, QA specialists, Test Case designer and testers of different software organizations related to finance domain like Accounting, Banking etc in Pakistan. This research can be helpful for organizations to learn from other's experiences in testing of financial applications.

2.5 AIMS AND ONBJECTIVES

The aim of intended study is to analyze the black box testing strategies used in industry which provides financial solutions e.g. payment card industry, financial accounting etc.

i. Study software testing process, testing strategies, black box testing strategies.

ii. Study primary purpose of each important functional testing strategy.

iii. Study *pros* and *cons* of functional testing strategies.

iv. Study testing strategies from complementary point of view.

v. Take interview from concern people.

vi. Study testing process of organization deeply

vii. Analyze different team structure for functional testing.

viii. Analyze their functional test design strategies.

ix. Analyze significance of important black box testing strategies

x. Analyze strategies used in rigid time deadline for functional testing of critical modules

2.6 RESEARCH QUESTIONS

In this study, researcher will answer one main question and four sub questions which are given as follows:

Q. In context of manual testing, analysis and comparison of black box testing techniques in finance industry from functional point of view?

Q1 What are the most important black box testing strategies?

Q2 What is the primary purpose of each of the black box testing strategy?

Q3 What are the advantages and disadvantages of important black box testing strategies?

Q4 How complementary are various combinations of black box testing techniques? Is there any best possible combination?

3 Brass-Tacks of Software Testing

Software testing is referred to *verification and validation.* Verification is set of activities which ensure that a software function is implemented correctly. Validation refers to set of activities which ensure that whether software is traceable to customer requirements. According to Bohem [23]: Verification: 'Are we building the product right?' and Validation: 'Are we building the right product?'

3.1 SOFTWARE TESTING---WHY AND WHAT

3.1.1 Why Quality Assurance

Quality assurance (QA) refers to set of activities which ensure that software meets its quality requirements. It provides data to higher management to gain insight, software meets its goals. The activities required to achieve quality are visualized in following figure 3.1:

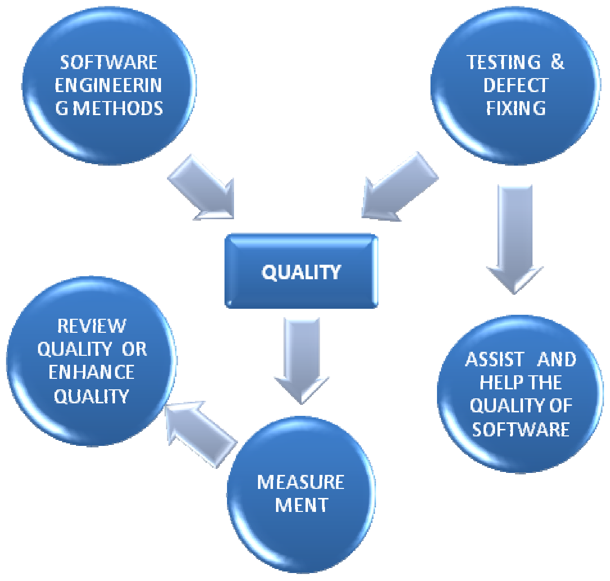

SOFTWARE ENGINEERIN G METHODS
TESTING & DEFECT FIXING
QUALITY
REVIEW QUALITY OR ENHANCE QUALITY
ASSIST AND HELP THE QUALITY OF SOFTWARE
MEASURE MENT

Figure 3.1 Sub set of quality activities

Walkthroughs or Formal Technical Reviews are useful to ensure the quality of work products produced as a consequence of each software engineering step. As a last bastion testing provides quality assessment more pragmatically. Testing does not show the presence of quality. The Miller describes the relationship of testing and quality assurance by stating that "*The underlying motivation of program testing is to affirm software quality with methods that can be economically and effectively applied to both large scale and small scale systems, Miller* [24]"

Verification & Validation encompass a wide array of QA activities such as formal technical reviews, quality and configuration audits, performance monitoring, simulation, feasibility study, algorithm analysis, documentation review, database review, development testing, qualification testing and installation testing, Wallace et al [25]. Although testing plays a vital role in *V & V* but other activities are also necessary to achieve high quality.

3.1.2 WHAT IS THE IMPORTANCE OF SOFTWARE TESTING

The software testing implications and importance with respect to software quality cannot be overemphasized *"The development of software systems involves a series of production activities where opportunities for injection of human fallibilities are enormous. Errors may begin to occur at the very inception of the process where the objectives… may be erroneously or imperfectly specified, as well as in later design and development stages… Because of human inability to perform and communicate with perfection, software development is accompanied by a quality assurance activity, Deutsch et al [26]"*

3.1.3 WHAT ARE THE OBJECT IVES OF TESTING

According to Glen Myers [27], there are following testing objectives:

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding an as-yet undiscovered error.
3. A successful test is one that uncovers an as-yet undiscovered error.

If testing is conducted according to above stated rules then it will uncover errors in the software. Software works according to its requirements and meets performance requirements then it is an indication of high software reliability. Testing cannot guarantee that software has no errors or defects

3.1.4 WHAT ARE THE TESTING PRINCIPLES

Software testing is always guided by testing principles. Davis [28] suggests following principles:

1. *All tests should be traceable to customer requirements.*

   The most sever defects are those that cause the program to unmeet customer requirements.
2. *Tests should be planned long before testing begins.*

   As soon as requirement model is completed then test planning should begin. After the solidification of design model, detailed design of test cases should begin.
3. *The Pareto principle applies to software testing.*

It is 80 and 20 rule. The 80 percent of all errors uncovered. Only 20 percent errors are traceable.

4. *Testing should begin 'in the small' and progress toward testing 'in the large'*
   The test plan and execution should begin from individual program modules. As testing progress, its focus shifts to find errors in integrated clusters of modules and ultimately in the entire system.
5. *Exhaustive testing is not possible.*
   It is impossible to test every path. On the other hand it is possible to uncover program logic and to ensure that all conditions in the procedural design have been exercised.
6. *To be most effective, testing should be conducted by an independent third party.*
   You see what your eyes want to see. When you test a module, try to show that it is functioning perfect. A number of defects are uncovered. That's why; testing should be conducted by third party.

3.1.5 WHAT IS TESTABILITY

It concerns how easily a computer program can be tested. How a set of tests cover the product adequately? The testable software has following attributes, Pressman [31]:

- *Operability:* "The better it works, the more efficiently it can be tested." There exist no such types of bugs which block the execution of tasks. Few bugs exist in the system.

- *Observability:* "What you see is what you test" The unique output is generated for each input. System state is observable during execution. Incorrect output is easily identified.

- *Controllability:* "The better we can control the software, the more the testing can be automated and optimized" Complete output is generated through some combination of input. Some combination of input executes the whole code. Tester have control over system variables, states etc.

- *Decomposability:* "By controlling the scope of testing, we can more quickly isolate problems and perform smarter retesting." The software constitute on independent modules and these modules are testable

- *Simplicity:* "The less there is to test, the more quickly we can test it." The software have functional simplicity; minimum set of features meets all requirements, Structural

Simplicity; module or layer based architecture, Code Simplicity; during development coding standard is adopted for ease of inspection and maintenance.

- *Stability:* "The fewer the changes, the fewer the disruption to testing." The changes to software are controllable, infrequent and do not invalidate existing tests.

- *Understand ability:* "The more information we have, the smarter we will test." The system designs, dependencies between internal, external and shared components are well understood. Technical documentation is well understood, instantly accessible and accurate.

3.1.6 WHAT IS SOFTWARE TESTING PROCESS

Software comprise on a no of components which have complex structure. It is hard task to test them as single, monolithic unit. In object oriented system, object encapsulates data and operations. A group of classes are combined to form service. At abstract level, object oriented system is collection of components collaborating to each other to generate a response to a particular input or set of input events. In agent based systems, an agent is autonomous entity which has capability to perceive environment, perform operations and give output. A comprehensive software testing process has several type of testing which are known as *Unit Testing, Integration Testing, System Testing, Stress Testing, Regression Testing, user acceptance tests and quality assurance testing, Pressman [31].*

- **Unit Testing:** It involves some combination of functional and structural tests which is done by programmer in their systems. Component testing is aided through unit testing frameworks.

- **Integration Test:** After completing development of individual components, their integration started to make sure that links, data sharing and necessary communications occur properly. It is different from system testing because it is not in operational environment. Integration is very crucial; it needs proper planning and subset of production-type data. Basic integration methods are three:

  Top Down: Top-Down testing build an initial skeleton that fills individual completed modules. It is fit to prototyping environment and lends to more structured organizations that plan out the entire test process. Interface errors are found earlier whereas errors in critical low-level modules can be found later. High level top-down modules provide an early working program that gives management and user more confidence in results early on in the process.

Bottom Up: Individual modules are tested by using driver routine which calls the module and provides it with needed resources. Bottom up testing is not suitable for structured shops because there is high dependency on availability of other resources to accomplish the test. Earlier bug finding is done in critical modules. The interface errors surface late in the process.

All At Once: It addresses simple integration problems, involving a small program possibly using a few previously tested modules.

- **System Test:** The system testing is performed after complete integration of modules in operational environment. It can occur in parallel with integration test, especially with the top-down method.

- **Stress / Performance Test:** Important phase of system test is load, volume or performance test. Performance test try to find breakpoint of a system under extreme conditions. They are useful when systems are being scaled up to larger environments or being implemented for the first time. Transaction management systems like in ATM (Automatic Teller Machine) card application or web sites require processing, multiple access and high TPS (transaction processed per second) rate. Stress testing stimulates loads on various points of the system and cannot stimulate the entire network as user experienced. Stress testing is performed at once if it completed successfully because in case of any changes in program, it need to rerun those tests. Performance testing conform the performance of system but not correct functionality of system. In case of high TPS (transaction per second) rate, much more chances of data corruption and liability than simply stopping or slowing the processing of correct transactions, Beizer [30].

- **Regression Testing:** Conformation of whether implementation changes adversely affected other functions or not is done by regression testing. It is applicable to all phases whenever change is made

- **Quality Assurance Test:** Quality groups exist in different a organization which provides a different point of view, applies tests in a different more complicated environment, and uses a different set of tests. This group conforms whether organization standards have been followed in coding and documentation of software. They verify software is properly implemented according to specifications and see things are ready to user to take a crack at it.

- **User Acceptance Test and Installation Test:** It is first stage where users 'get their first crack' at the software. If user have not been involved with the design, not seen prototypes and not understood the evolution of the system, they are inevitably going to be unhappy with the result. If every test is user acceptance test then there are more chances of a successful project.

3.2 TYPES OF SOFTWARE TESTING

The user views the objects from two angles: one from inner side and other from outer side. On bases of user view testing have two types: white box testing and black box testing.

3.2.1 WHITE BOX TESTING

White Box tests tries to verify internal working of the software on base of complete knowledge about object source code. It is also known as structural testing because it checks the structure of software, Sommerville [29]. Structural testing makes sure that is program structure contributes to efficient and proper program execution? Well designed program: good control structure, sub-routines and components followed by good programming practices and skills are good.

Majority of defect classes are identified by applying code inspection or walk-through and classic structural test. The proofreading of codes is called inspection or code review which tries to find the author mistakes, the 'typos' and logical errors. Debugging tools facilitate white-box testing.

White Box testing has also disadvantages because code inspection is not *piece of cake* because code inspection demand highly skilled technicians in tools, environment and language. Distributed and Agent based system are not comprise on one program, so correct program might call another program which provides bad data. In large systems, the program execution path which is series of calls, input and output and structure of common files is important. The testing employed on this type of system at intermediate or integration stages is hybrid testing

3.2.2 Black Box Testing

Behaviour of software is examined by functional testing as evidenced by its output without reference to internal functions. By analogy it is sometimes called black box testing, Sommerville [29]. The source code features are irrelevant if the program consistently provides the desired features with acceptable quality. Black box testing is suitable for modern programming paradigm like object oriented paradigm and agent paradigm etc.

Black box testing focuses on quality of system. People target on functionality of system whether it meets their needs or not. Quality criteria are developed in the beginning. It have scientific basis, like that functional tests must have a hypothesis, a defined method or procedure, standard notation to record the results, and reproducible components. These tests are reusable after making changes to conform changes only produce intended results or not.

4 BLACK BOX TESTING TECHNIQUES FOR FUNCTIONAL TESTING

Black box or functional testing is used to test a metaphor or system's functionality with out knowing its inner detail. It is method of test design and it is applicable to all levels of software testing: system and acceptance, functional testing, integration and unit testing.

In this chapter, black box testing strategies like boundary value analysis, equivalence partitioning, decision table, state diagram, use case and classification tree are discussed from functional point of view. The primary purpose of each technique is explored; its scope is examined, explained by different examples which show their strength. The pros and cons of each technique are discussed from different perspective like most defects guessing, defects catching, dependencies (logical or data), time, effort and cost. Almost quantitative analysis of each technique is done in terms of number of test cases it generates. Various formulas are derived to calculate number of test cases. The strength of each technique is discussed for robust, exploratory testing. The suitable combination of techniques is also discussed to do more effective testing and to know that how much complementary they are. On base of above mentioned criteria results are discussed in form of analysis and conclusion form. Authors suggested a new test case design strategy for use case testing of business critical applications in finance industry.

4.1 BOUNDARY VALUE ANALYSIS BASED TESTING

Boundary value analysis is a black box testing techniques to identify test cases. This technique follows a fundamental assumption that the majority of program errors will occur at critical input/output boundaries, points where the mechanics of calculation or data manipulation must change with an objective for program to produce a correct result, Jorgensen [32].

The 'Typing' of languages and boundary values analysis are associated to each other. PASCAL and ADA are strongly typed languages that required all variables or constants defined must have an associated data type, which dictates the data ranges of these values upon definition, Cardelli [33]. A logical reason is of putting such constraint to prevent the nature of errors that BVA is used to discover. Even though, BVA is not ineffective when used in conjunction with languages of this nature. The systems created using strongly typed languages are not suitable for BVA. Systems developed using weekly typed 'FREE-FORM' languages like FORTRAN and COBOL is suitable candidate for BVA. The free form languages allow one type to be seen as another e.g. A String is an Integer. It cause bugs and these bugs are normally found in the ranges that BVA operates in and can catch these errors.

The function input variables are targeted by BVA. Figure 4.1 shows, two variables x_1, x_2 and their ranges:

$$\text{E.g.} \quad F(x_1, x2) = 2x_1+x_2^2$$

$$A<= x_1<=B \ (x_1 \text{ lies between A\&B})$$

$$C<= x_2 <=D \ (x_2 \text{ lies between C\&D})$$

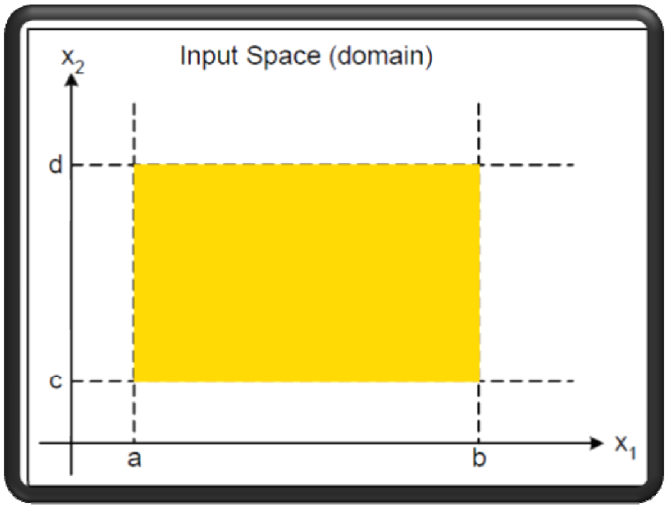


Figure 4.1 Sample Boundary Value Analysis Baselines for two Variable

The shaded area shows valid input domain of above function. BVA motivation is that errors tend to occur near the boundaries of the input variables. The defect possibilities can be countless but many common faults which result in errors more collated towards the boundaries of the input variable. For example in assembly language, the erroneously usage of byte variable instead of word variable can be checked by boundary value analysis.

4.1.1 TEST CASE SELECTION

The reliability requirements of software under test and the underlying assumptions on the likelihood of single versus multiple range checking faults, determines the number of test case.

The discussion related to single-variable and multi-variable BVA are derived from the boundary value analysis taxonomy presented in, Jorgensen [32].

4.1.1.1 USING SINGLE VARIABLE

The identification of boundary value typically from the input point of view, is baseline procedure for BVA. These boundary values are incorporated into the set of test cases as well as values near to boundaries. The boundary-adjacent values are helpful to exercise the program's bounds-checking logic. E.g. the testing of range of a value in a loop statement or a branching, the developer may use '<' a less-than operator when the correct operator should have been '<=' a less-than-or-equal to operator or '>' a greater than operator which is adjoining to the less-than operator on most keyboard layouts. These mistakes cause logical errors; programs containing logical errors compile but produce incorrect results. To catch these types of errors, values adjacent to the boundary values must be included in the set of test cases. The base line BVA procedure includes some nominal value of input or output in the set of test cases in addition to boundary and boundary-adjacent values.

Consider a procedure which takes one input variable K that has defined output only for the values of K in the range x<= K <=y. At minimum, the set of test cases selected would be a set of values with baseline [x, x+, z, y-, and y] where x+ is value greater than x, y- is value less than y and the value z is nominal value which lies between x+ and y-. This BVA baseline procedure identifies five test cases. The selected test cases are within range as shown in following graph by shaded area.

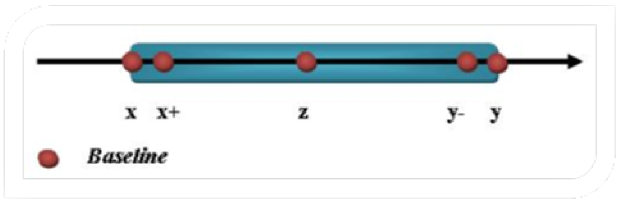


Figure 4.2 Boundary Value Analysis Baselines for Single Variable And Single Range [derived from 32]

The systems where error handling is critical such as nuclear reactor control, space craft or missiles etc, robustness (amplification) tests (values outside the allowable range) are included in BVA baseline procedure. The baseline tests identified above incorporated with the values [x- and y+] where x- is a value just below the minimum acceptable value x and y+ is a value just above maximum acceptable value y. The new identified test cases should force execution of any exception handler or defensive code. There are total seven test cases in single-input, single-range [x- , x, x+, z, y-, y, y+] example as shown in figure.

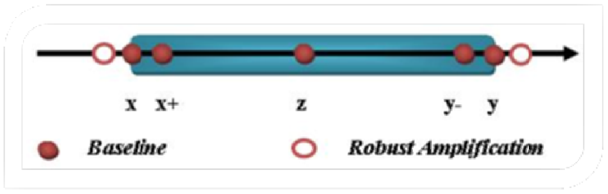


Figure 4.3Amplification of BVA baseline test cases, for Single Variable And Single Range [derived from 32].

In case of multiple sub-ranges for single variable, baseline and robust BVA procedures may also be applied. Consider a single input variable L with two adjacent sub-ranges where range 1 is a<= K <b and range 2 is b<= K <=d. The set of test cases will be the union of two sets of test cases for range1 and range 2 respectively, is given by the following:

$$L_{baseline} = \{a, a+, e, b-, b\} \text{ U } \{b, b+, f, d-, d\} = \{a, a+, e, b-, b, b+, f, d-, d\}$$

The amplification or robustness of BVA baseline for multiple ranges can be increased by including extreme values {a- , d+}. The union of two set of test cases increases number of test case. The baseline BVA identifies nine test cases and robust BVA identifies eleven test cases.

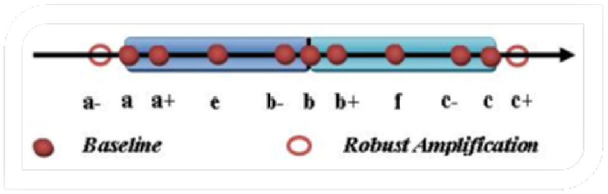


Figure 4.4 Amplification of BVA baseline test cases for Single Variable and Multi Range [derived from 32].

4.1.1.2 USING MULTIPLE VARIABLE

For multi-variable problems, BVA test case selection procedure also requires consideration of fault likelihood or fault model. The single fault model assumed that a failure is the result of a single fault due to the low probability of two or more faults occurring simultaneously, Turing [1]. On the other hand, multi-fault model assumes that the likelihood of multiple simultaneous faults is no longer insignificant, and thus additional test cases must be identified to address situations such as erroneous range checking on multiple variables simultaneously, Turing [1]. Suppose the single fault model for a problem has two inputs, X and Y, values of X in the allowable range a<= X <=c and where allowable values of Y span range given by d<= Y <=f. The baseline single variable test cases identified for X and Y respectively are the following:

$$\boldsymbol{X}_{\boldsymbol{baseline}} = \{a, a+, b, c-, c\}$$

And

$$\boldsymbol{Y}_{\boldsymbol{baseline}} = \{d, d+, e, f-, f\}$$

Multi-variable BVA test cases are selected on base of single-fault assumption which exercise the boundaries of one variable while the other variables are held at a nominal value. The final set of test cases is the union of all test cases identified as this procedure is applied to each individual input in turn. The following procedure shows the application of above procedure: the variable Y is fixed to its nominal value 'e' while variable X varies in $\boldsymbol{X}_{\boldsymbol{baseline},}$ similarly variable X is fixed to its nominal value 'b' while variable Y varies in $Y_{\boldsymbol{baseline.}}$ The total no of test cases identified by applying this procedure are 13, robustness testing adds only 4 test cases.

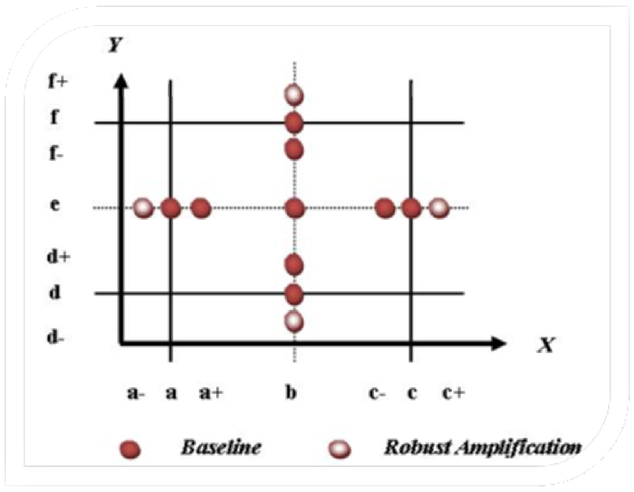


Figure 4.5 Single Fault, Baseline, and Robust Test Cases assuming X and Y are at its nominal value. The set of all test cases identified [derived from 32]

Multiple fault assumption increases number of test case, to detect multiple, simultaneous faults such as erroneous range checking on two variables at the same time. It starts from identification of baseline test cases if bounds checking are not critical or robust test cases if bounds checking are a high priority. Under multiple fault assumption, BVA test cases include the Cartesian product of Xbaseline X Ybaseline.

Given two sets X and Y, the Cartesian product of X and Y is defined as follows:

X x Y = {(x, y) | x ε X ^ y ε Y}

Where (x, y) denotes an ordered pair [37]. In other words, X x Y is the set that consists of all possible ordered pairings of an element from set X with an element of set Y. So, if set X contains i elements and set Y contains j elements, the resulting set X x Y will contain a total i*j total elements. The baseline and robust BVA test case identified for our problem under multiple fault assumption models are shown in following figure. The significant increase in the total number of tests identified. Twenty five baseline test cases were identified for this problem plus an additional twenty four for worst-case robustness testing.

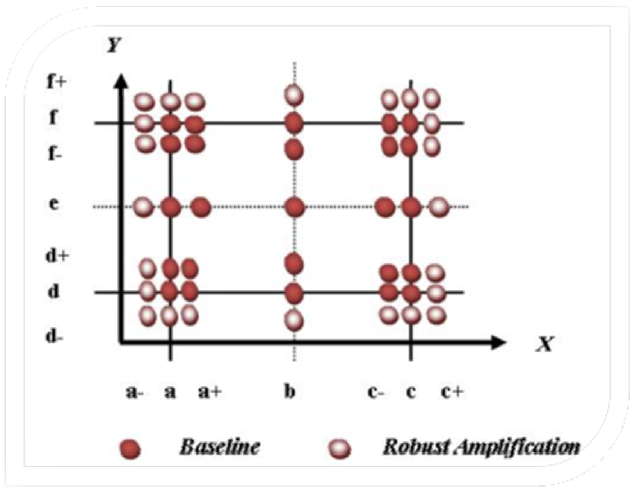


Figure 4.6 Multi-Fault, Baseline and Robust test [derived from 32]

The following table 4.1 sums up the number of test cases identified under reliability requirement and fault assumptions. The multiple fault assumption significantly increases the number of test cases if variables have single range.

Table 4.1 Test cases identified for two variables after applying BVA

Number of test cases		Fault Model	
Reliability Requirement		Single Fault	Multi-Fault
	Baseline	9	25
	Robust	13	49

4.1.2 ANALYSIS

Boundary Value Analysis process can be done in a uniform manner. It maintains one of the variables at their nominal or average level and allows remaining variables to take on its extreme values. To test extremities, following types of values are used: Min (Minimal), Min+ (Greater than Minimal), Nom (Average), Max- (less than Maximum) and Max (Maximum). Using example shown in figure 4, possible no of test cases which can be generated is given:

'X' at its nominal value

$\{\{X_{nom}, Y_{min}\}, \{X_{nom}, Y_{min}-\}, \{X_{nom}, Y_{min}+\}, \{X_{nom}, Y_{max}\}, \{X_{nom}, Y_{max}-\}, \{X_{nom}, Y_{max}+\}\}$

'Y' at its nominal value

$\{\{Y_{nom}, X_{min}\}, \{Y_{nom}, X_{min}-\}, \{Y_{nom}, X_{min}+\}, \{Y_{nom}, X_{max}\}, \{Y_{nom}, X_{max}-\}, \{Y_{nom}, X_{max}+\}\}$

Why we are concerned with only one of the values taking on their extreme values at any one particular time, its main reason is that generally BVA uses *Critical Fault Assumption.*

4.1.2.1 KEY EXAMPLES

To explain the necessity of certain methods and their pros authors will introduce two testing examples proposed by Jorgensen [32].It will be helpful to show where certain testing techniques are required and provide a better overview of the methods usability.

Problem1. Calculate NextDate.

The NextDate function takes three parameters: year, month, and day and returns the date of the day after that of input. The input parameters have following constraints:

Constraints

$$\{\{1\leq \text{Day} \leq 31\}, \{1\leq \text{Month} \leq 12\}, \{1812\leq \text{Year} \leq 2012\}\}$$

To keep no of test cases limited, year has been restricted. Further constraints can be in form of variable dependencies e.g. there is a never 31st of June no matter what year we are in. Due to these dependencies, this example is useful for us.

Problem2. The Triangle

The triangle problem was introduced by Gruenburger in 1973. It is most famous problem in testing literature. The triangle function accepts three parameters a, b, and c of integer type which represents the sides of a triangle. On base of these values the type of a triangle is checked whether it is *Equilateral, Isosceles*, *Scalene* or *not a triangle.* It must obey following constraints:

Constraints:

$$\left\{ \{1 \le a \le 150\}, \{1 \le b \le 150\}, \{1 \le c \le 150\}, \{a < b + c\}, \{b < a + c\}, \{c < a + b\} \right\}$$

If it does not satisfy above constraints then triangle is declared not a triangle. The types of triangle are determined as follows:

- If values of three sides are equal, the triangle is Equilateral.
- If values of exactly one pair of sides are equal, the output is Isosceles.
- If values of no pair of sides are equal, the output is Scalene.

4.1.2.2 RELIABILITY THEORY

Reliability theory states fault assumption such as single fault assumption or multiple fault assumption. Single fault assumption is also known as Critical Fault Assumption. On base of this assumption we can reduce the no of test cases dramatically. Under single fault assumption, baseline and not robust test cases, the total no of test cases generated for example shown in figure 4 are nine. The function *f (n)* which computes the number of test cases for a given number of variables n can be shown as:

$$f(n) = 4n+1$$

4.1.2.3 GENERALIZATION

The boundary value analysis can be generalized by two ways: generalize the number of variables or the ranges of these variables. The variable generalization is easy and simple. We do this by Critical Fault Assumption. Range generalization depends on the type of the variable e.g. the NextDate problem proposed by Jorgensen [32], have variable for the year, month and day.

In FORTRAN the month variable is encoded so that January corresponds to 1 and February corresponds to 2 etc. In Java or other languages, it would be possible to declare an enumerated type {January, February, March… and December}. This type of declaration is simple because ranges have set values. When explicit bounds are not given then we have to create our own which are known as artificial bounds and can be illustrated by using Triangle problem. According to Jorgensen [32] discussion, we can easily impose a lower bound on the length of an edge for the triangle as the negative length edge would be "ridiculous". It is problematic to set upper bound on length of each side, the possibilities can be set certain integer value, and allow the program to use the highest possible value of integer, long variable. The arbitrary nature of problem can lead to non concise test cases, to messy results etc.

4.1.2.4 LIMITATIONS

If the Program under Test (PUT) is "function of several independent variables which represents bounded physical quantities" then Boundary Value Analysis works well, Jorgensen [32]. Under good conditions it works well if conditions are not good then we can find deficiencies in the results. E.g. in NextDate function, where BVA would place an even testing rule equally over the range. The tester's intuitions and common sense shows that we require more emphasis towards no of days in a month, end of February or on leap years. Due to its poor performance or nature the BVA can compensate or take into consideration the nature of a function or dependencies between variables. The lack of understanding or inkling for the variable nature meant that BVA can be seen as quite immature.

4.1.3 ROBUSTNESS WORST CASE TESTING

4.1.3.1 ROBUSTNESS TESTING

Robustness testing is an extension of Boundary Value Analysis which encompasses the idea of running sparkling and grubby test cases. By sparkling we mean input variables which lie in the legitimate input range and by grubby we mean using input variables which fall just outside the input domain. The aforementioned five testing values (Min, Min+, Nom, Max-, and Max) are extended by adding two more values for each variable (Min-, Max+) which shows values outside the input range. As there are four extreme values for any variable. The addition of constant 1 represents nominal value. The function *g (*n) computes the number of robust test cases for given number of variables n can be shown as:

$$g\ (n) = 6n+1$$

As there are four extreme values and two robust values for any variable which accounts for 6n. As previous interest lied in the input to the program, the robustness testing ensures a sway in interest. Its main focus is on expected output when input variable has exceeded the given input domain. E.g. in NextDate function when we pass parameter 31st September we would expect an error message to the effect of 'Invalid date, it does not exist. Please try again'

Due to robustness testing desirable property, it forces attention on exception handling. In strongly typed languages, robustness testing is somewhat awkward where it can show up altercations. In FORTRAN or PASCAL if a value has predefined range and values that falls outside that range result in the runtime errors that would terminate any normal execution. Due to this reason exception handling mandates Robustness Testing.

4.1.3.2 WORST CASE TESTING

BVA assumes critical fault and tests only single variable at a time by considering its extreme values. By disregarding this assumption we can test the outcome if more than one variable were to assume its extreme value. In the field of electronic circuit this called Worst Case Analysis in which above stated idea is used to create test cases. By taking Cartesian product of 5-tuple set (Min, Min+, Nom, Max-, and Max) for multi-variables we can generate test cases which results into a much larger set of results than we have seen before. For example, two variables X and Y and their corresponding 5-tuple sets are given respectively:

$$\{X_{min}, X_{min}+, X_{nom}, X_{max}-, X_{max}\} \text{ X } \{Y_{min}, Y_{min}+, Y_{nom}, Y_{max}-, Y_{max}\}$$

Cartesian product of above sets is: $\{\{X_{min}, Y_{min}\}, \{X_{min}, Y_{min}+\}, \{X_{min}, Y_{nom}\}, \{X_{min}, Y_{max}-\}, \{X_{min}, Y_{max}\},\{X_{min}+, Y_{min}\}, \{X_{min}+, Y_{min}+\}, \{X_{min}+, Y_{nom}\}, \{X_{min}+, Y_{max}-\}, \{X_{min}+, Y_{max}\}\{X_{nom}, Y_{min}\}, \{X_{nom}, Y_{min}+\}, \{X_{nom}, Y_{nom}\}, \{X_{nom}, Y_{max}-\}, \{X_{nom}, Y_{max}\},\{X_{max}-, Y_{min}\}, \{X_{max}-, Y_{min}+\}, \{X_{max}-, Y_{nom}\}, \{X_{max}-, Y_{max}-\}, \{X_{max}-, Y_{max}\}\{X_{max}, Y_{min}\}, \{X_{max}, Y_{min}+\}, \{X_{max}, Y_{nom}\}, \{X_{max}, Y_{max}-\}, \{X_{max}, Y_{max}\}\}$.

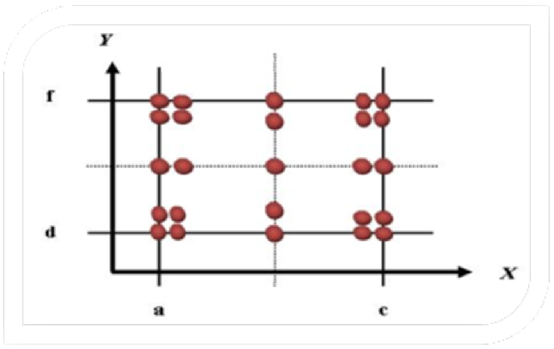


Figure 4.7 Worst Case Test Case for function of two variables.

From Cartesian product and figure 4.7 we can see that worst case testing is a more comprehensive testing technique in their coverage and constitute. It can be say that standard BVA test cases are a proper subset of Worst-Case test cases. BVA results in 4n+1 number of test cases where Worst-Case testing results into 5^n test cases. Due to this reason Worst-Case testing is used for situations that require a higher degree of testing where program malfunctioning can result into high cost. It has less regard for the time and effort required. In many situations it is too expensive to justify.

4.1.3.3 ROBUST WORST CASE TESTING

As the name suggest draws its attributes from Worst-Case and robust testing. It can be used for functions of greatest importance. Test cases are generated by taking Cartesian product of 7-tuple set which is (Min -, Min, Min+, Nom, Max-, Max and Max+) and results into larger set of test results we have seen so far and require the most time to produce it. The function ***f*** to calculate the number of test cases can be adapted to calculate the amount of Robust Worst-Case test cases. As each variable assumes 7 values we find the function ***f*** to be:

$$f(n) = 7^n$$

This function is also present in paper of title "A Testing and Analysis tool for certain 3-variable functions", Naryan [35]. Figure 4.6 shows the result of Robust Worst-Case Testing.

4.1.4 ANALYSIS & CONCLUSION

Boundary Value Analysis process is a systematic way of evaluating the completeness and quality of program under test. But there is redundancy in the set of test cases generated by BVA. We have found that the symmetry and mechanical nature of BVA help to make the test procedure easier to remember and apply in practice. It provides basis for learning other activities, in particular, equivalence partitioning. Due to its simplicity, it can be used misused and do not use it with full potential. It can be effective if used correctly. In case of variable dependencies and need for foresight into the system's functionality, we can find BVA restrictive. This is fact that BVA are computationally and theoretically inexpensive in the creation of test cases and can be desirable in its results to effort ratio. Finally BVA have importance and has a part to play in modern day testing practices.

4.2 EQUIVALENCE CLASS TESTING

Equivalence class testing bases upon supposition that a program's input and output domains can be partitioned into a limited no of classes means that all cases present in single class exercises the same functionality or exhibit the same behaviour. To test the input or output domain partitions test cases are designed and only one test case is selected from each partition. No of test cases are decreased and functional coverage increased, Myers et al [27]. The success of this approach is depends upon the classes for input and output identified by the tester. In reality, test cases selected from different domains causes distinct sequence of program source code to be executed, Reed [36]. According to Hamlet and Taylor [37], "Partition testing can be no better than the information that defines its sub-domains." If one input from class triggers program into error state then all other inputs in that class must also cause an error. If an equivalence class does not do it then it is not valid partition and thus the identification of additional partitions may be required.

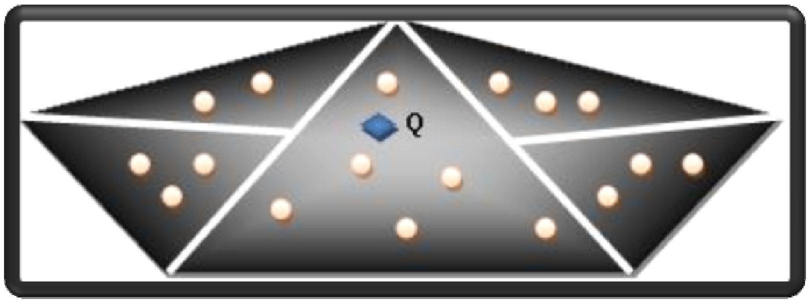


Figure 4.8 Sample set of Equivalence Partitions

The figure 4.8 shows the sample set of data comprise on input or output that we wish to use for producing test cases. The data is partitioned into subsets, each subset contain some part of original data. The key point is that the whole set is being included which shows completeness and the disjoint subsets ensures non redundancy. The equivalence relation devises a way of creating subsets. We have split up our data into disjoint groups, assume that all data within groups will behave similarly and we can carry out our test cases. We pick one point 'Q' to test it instead of testing the whole subset and can say that: *If we produce test cases using point 'Q' then the result or test cases will be same for all the points x_1 to x_n within the subset, Jorgensen [32].* This argument support our point that redundancy is greatly reduced because check one point instead of whole subset to be Checked. It is obvious advantage of Equivalence class testing.

4.2.1 BAKGROUND

The dominant style of programming in the 1960s and 1970s gave the idea of equivalence class testing. Myers, 1979; and Mosley, 1993 discuss the weak robust equivalence class testing in their standard testing texts. Input data validation was hot issue at that time. The programmer's watchword was "garbage in, garbage out". That issue was addressed with extensive input validation sections of a program. In the classic architecture of structured programming, authors and seminar leaders frequently commented that the afferent portion often represented 80% of the total source code. In this context, the main emphasize was on input data validation. The gradual move to modern languages (Strong typed or GUI) eliminated the much of the need for input data validation. Jorgensen [32]

4.2.2 EQUIVALENCE RELATION

Equivalence relation is a *binary relation* **R** on a set **A** if and only if **R** is reflexive, symmetric and transitive. Mathematically the relation between two sets x and y is denoted by Toida [38]: "x ~ y"

4.2.3 EQUIVALENCE CLASS

The set **A** which have equivalent relation **R** whose elements are related to an element, say a, of A is called Equivalent class of element 'a' and it is denoted by **[a]** Toida [38].

4.2.4 PARTITION

Suppose a set A have subsets $\{A_1, A_2 \ldots A_n\}$, these subsets are called partitions of A, if and only if Toida[38]: $\bigcup_{i=1}^{n} A_i = A$, *And* $A_i \cap A_j = \emptyset$, if $A_i \neq A_j$, $1 \leq i, j \leq n$.

4.2.5 TYPES OF EQUIVALENCE CLASSES

The different methods based on equivalence class testing underpin the functional testing and have two important properties: single fault assumption and multiple fault assumption. The two different types of equivalence class testing: weak and strong bases upon these properties. If we decide to test invalid and valid input/output, then we can produce two further types of equivalence class testing: normal and robust. Normal takes care of valid data values, whereas robust does not. The four different types of equivalence testing methods emerge: weak normal, weak robust, strong normal and strong robust.

4.2.5.1 WEAK NORMAL EQUIVALENCE CLASS

This type of equivalence class testing is based on the single fault assumption. The assumption is that rarely an error caused by two or more faults occurring simultaneously, Jorgensen [32]. That's why weak equivalence class testing only takes one variable from identified equivalence classes. See figure 4.9.

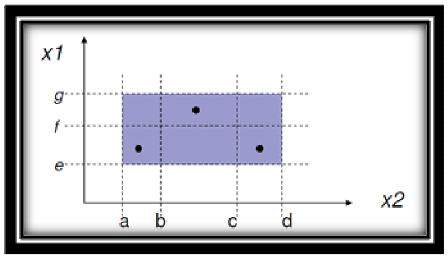


Figure 4.9: weak normal equivalence class test cases [32]

4.2.5.2 STRONG NORMAL EQUIVALENCE CLASS

Strong normal equivalence class assume multiple faults means that errors will result in a combination of faults, Jorgensen [32]. Therefore it tests every combination of elements formed as a result of the Cartesian product of equivalence relation. See figure 4.10.

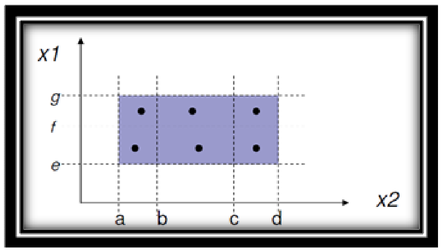


Figure 4.10: strong normal equivalence class test cases [32]

4.2.5.3 WEAK ROBUST EQUIVALENCE CLASS

The weak robust equivalence class testing tests one variable and invalid data values as well, Jorgensen [32]. The test case will have one invalid value and the remaining values will all be valid.

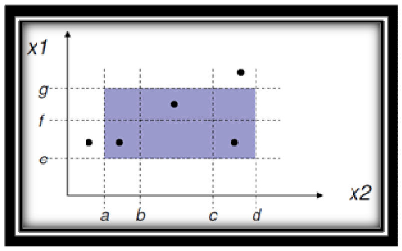


Figure 4.11 weak robust equivalence class test cases [32]

4.2.5.4 STRONG ROBUST EQUIVALENCE CLASS

This type of equivalence class testing produces test cases for all valid and invalid elements of the Cartesian product, Jorgensen [32].

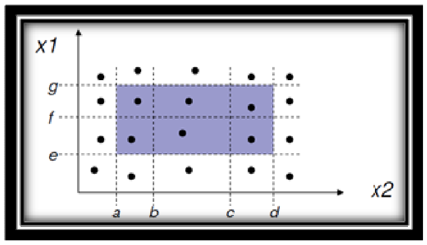


Figure 4.12 strong robust equivalence class test cases [32]

We get massive redundancy in strong robust equivalence class testing. As we compare it with Boundary Value Analysis where our ultimate objective is to produce useful test cases while reduce redundancy but in this case the first is usually achieved while the latter is not. It requires massive time of tester to design test cases and expected output against every invalid output which slows down the overall progress of the testing process. The vital problem is that it might be a waste of time. The strongly typed languages discarded the need to test for invalid inputs.

4.2.6 EQUIVALENCE TEST CASES

The equivalence relation plays a vital role in equivalence class testing. The optimal and useful test cases are created through a meaningful equivalence relation. To understand it, we will look at the NextDate function, Jorgensen [32].

4.2.6.1 NEXT DATE PROBLEM

The function under test is NextDate ( ) which takes date as input and produces as output the next date in the Georgian calendar. The three variables (day, month and year) are utilized which have valid and invalid intervals.

4.2.6.1.1 Weak Equivalence Relation

The following intervals are produced using weak equivalence relation:

Table 4.2 Interval produced using weak equivalence relation

No	Variables	Intervals	Valid
1	M_1	$\{ 1 \leq month \leq 12 \}$	☒
2	M_2	$\{ month<1 \}$	☐
3	M_3	$\{ month>12 \}$	☐
4	D_1	$\{ 1 \leq day \leq 31 \}$	☒
5	D_2	$\{ day<1 \}$	☐
6	D_3	$\{ day>31 \}$	☐
7	Y_1	$\{ 1812 \leq year \leq 2012 \}$	☒
8	Y_2	$\{ year<1812 \}$	☐
9	Y_3	$\{ year>2012 \}$	☐

At glimpse, it looks that all information needed to create test cases is available. We produce test cases for four types of equivalence class testing techniques.

Table 4.3 Strong and weak equivalence class

Strong And Weak Equivalence Class				
No	**Day**	**Month**	**Year**	**Output**
1	17	6	1912	18-6-1912

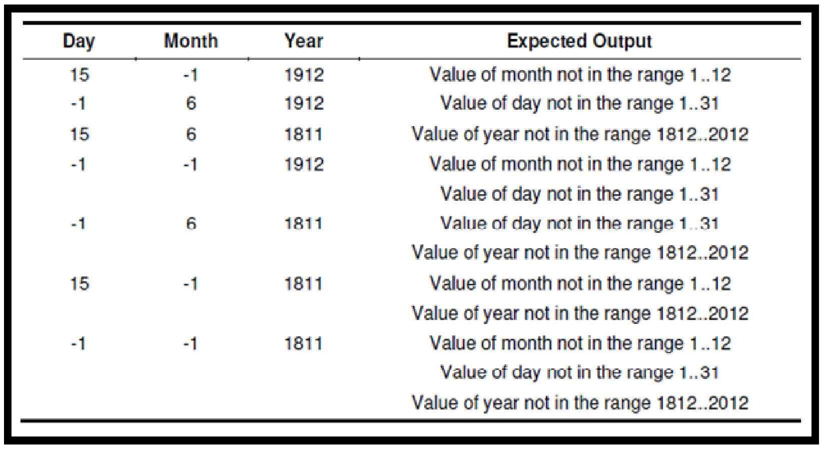
Only one weak normal equivalence class test case occurs because the number of variables is equal to the number of valid classes. See table 4.3.

Table 4.4Weak normal equivalence class test cases

Weak Robust				
No	**Day**	**Month**	**Year**	**Output**
1	17	6	1912	18-6-1912
2	-1	6	1912	Day is not in Range.
3	32	6	1912	Day is not in Range.
4	15	-1	1912	Month is not in Range.
5	15	13	1912	Month is not in Range.
6	15	6	1811	Year is not in Range.
7	15	6	2013	Year is not in Range.

As we can see from table 4.4 that weak robust equivalence class testing will just test the ranges of the input domain once on each class. We are testing weak and not normal, there will only be at most one fault per test case unlike Strong Robust Equivalence class testing.

The test cases under multiple fault assumption given in figure 6, shows one corner of the cube in 3D space. The three other corners would include a different combination of variables since the complete table would be too large to show.



Day	Month	Year	Expected Output
15	-1	1912	Value of month not in the range 1..12
-1	6	1912	Value of day not in the range 1..31
15	6	1811	Value of year not in the range 1812..2012
-1	-1	1912	Value of month not in the range 1..12
			Value of day not in the range 1..31
-1	6	1811	Value of day not in the range 1..31
			Value of year not in the range 1812..2012
15	-1	1811	Value of month not in the range 1..12
			Value of year not in the range 1812..2012
-1	-1	1811	Value of month not in the range 1..12
			Value of day not in the range 1..31
			Value of year not in the range 1812..2012

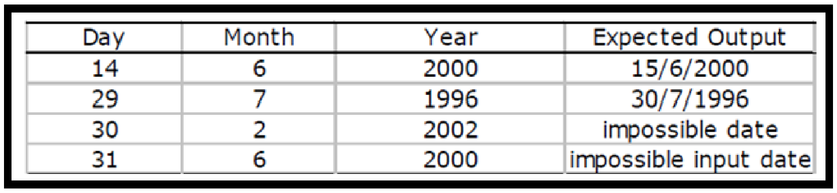
Figure 4.13 Strong Robust Test Cases [32]

4.2.6.1.2 Strong Equivalence Relation

The previously generated test cases are inadequate because our equivalence relation is not strong. These cases do not address the NextDate function problem adequately and satisfactory. We cannot get information related to Georgian style calendar because of poor design. We refine our result by thinking about questions: What happens when a month only has 30 days? What happens if the month is February? What happens if we have a leap year and the month is February? By taking care of these questions we produce following equivalence classes. See table 4.5.

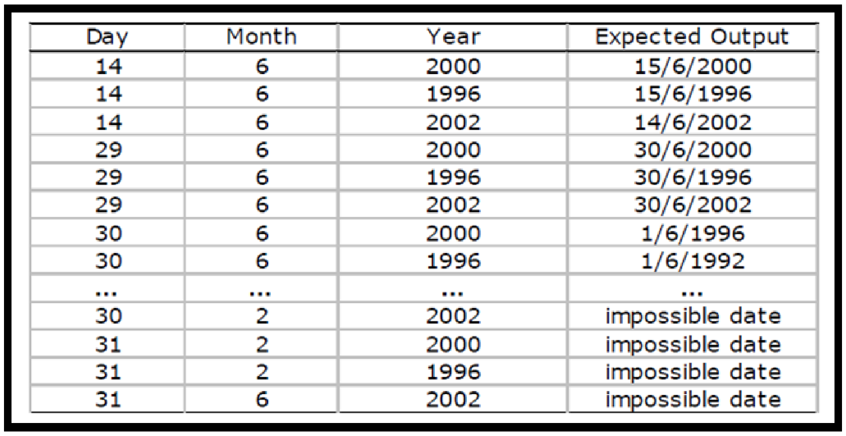
Table 4.5 Equivalence Classes

No	Variables	Intervals	Comments
1	M_1	Month has 30 days.	April, June, September and November
2	M_2	Month has 31 days.	January, March, May, July, August, October and December
3	M_3	Month is February.	
4	D_1	$\{ 1 \leq \text{day} \leq 28 \}$	February
5	D_2	Day = 29	February
6	D_3	Day = 30	April, June, September and November
	D4	Day = 31	January, March, May, July, August, October and December
7	Y_1	Year = 2000	
8	Y_2	Year is a leap year.	
9	Y_3	Year is a common year.	



Day	Month	Year	Expected Output
14	6	2000	15/6/2000
29	7	1996	30/7/1996
30	2	2002	impossible date
31	6	2000	impossible input date

Figure 4.14 Weak Normal Test Cases [32]



Day	Month	Year	Expected Output
14	6	2000	15/6/2000
14	6	1996	15/6/1996
14	6	2002	14/6/2002
29	6	2000	30/6/2000
29	6	1996	30/6/1996
29	6	2002	30/6/2002
30	6	2000	1/6/1996
30	6	1996	1/6/1992
...	...	...	...
30	2	2002	impossible date
31	2	2000	impossible date
31	2	1996	impossible date
31	6	2002	impossible date

Figure 4.15 Strong Normal Test Cases [32]

As we can see figure 7 and 8, there are impossible dates because equivalence class testing uses automatic test generation means that values are selected automatically, from middle of the class.

Due to this automation we can see that this process has no knowledge of a Georgian calendar. Dates in which tester is interested would be on leap years in February where the days 28 and 29. Other dates would be at the end of months to test whether the months have the correct number of days in them. Automatic test generation don't care these conditions and will always be problematic. As we move from weak class to further classes, redundancy is introduced. Strong Robust equivalence class test case design technique generates 150 test cases, Jorgensen [32]. This equivalence relation is not perfect; it shows how crucial it can be to produce a high quality Equivalence Relation.

4.2.7 FUNCTIONAL TESTING

Sophistication means that how useful test cases produced by the testing type in testing a particular function, high sophistication shows a testing type would result in test cases which prove very useful to the tester. Effort concerns about how much analysis and design work is required to generate test cases. If we relate sophistication to Equivalence class testing then we can see that it produces relatively low number of test cases. Equivalence class testing improves BVA by reducing the number of test cases. As we compare Equivalence class testing with decision table, it does not seem sophisticated but it has some advantages. BVA required least effort to identify and produce test cases but the test cases will be inadequate and large in number. Equivalence class testing requires great effort to identify test cases. The good equivalence relation depends on time and effort. While Equivalence partitioning is not sophisticated as Decision-table based testing, it requires less effort to implement. See figure 4.16.

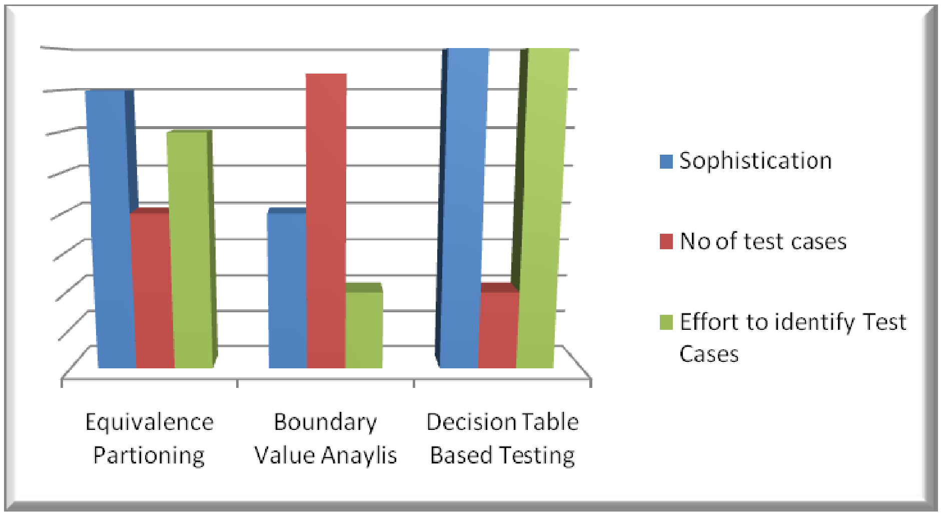


Figure 4.16 Comparison of Testing Strategies

4.2.8 ANALYSIS & CONCLUSION

The strong base of functional testing is makes up by Equivalence class, BVA and decision table based testing. Equivalence class testing partitions the input/output of a data domain and each part of partition is treated same. Based upon this assumption the equivalence class testing need only one element from each subset to prove that all elements within the set are correct which makes it appealing. For effective Equivalence class testing and its potential implementation, time must be spent on designing a strong equivalence relation.

4.3 DECISION TABLE BASED TESTING

A decision table is composed of inputs (causes) their associated outputs (effects) which can be used to design test cases. Decision table based testing is black box testing technique in which test cases are designed to execute the combination of causes present in a decision table to see its effects, Veenendaal [39]. In 1960, decision table based testing has been introduced. It is used to build a model and complex logical relationships between input data. Two functional testing methods are closely related to it: The Cause-Effect Graphing, Mosley et al [40] and The Decision Tableau Method, Vauthier [41]. These methods use similar concepts used in decision table. Cause-Effect Graphing is same as decision table where input produces output shows logical relationship which is shown in form of a graph. Graph is like Finite State Machine (FSM). Symbols show the relationship between input conditions; these symbols are similar to propositional logic symbols.

4.3.1 DECISION TABLE

4.3.1.1 ELEMENTS

Decision table provides test cases without using the internal structure of program. For creating test cases it uses a table which is split up into four sections: Condition Stubs, Action Stubs, and Rules.

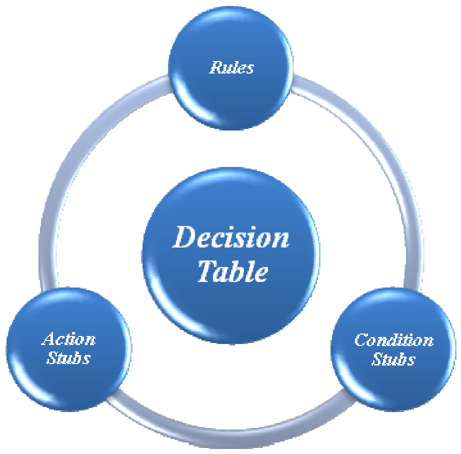
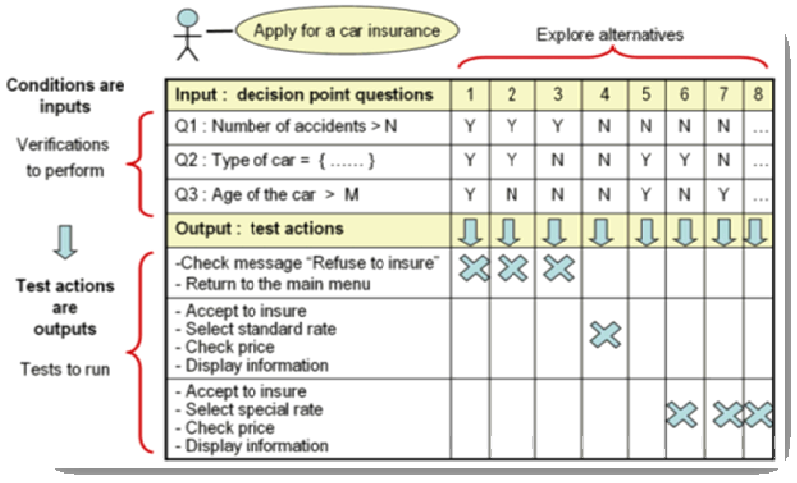


Figure 4.17 Decision Table Components

Figure shows composition of decision table: Condition Stubs describe factor or conditions that affect on decision or policy. They are present on upper section of decision table. Action Stubs describe possible decisions or policy actions. They are listed in lower section of decision table.

Rules are used to decide which actions are to be taken under a specific combination of conditions. First list different combinations of condition attribute values and then put X's in appropriate columns of the action section of table. Figure 2 is typical example of decision table which shows conditions, actions and rules to apply for car insurance, Roggenbach et al [42].


Apply for a car insurance
Explore alternatives
Conditions are inputs
Verifications to perform
Test actions are outputs
Tests to run

Input : decision point questions	1	2	3	4	5	6	7	8
Q1 : Number of accidents > N	Y	Y	Y	N	N	N	N	...
Q2 : Type of car = { ...... }	Y	Y	N	N	Y	Y	N	...
Q3 : Age of the car > M	Y	N	N	N	Y	N	Y	...
Output : test actions	⇩	⇩	⇩	⇩	⇩	⇩	⇩	⇩
-Check message "Refuse to insure" - Return to the main menu	✗	✗	✗					
- Accept to insure - Select standard rate - Check price - Display information				✗				
- Accept to insure - Select special rate - Check price - Display information						✗	✗	✗

Figure 4.18 Decision Table Examples

4.3.1.2 TYPES

There are two types of decision tables: Limited and Extended Entry tables. Limited entry decision table contains only Y, N or – in condition entry area and X or – in the action entry area.

Table 4.6 Limited Entry Decision Table

Limited-entry Decision Table	**Decision Rules**					
	1	2	3	4	5	6
Condition Stub	**Condition Entries**					
Condition 1	Y	Y	Y	N	N	N
Condition 2	Y	Y	N	-	Y	-
Action Stub	**Action Entries**					
Action 1	X	-	X	-	X	X
Action 2	X	X	-	X	-	X
Action 3	-	-	X	X	X	X

Where:

Y	=	Yes	= Condition is true
N	=	No	= Condition is false

-	=	Irrelevant	= Condition does not matter
-	Or	Blank	= Action is not performed
X			= Action is performed

In extended entry decision table, entry portion contains a portion of the condition as shown in following figure.

Table 4.7 Extended Entry Decision Table

Extended-entry Decision Table	**Decision Rules**					
	1	2	3	4	5	6
Condition Stub	**Condition Entries**					
Variable 1	<7	>7	=7	<=7	>7	>=7
Variable 2	=2	!=3	-	-	-	=1
Action Stub	**Action Entries**					
Execute Procedure	10	05	50	34	33	21

Mixed entry table contains both extended and limited entries as shown in table 4.8 below:

Table 4.8 Mixed Entry Decision Table

Extended-entry Decision Table	**Decision Rules**					
	1	2	3	4	5	6
Condition Stub	**Condition Entries**					
Variable 1	<7	>7	=7	<=7	>7	>=7
Condition 1	Y	N	-	-	N	Y
Action Stub	**Action Entries**					
Execute Procedure	10	05	50	34	33	21
Action 1	-	X	-	X	-	X

4.3.1.3 RULE COUNTS

To check decision table completeness, rule counts are used along with don't care entries. The amount of test cases can count in a decision table using rule counts and compare it with a calculated value. Following formula is used to calculate number of rules:

$$\text{Total number of Rules} = 2^{\text{Number of conditions}}$$

E.g. If no of conditions are 5 then total no of rules are 32

"Don't care" entries play an important part in counting rule count of a rule. Each "don't care" entry in a rule doubles the rule count of a rule. Initially each rule has a rule count of 1, if don't

care entry exists then this rule count doubles for every "don't care" entry. Both ways of calculating rule count brings same value which shows the completeness of decision table.
When rule counts calculated by counting rules or by equation are not equal then decision table is not complete, and therefore needs revision.

4.3.1.4 REDUNDANCY & INCONSISTENCY

The "don't care" entries introduce redundancy and in consistency within a decision table. The redundant rule counts exist in following table 4.9:

Table 4.9 Example of Redundant Rule

Conditions	1-5	6	7	8	9	10
Condition Stub	**Condition Entries**					
Condition 1	T	T	T	F	F	T
Condition 2	T	F	F	T	F	T
Condition 3	-	T	T	F	F	F
Condition 4	-	T	F	T	F	F
Action Stub	**Action Entries**					
Action 1	X	X	X	-	X	X
Action 2	-	X	-	X	-	-

In above table we can see conflict between rule 1-5 and rule 10, rule 1-5 use "don't care" entries as an alternative to false, but rule 10 replaces those "don't care" entries with "false" entries. So when condition 1 is met, rules 1-5 or 10 may be applied, in this instance these rules have identical actions so there is only a simply correction to be made to complete above table.

If we actions of the redundant rule differ from that of rule 1-5 then we have a problem, which is shown in table below. If condition 1, 2 is true and 3, 4 are false then rules for 1-5 and 10 could be applied. It will lead toward a problem because actions of these rules are inconsistent so therefore the result is non-deterministic and would cause decision table to fall short.

Table 4.10 an example of Inconsistent Rules

Conditions	1-5	6	7	8	9	10
Condition Stub	**Condition Entries**					
Condition 1	**T**	**T**	**T**	**F**	**F**	**T**
Condition 2	**T**	**F**	**F**	**T**	**F**	**T**
Condition 3	**-**	**T**	**T**	**F**	**F**	**F**

Condition 4	-	T	F	T	F	F
Action Stub	**Action Entries**					
Action 1	**X**	**X**	**X**	**-**	**X**	**-**
Action 2	**-**	**X**	**-**	**X**	**-**	**X**

4.3.2 DECSION TABLE CREATION PROCESS

In creating a decision table care must be taken when choosing stub conditions and type of decision table. Extended entry decision table are difficult to create instead of limited Entry Decision tables. Here is some steps how to create a simple decision table using following example.

"A marketing company wishes to construct a decision table to decide how to treat clients according to three characteristics: Gender, City Lahore, and age group: A (under 30), B (between 30 and 60), and C (over 60). The company has four products (I, J, K and L) to test market. Product I will appeal to female city Lahore. Product J will appeal to young females. Product K will appeal to Male middle aged shoppers who do not live in cities. Product L will appeal to all but older females."

Step1. Identify Conditions & values

There are three data attributes tested by conditions: gender (Male, Female), city Lahore and age group (A, B, and C).

Step2. Compute Maximum Number of Rules

Maximum no of rules are: 2*2*3 = 12

Step3. Identify Possible Actions

Possible actions are: Market Product I, Market Product J, Market Product K and Market Product L.

Step4. Enter All Possible Rules &

Step5. Define Actions

Table 4.11 Sample Decision Table

Limited-entry Decision Table	**Decision Rules**											
	1	2	3	4	5	6	7	8	9	10	11	12
PROCESS	**Condition Entries**											
Sex	F	M	F	M	F	M	F	M	F	M	F	M
City	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Age	A	A	A	A	B	B	B	B	C	C	C	C
MARKET	**Action Entries**											
I	X				X				X			

J	X		X									
K								X				
L	X	X	X	X	X	X	X	X		X		X

Step5. Verify Policy

Our assumption is "client is agree with our decision table"

Step5. Table Simplification

Rule no 2, 4, 6, 7, 10, 12 can be merge because they have same action pattern. Rules 2, 6 and 10 have two of three condition values (city Lahore and gender) identical and all three of the values of the non-identical value (age) are covered, they can be condensed into a single column 2. The rules 4 and 12 have identical action pattern, but they can't be combined because the indifferent attribute "Age" does not have all possible values which cannot be covered in two columns because third age group B is missing. The simplified table is given:

Table 4.12 Simplified Decision Table

Limited-entry Decision Table	**Decision Rules**									
	1	2	3	4	5	6	7	8	9	10
PROCESS	**Condition Entries**									
Sex	F	M	F	M	F	F	M	F	F	M
City	Y	Y	N	N	Y	N	N	Y	N	N
Age	A	-	A	A	B	B	B	C	C	C
MARKET	**Action Entries**									
I	X				X			X		
J	X		X							
K							X			
L	X	X	X	X	X	X	X			X

4.3.3 ANALYSIS & CONCLUSION

When we think about functionality, following methods show up: *Boundary Value Analysis, Equivalence Class Testing and Decision Table Based Testing*. These methods are complement to each other. We cannot achieve functional testing at satisfactory level by using only one of these functional testing Strategies or even two. Decision table based testing is evolutionary form of Equivalence class testing because ECT groups together inputs of the same manner which behave same. Decision table based testing follows ECT by grouping together the input and output behaviours into an "equivalence" rule and test logical dependencies of these rules. These rules are test cases after simplification, Roggenbach et al [42].

All methods are different in terms of application and effort. Boundary Value Analysis is domain based testing and does not discuss any data or logical dependencies. It is less sophisticated and no of test cases generated by it is high. It requires low effort to develop test cases. Equivalence Class Testing is concerned with data dependencies and generates test case classes of similar input and output. It is sophisticated than BVA and generates fewer test cases. Decision Table Based testing is similar to ECT but it tests logical dependencies. It requires much effort to identify test cases which leads to sophistication. The amount of rules required to complete the set of test cases is reduced because it checks logical dependencies, Roggenbach et al [42].

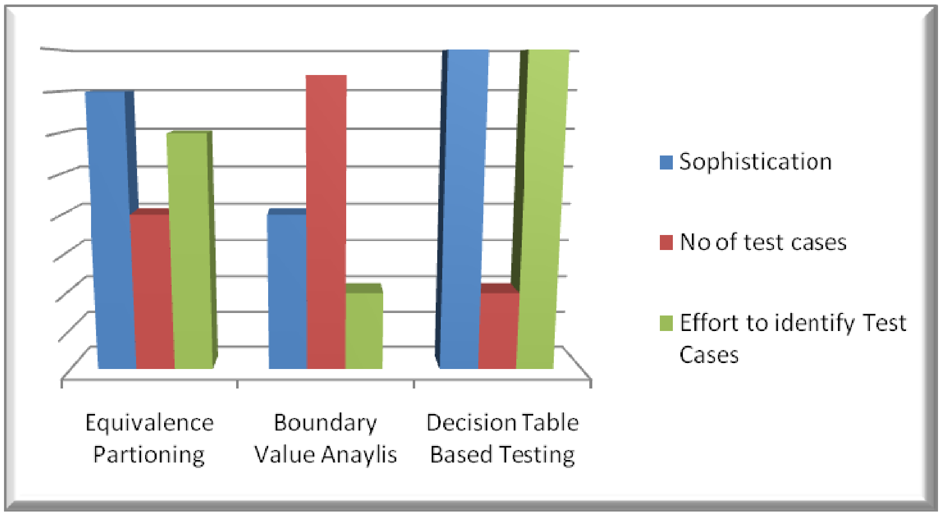


Figure 4.19 Comparison of Testing Strategies

Average no of test cases generated by Boundary Value Analysis are five times greater than Decision Table Based testing. One and half times as many test cases generated by Equivalence Class Testing. Although, it shows test case redundancy or unreachable test case which reduces efficiency of these testing strategies but supports" How efficient Decision Table Based Testing is". As above stated test strategies are complement to each other but we cannot deny their importance and these are not totally redundant in all scenarios.

4.4 USE CASE BASED TESTING

Common thinking is that software is not well tested before it is delivered. The people mentality was based on two propositions; software is very difficult to test and second testing is done without clear methodology but the Rational Unified Process model assumes that it is better to log user requirements in natural language, in easy and understandable format and it is better to start testing as early in the software development process as possible. Late start of testing activities until all development is done is a highly risk way to proceed. In case of severe level bugs, the schedule can slip. In this section of Chapter we will discuss use cases to generate test cases. How it can be helpful to launch the testing process early in the development lifecycle and also help with testing methodology.

4.4.1 BACKGROUND

The idea of use case, usage scenario was given by Ivar Jacobson during definition of architecture for Ericsson's AXE system back in 1967 [46]. He coined the Swedish term anvendningsfall which means "situation of usage" or "usage cases", Jacobson [46]. Cockburn [47] presented an effective and widely used technique for specifying Use Cases which was based on natural language specification for scenarios and their extensions and it makes requirement documents easy to understand and communicate to non-technical people like stakeholders, project manager etc.

4.4.2 USE CASE AND REQUIREMENT TYPES

Requirement can be definition, capability or condition; it shows how much system must ensure user expectations, Fowler [47]. The objectives or restrictions placed by user to solve a problem can be requirement. The capability of system to satisfy a contract, standard, specification or other formally imposed document. The requirement stack is shown in figure 4.20.

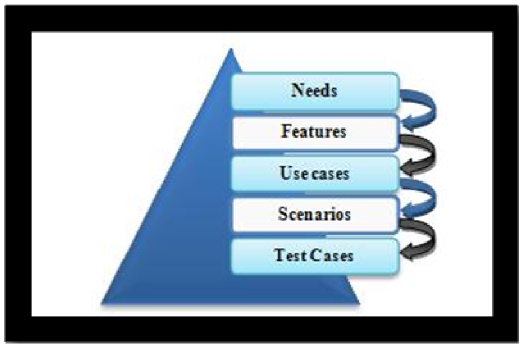


Figure 4.20 Requirements Stack

Stakeholder needs are at top of the stack. The number of high level needs vary from six to fifteen. Below it, features, use cases and scenarios exist. Top to bottom in stack shows requirements from coarse grained to fine grained. E.g. User needs can be: data should save in encrypted form. The feature can refine this requirement to be: Any encryption algorithm should use. Further down: DES algorithm should be used.

4.4.3 USE CASE AND REQUIREMENT TRACEABILITY

The relationship between requirements is found by a technique called Requirement Traceability, Fowler [47]. This technique helps to find the origin of the requirement. Figure 4.21 shows how requirements are expanded or mapped from abstract level to detail level. Initially, vision is small and exists in form of stakeholder needs. Then visions is made broad by mapping each need to couple of features, this process goes through uses cases, scenarios and finally needs are mapped to test cases with broad vision of user requirements.

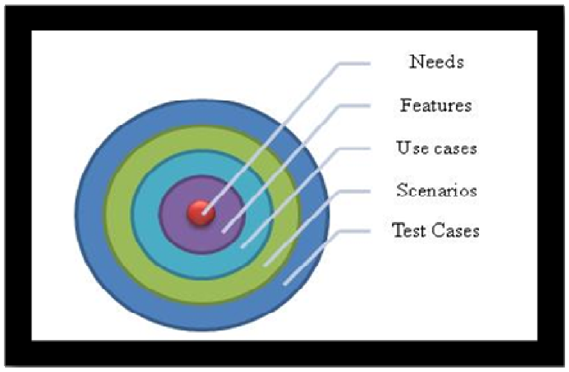


Figure 4.21 Requirement Traceability and Vision expansion from Needs to Test Cases

According to Zielczynski [47], traceability plays an important role: to verify that an implementation fulfils all requirements e.g. everything that the customer requested was implemented, to verify that the application does only what was requested e.g. do not implement things which customer never asked.

4.4.4 USE CASES

According to Wood and Reis [48] a top level category of system functionality is use case. A use case can be visualized and described. The visualization by using different diagrams identifies all the actors (primary, secondary etc) involved in the function and the description provides an indication of how the use cases is initiated and evolved. The context diagram of system interfaces is collection of use cases diagrams. A use case is comprised on a complete list of events initiated by an actor and specifies the interaction that takes place between an actor and the system.

A use case shows the system in opaque form means only the inputs, outputs and functionality matter, Wu et al and Jia et al [49, 50]. According to Illes et al[51], purpose of use case may include; Promoting communication, Understanding requirements, Focusing on' what' rather than 'how' and Providing prototype test cases. Scenarios can be elaborated by use cases which ensure the functionality required, can be supported and can be used to discover the objects that will construct a system that satisfies all functional requirements. At conceptual level, functionality of a system is a set of processes that run horizontally and the objects as subsystem components that stand vertically. It is not necessary that all functionality should be use by every object but each object may be used by many functional requirements. Use cases and scenarios make transition from functional to an object point-of-view.

4.4.5 SCENARIOS

A scenario is part of use case, an instance of a use case, or a complete path through the use case , Beizer [52]. To perform functionality, various paths are exercises in the use case. By using scenarios, the use cases can generate multiple flows of events. Each use case is comprised on one primary scenario and multiple secondary scenarios; both scenarios can have its flow of events as shown in figure 3. The primary scenario should cover the flow of events what normally happens when the use case is performed. The exceptional character relative to normal behaviour is shown by secondary scenario.

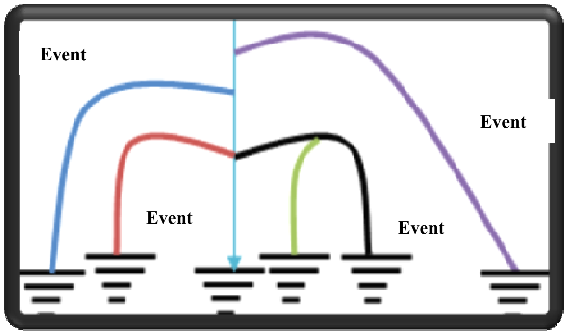


Figure 4.22In use case multiple flows of events.

Table 4.13 shows an example use case "Register for Course" which constitutes on scenarios. The primary scenario events yield a successful registration for a course. Each secondary scenario has its own flow of events which require a significant amount of detail to fully specify a use case. The precondition shows the required state of the system prior to the start of the use case. The post condition is the state of the system after completion of functionality.

Table 4.13 RegisterForCourse use case

Use Case	
Name:	RegisterForCourse
ID	UC4
Actor:	Participants
Preconditions:	
• Participants have an account on the system.	

• Participants are not already registered for the courses.
Primary Scenario:
Login: • The use case begins when the participants access web interface.   • System ask for user id and password   ***Select a Course:***   • The system shows a list of available courses that participants have not yet registered for. Participant selects a course.   ***Enter Additional Information:***   • The System ask the participant for certain courses-specific information   ***Enter Payment information:***   • The system asks the participant to select payment method and make payment.   ***Submit Information:***   • The participant submits completed information. The system verifies the information and checks if the participant has already registered for that courses.   ***Display Conformation:***   • The system displays a confirmation message of the participant registration
Secondary Scenarios:
• User Not Found   • Registration Closed.   • Invalid Payment information   • Invalid Information   • System Unavailable   • Quit.

Post conditions:
Participant receives a confirmation receipt for the course registration.

4.4.6 TEST CASE GENERATION

A set of test inputs, execution conditions, and expected results developed for particular objective shows a test cases e.g. to exercise path execution in a program or verify compatibility with requirements. Test case helps to identify and communicate conditions that will be implemented in the test. For successful and acceptable implementation of the product requirements test cases are necessary. The five-step process for generating test cases is described below, Tsun [53]. The process has been extended by adding two new steps regarding prioritization mechanism which are called use case prioritization and scenario prioritization, along with additional guidelines for generating scenarios has also been suggested.

Step 1. Requirement traceability matrix and Use case prioritization

Step 2. Generate scenarios for each use case.

Step 3. Scenario Prioritization.

Step 4. For important scenarios, generate at least three test cases, one test case for other scenarios and the conditions that will make it execute.

Step 5. Identify the data values for each test case.

4.4.6.1 USE CASE PRIORITIZATION

Financial applications are developed under tight time deadlines and focus is to capture the market. The fast and furious development involves from one to three months with small team of developers. Product based companies focus on software releases. In such situations, requirements are prioritized to manage work in releases and deliver each release on time. The frequency of usage of functionality shows its importance. The system requirements mentioned in use case used to drive requirement traceability matrix and guiding the selection of test cases for the system. The table 4.14 shows the format of requirement traceability matrix where each requirement is rated according to scale (High (3), Medium (2), Low (1)). The use case which has highest total value is most important.

Table 4.14 Requirement Traceability Matrix

Sr. #	Rank	Initial Requirements	UC-01	UC-02	UC-03	UC-04
1	Highest (3)	R1	X	X	X	**X**
2	Highest (3)	R2		X		**X**
3	Medium (2)	R3	X			**X**
4	Medium (2)	R4				
5	Low (1)	R5		X		**X**
TOTAL			**5**	**7**	**3**	**9**

4.4.6.2 SCENARIO GENERATION

The process of scenario generation starts from highest priority use case. Sufficient set of scenarios are generated. The scenario which has detailed description of input variables, preconditions and post conditions can be very helpful in the test case generation, Tsun [53]. Maximum possible scenarios for 'RegisterForCourse' use case are given table 4.15. For important use cases, there must be a primary scenario and at least two secondary scenarios, which could be used to represent abnormal behaviour of the application and to certify the validity of functionality. For comprehensive modelling and more thorough testing, more secondary scenarios can be used.

Table 4.15 Scenarios for 'RegisterForCourse' use case

Sr. #	Scenario ID	Scenario Name	Priority
1	S1	SuccessfulCourseRegisteration	**2**
2	S2	UserNotFound	**2**
3	S3	InvalidPayment Information	**3**
4	S4	InvalidCourseInformation	**2**
5	S6	SystemUnavailable	**1**
6	S7	DuplicateRegistration	**3**

4.4.6.3 SCENARIO PRIORITIZATION

Scenarios are prioritized according to importance of functionality. For high priority scenarios, at least three test cases are suggested, for detail and comprehensive testing of functionality. Some scenarios are very important any mistake in it can cost company e.g. the money transfer, commission calculation etc. In table 4.15, priority is assigned to each scenario identified from use case given in table 4.13.

4.4.6.4 TEST CASE GENERATION

After scenario generation and use case, scenario prioritization, test case identification process starts. Table 4.16 shows the test cases for each scenario in form of test case matrix which represents a framework for testing without involving any specific data values. The values V, I and N/A indicate valid, invalid and not applicable respectively. Matrix formulation is intermediate step and provides a good way to document the conditions that are being tested. If test cases have identical entries in all columns, then it means that some more conditions needs to be added to it to specifically address these scenarios, Tsun [53].

Table 4.16 Test Case Matrix for 'RegisterForCourse' use case

Test Case #	**Scenario**	**Email ID**	**Password**	**Card No**	**Amount**	**Course Information**	**Already Registered**	**Registration Open**	**Expected Output**
1	SuccessfulCourse Registeration	V	V	V	V	V	V	V	**Display Conformation Message**
2	UserNot Found	I	N/A	N/A	N/A	N/A	N/A	N/A	**Error: User not found**
3	InvalidPayment Information	V	V	I	V	V	N/A	N/A	**Error: Invalid Payment Info**
4	InvalidPayment Information	V	V	V	I	V	N/A	N/A	**Error: Invalid Payment Info**
5	InvalidPayment Information	V	V	I	I	V	N/A	N/A	**Error: Invalid Payment Info**
6	InvalidCourseInformation	V	V	V	V	I	N/A	N/A	**Error: Invalid Course Info**
7	SystemUnavailable	V	V	N/A	N/A	N/A	N/A	N/A	**Error Message**

8	DuplicateRegistration	**V**	**V**	**V**	**V**	**V**	**I**	**N/A**	**Error Message**

4.4.6.5 *DATA VALUES IDENTIFICATION*

The test cases should be reviewed, validated to ensure accuracy, completed and identify redundant or missing test cases. After this step, next step is to plug in the data values for each Valid and Invalid entries. Before application deployment in real environment, it should be tested with real data to see other issues like performance, availability etc. Test data generation is itself an extensive subject of discussion, Tsun [53].

Table 4.17 Test Case Matrix with data values for 'RegisterForCourse' use case

Test Case #	Scenario	Email ID	Password	Card No	Amount	Course Information	Already Registered	Registration Open	Expected Output
1	SuccessfulCourseRegisteration	Umar@yahoo.com	Abc	123456	2000 Kr	Ok	No	yes	**Display Conformation Message**
2	UserNotFound	unknown@hotmail.com	N/A	N/A	N/A	N/A	N/A	N/A	**Error: User not found**
3	InvalidPaymentInformation	Umar@yahoo.com	Abc	123459	2000 Kr	V	N/A	N/A	**Error: Invalid Payment Info**
4	InvalidPaymentInformation	Umar@yahoo.com	Abc	123456	-2000 Kr	V	N/A	N/A	**Error: Invalid Payment Info**
5	InvalidPaymentInformation	Umar@yahoo.com	Abc	123459	-2000 Kr	V	N/A	N/A	**Error: Invalid Payment Info**

6	InvalidCourseInformation	Umar@yahoo.com	Abc	123456	2000 Kr	I	N/A	N/A	**Error: Invalid Course Info**
7	SystemUnavailable	Umar@yahoo.com	Abc	N/A	N/A	N/A	N/A	N/A	**Error Message**
8	DuplicateRegistration	**Umar@yahoo.com**	Abc	123456	2000 Kr	Ok	Yes	N/A	**Error Message**

4.4.6.6 ANALYSIS & CONCLUSION

Use case based testing claims a lot of advantages. Most object oriented analysis and design methodologies place it in inherent parts. It provides base for both software development as well as for testing, provides a uniform notation and a high reusability of requirements engineering artifacts. By developing test cases in parallel to use cases enables an early validation of the requirements.

The defect taxonomy proposed in, Mealy [54] is used, defects can be categorized into: *Completeness defects* related to an incomplete implementation of the specified functionality, *Input/output defects* related to wrong input respectively to wrong output data, *Calculation defects* resulting from wrong formula, *Data handling defects* related to the lifecycle and the order of operations performed on data, *Control flow and sequencing defects* related to the control flow or the order and extent to which processing is done, *Concurrency defects* related to the concurrent execution of parts, *GUI defect* related to user interface and *Non-Functional requirement defects* related to the quality like availability, functionality, reliability etc.

To uncover aforementioned defects, different strategies are used but do not provide satisfactory results. To uncover missing information, analysis per use case is required, convert it to state chart diagram for complete path coverage, input/output defects cannot covered if enough detail in use case is not available, to uncover it and calculation defects, Moore [55] suggests the concept of extended use case, and tabular representation to relate a combination of inputs and system states to outputs, to uncover control flow defects use case needs to convert into many models like state chart, system sequence diagram, sequence diagram etc, Mealy [54], to remove concurrency defects, information related to dependency is required, use cases answer to 'what' should be realized by the system but not to 'how well' a requirement should be realized, Mealy [54], to remove GUI defects intermediate models are required like story boards etc.

Authors conclude on base of above discussion that requirement documentation in form of use cases is not enough to detect defects, challenging defects are uncovered until or unless use case is not converted to intermediate UML models like sequence diagram, activity diagram etc. The transformation of use case into intermediate models requires time and effort. To boost up work, tools can be used which provide rapid transformation.

4.5 STATE DIAGRAM BASED TESTING

State based testing is used where some system aspects are visualized in form of a model which is called a 'finite state machine'. The system has finite set of states and transitions from one state to another which are determined by the rules of the 'machine'. Test cases are based on this model. It is used in event driven systems, real time events etc. A state diagram or state graph shows a model from system point of view. For example, when you want to change data in database it is required that first fetch the data, change data and update it. This system has view and update state.

4.5.1 MODEL BASED TESTING

MBT or Model based testing is a general term which signifies an approach that bases common testing tasks such as test case generation and test result evaluation, Jorgensen [32] on a model of application under test. A State Diagram Based Model is composed of states and transitions

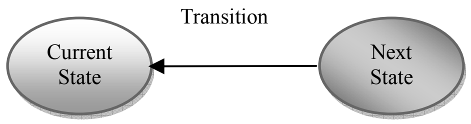


Figure 4.23 State Diagram

4.5.1.1 WHY MODEL BASED TESTING

To specify, develop and understand systems in many disciplines, models are used. It is used to promote understanding and provide a reusable framework for product development from DNA and gene research to the development of the latest fighter aircraft. Models are part of object oriented analysis and design approach by all of the major Object Oriented methodologies. Modelling is an economical way of capturing knowledge about a system and then reusing this knowledge as the system grows. Model is like crystal for a testing team: what percentage of a test engineer's task is spent trying to understand what the System under Test (SUT) should be doing? Model provides a mechanism for a structured analysis of the system to team. Test planes are developed in the context of SUT and test coverage is understood beside of it all of this work is not lost. If the requirement is to add a new feature in a product then it can be incrementally added to the model.

4.5.1.2 SOFTWARE TESTING AND MODELS

The most famous and useful models for software testing are: finite state machines, state charts and Markov chains.

- *FINITE STATE MACHINES*

  Let consider a testing scenario in which tester selects an input and then appraises the result. Tester selects another input on base of prior result and appraises the next set of possible inputs. Tester makes choice for inputs from a specific set of inputs. The selection is based on prior input result. This deterministic characteristic of software makes state based models a logical fit for software testing: software is always in a specific state and the current state of the application governs what set of inputs testers can select from. On base of above stated behaviour software is considered a finite state machine which is also known as finite automata (FA). Before the inception of software engineering, FA has been around it is a stable and mature theory of computing. The usage of FA in software and hardware component testing has been long established and is considered a standard today. Chow [54] article is considered one of the earliest articles addressing the use of FA models to design and test software components. Mealy and Moore proposed variations of finite state machine models, Morre and Rosaria et al [55, 56]. The Robinson and Rosaria [57], Liu & Richardson [58] did work on finite state machines in testing with varied tones, purposes and audiences.

  DEFINITION

  A finite automata represents a software system in form of quintuple (I, S, T, F, and L) Where:

  - 'I' is the set of all system inputs.
  - 'S' is the set of all system states.
  - 'T' is a function which determines whether a transition occurs when an input is applied to the system in a particular state.
  - 'F' is a set of final state where all inputs end.
  - 'L' is software launching state.

A machine is always in one state at a particular time. The transition from one state to another depends on an input in 'I'.

The Finite State model may be restricted to having only one starting state and one final state. Some models have multiple starting and final states. In addition to the current state and the application of an input, on some condition external to the machine, a transition is dependent.

- STATE CHARTS

State charts are used to model complex or real time systems. It is an extension of finite state machines. The state machine is specified in form of hierarchy where a single state can be expanded into another lower level state machine. State charts are equivalent to most power full form of automata: the Turing machine. Hong [59] and Kemey et al [60] had done work on testing with state machines.

- MARKOV CHAINS

Kemeny and Snell introduced stochastic models or Markov models, Graham et al [61]. The classes of Markov chains are: the discrete parameter, time homogenous, finite state, irreducible Markov chain has been used to model the usage of software. They have resemblance to finite state machines and can be thought of as probabilistic automat. It also gathers and analyzes failure data to estimate such measures as reliability and mean time to failure.

4.5.2 STATE DIAGRAM BASED TESTING PROCESS

The state diagram based testing process has following steps: create a state diagram, create a state table and create a set of test cases.

'A user wants to withdraw money from bank by using card. To withdraw user have to give PIN (Personal Identification Number). User can enter invalid pins only for three times, on fourth time machine will block the card. In case of valid PIN machine allow user to make transaction.'

Step 1 Create a state diagram.

States: START (**S1**), WAIT FOR PIN (**S2**), FIRST TRY (**S3**), SECOND TRY (**S4**), THIRD TRY (**S5**), BLOCK CARD (**S6**), ALLOWS ACCESS TO ACCOUNT (**S7**)

Transitions: Card Inserted, Enter PIN, PIN is OK, PIN is Not OK.

State Diagram:

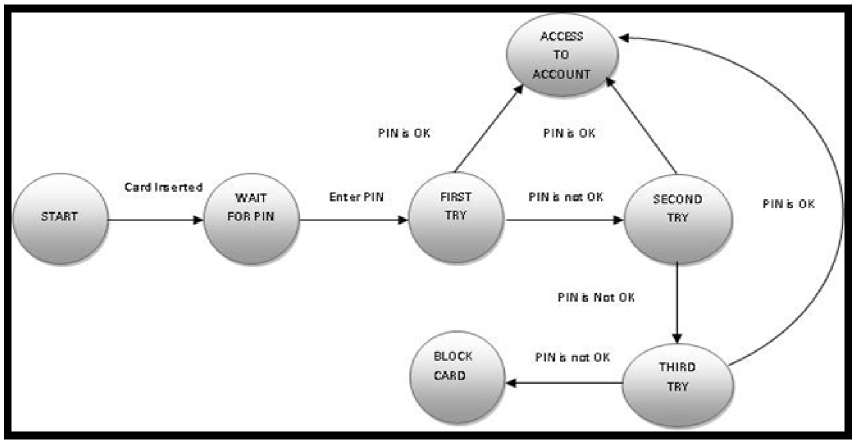


Figure 4.24 State Diagram for PIN validation

Step 2 Create a state table.

Table 1 is derived from state diagram as shown in figure 2. It contains three elements or columns 'from state', 'transition' and 'to state'. 'From State' shows current state of system and 'To State' shows the state where it reached by performing transition.

Table 4.18 State table for 'PIN validation'

From State	***Transition***	***To State***
S1	Card Inserted	**S2**
S2	Enter PIN	**S3**
S3	PIN is not OK	**S4**
S3	PIN Is OK	**S7**
S4	PIN is not OK	**S5**
S4	Pin is OK	**S7**
S5	PIN is not OK	**S6**
S5	PIN is OK	**S7**
S6	----	
S7	----	

Step 3 Create a set of test cases

For state machines test cases can be designed using ‘coverage notion’, Grochtmann [62]. All individual transition coverage is also known as o-switch coverage, coverage of all transition pairs is 1-switch coverage, transition triples coverage is 2-switch coverage, transition quad coverage is 3-switch coverage. In case of 0-switch coverage each row of state table is a test case. Using 1-swtich coverage, test cases are given. Test cases given in table 2 covers functionality ‘Allow access to account’ and ‘Block card when numbers of wrong tries are three’

Table 4.19 Test cases for ‘PIN validation’ using 1-switch coverage

Test Case 1	**Test Case 2**	**Test Case 3**
Enter PIN [S3].	PIN is not OK [S4]	PIN is not OK [S4]
PIN is not OK [S4]	PIN is not OK [S5]	PIN is not OK [S5]
PIN is OK [S7]	PIN is OK [S7]	PIN is not OK [S6]

4.5.3 ANALYSIS & CONCLUSION

In state based testing, the behaviour of a system under test is compared with the behaviour of state diagram. The issues in testing arise when there is fault in human developed model. During testing an implementation against a state machine, one study the following control faults: incorrect transitions (Incorrect result), missing transition (Event has no effect), incorrect or missing event, incorrect or missing action (as a result of a transition wrong things happen), corrupt or missing state, trap door (the implementation accepts undefined events) sneak path (an event is accepted when it should not be). Test cases can be designed by all event or states or action or transitions or n-transitions sequences or round trip path. Exhaustive testing (At least once every path over the state machine is exercised) by using state based testing is totally impossible or impractical. Finally, for state based testing, automated tool can be helpful for test case design, test case execution, monitoring and making a benchmark for validation of implementation.

4.6 CLASSIFICATION TREE BASED TESTING

Grochtmann and Grimm gave the idea of classification tree method (CTM), Singh et al[63]. It has been used in applications like an airfield lighting system and a simplified part of adaptive cruise control systems, Chen et al [64]. For test case generation, it has been enhanced by others into integrated methodology, Poon et al and Chen et al [65, 66].

4.6.1 CLASSIFICATION TREE METHOD

CTM as a black box testing technique demands from tester, to identify important aspects from the specification for the purpose of testing. Further these aspects are used as classification criteria to divide the input domain into disjoint sub-domains called classes. The relationship among classifications and classes is shown in form a tree which is called 'Classification Tree (CT)'. Test cases are generated from CT by selecting relevant combinations of classes to identify all valid combinations of classes. As far as possible references to the constraints are not captured in the classification tree, invalid combination of classes as test cases cannot be identify. The invalid combinations of classes to make test cases are infeasible and have to be removed before testing the program. The algorithm to design test cases by using classification tree method is given

```
Algorithm 1: Build Classification Tree (Param: S)
{
        // U: Functional Units          CT: Classification Tree      S: Specification
        // T: Classifications      TT: Test case Table.
        // C: Classes              FC: Feasible combination of classes
          Step1. Divide S and add each part into set U
          For each element 'e' of U repeat step 2 to 5   {
          Step2. Identify a set T and C for 'e' where each member of T is defined as the
        parameter and environment conditions (system states) of 'e'. C set has disjoint   subsets
  of for each member of set T
          Step3. Construct a classification tree CT
          Step4.Construct a test case table TT
          Step5. Identify a FC from test table. Each FC is test case.
          }// end for
}
```

4.6.1.1 WHY CLASSIFCIATION TREE BASED TESTING

According to Cohen [67] and it is obvious from algorithm 1 that CTM have well defined steps, at each step clearly defined sets are required. If specification is not too complex, then it is easy to build test cases using classification tree method, on the other way automated testing tool will be required. Specification can be formal or informal. By using both specifications it is easy use CT method. As informal specification can be used, it is also not require that tester should have development experience.

4.6.2 CLASSIFICATION TREEBASED TESTING PROCESS

Automated Teller Machine (ATM) is used by user to make transactions, cash withdrawal, and statement request, balance inquiry and change PIN as shown in figure.

Figure 4.25 ATM Machine

On base of intended use of ATM machine, for business, functional and component testing, test cases are given. For functional testing complete set of test case has been identified. For other type testing, it shows CT can be helpful for these types of testing's.

4.6.2.1 BUSINESS TESTING

- *Identify Classifications and Classes*

  Following classifications and classes has been identified by using algorithm 1 for functional unit '*Type of Transaction*'

  Functional unit {Type of Transaction}
  Classifications {Balance enquiry, statement, change PIN, cash withdrawal, other}
  Classes {yes, no, screen, print, cancel, invalid}

- *Classification Tree Construction*

  The classification tree based on classifications and classes is shown in figure 2.

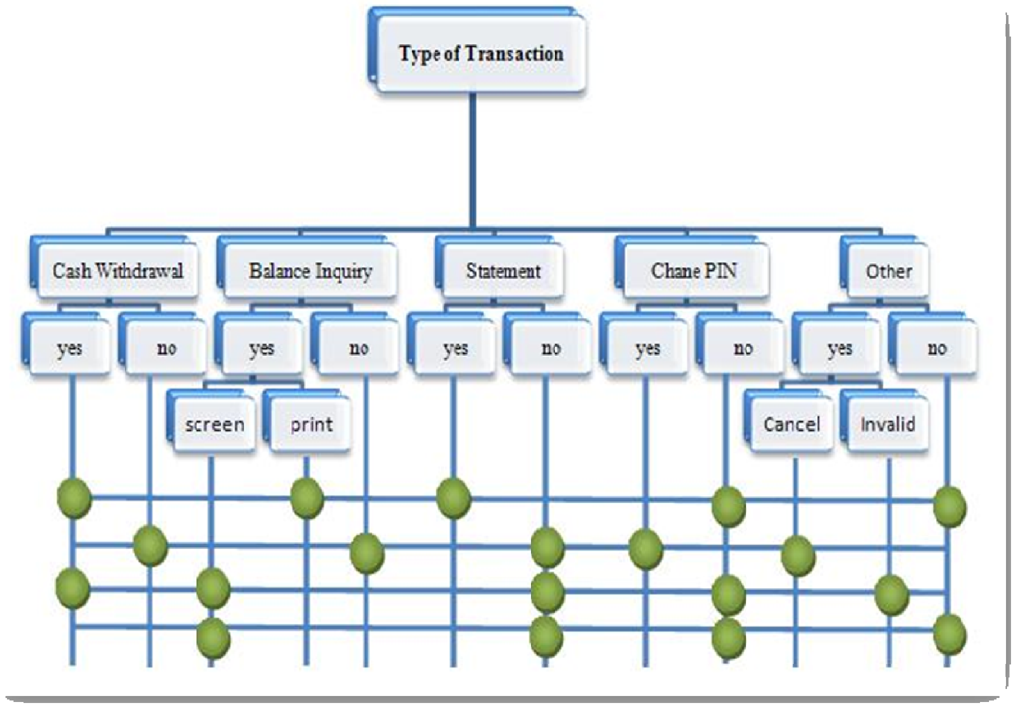


Figure 4.26 Classification Tree for 'Type of Transaction'

4.6.2.2 FUNCTIONAL TESTING

- *Identify Classifications and Classes*

  Following classifications and classes has been identified by using algorithm 1 for functional unit '*Cash withdrawal'*

  Functional unit {Cash Withdrawal}
  Classifications {amount, receipt, confirm, another transaction}
  Classes {yes, no, 10, 100, 500, other, multiple of 10, invalid, cancel}

- *Classification Tree Construction*

  The classification tree based on classifications and classes is shown in figure 3.

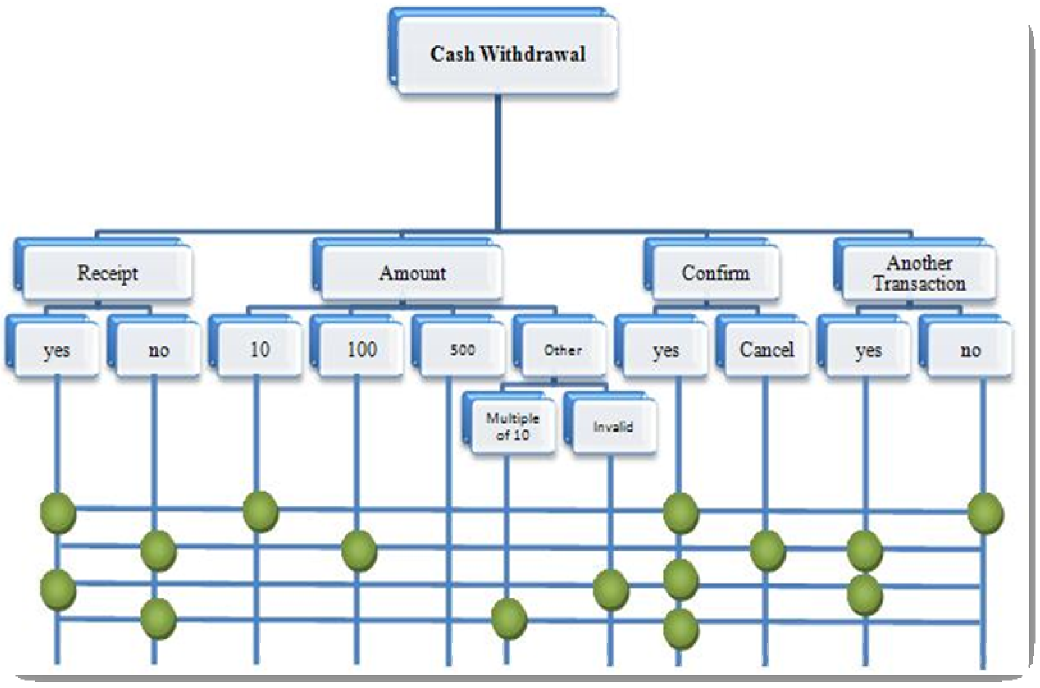


Figure 4.27 Classification Tree for 'Cash Withdrawal'

- *Test Case Table Creation*

  On base of classification tree, test case table which contains the feasible combination of classes is given below.

Table 4.20 Test cases for 'Cash Withdrawal'

Functional Test Cases				
Test Cases #	***Receipt***	***Amount***	***Confirm***	***Another Transaction***
1	Yes	10	Yes	No
2	**No**	100	Cancel	Yes
3	**Yes**	Invalid	Yes	Yes
4	**No**	Multiple of 10	Yes	--

As each combination is a feasible combination which ensure that it is a complete test case.

4.6.2.3 COMPONENT TESTING

- *Identify Classifications and Classes*

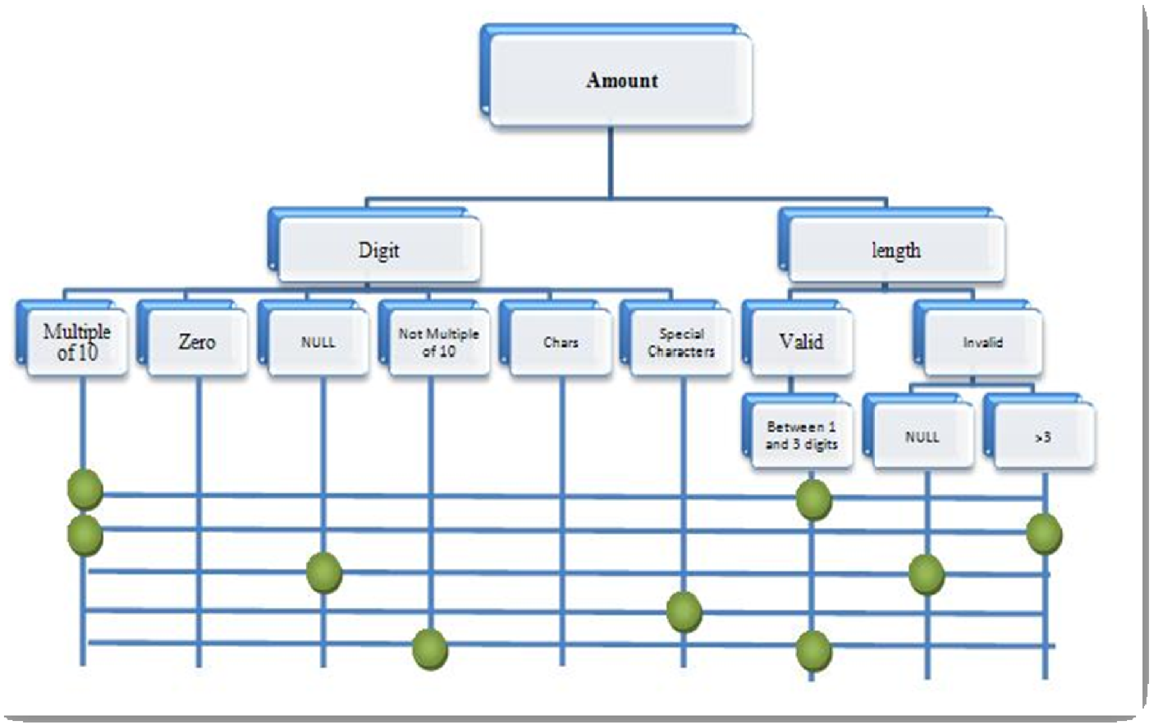
  Following classifications and classes has been identified by using algorithm 1 for functional unit '*Cash withdrawal*'

  Functional unit {amount}
  Classifications {digit, length}

Classes {multiple of 10, not multiple of 10, 0, NULL, chars, special characters, valid, invalid, >3, between 1 and 3 digits}

- *Classification Tree Construction*

The classification tree based on classifications and classes is shown in figure 4.



4.28 Classification Tree for 'Amount'

4.6.2.4 ANALYSIS & CONCLUSION

Classification tree is like Equivalence Partitioning (classes), Boundary Value Analysis, can it be used in exploratory testing, in non-functional testing, and is it unrealistic? The answer is that it can be applied with requirement specifications, user stories and use cases. It can be helpful for business testing, functional testing and component testing as illustrated above by example. Too large classification trees are harder to maintain. The number of test cases generated by applying classification tree method varies according to minimal and maximal criteria. According to minimal criteria, the minimum number test cases are number of disjoint subsets (classes) of a particular classification. The maximum numbers of test cases are result of multiplication of, total number of disjoint classes of each classification with each other. It can be calculated by using following formula:

$$NDC_1 * NDC_2 * NDC_3 * \ldots\ldots * NDC_M$$

Where

'NDC' *represents number of disjoint classes*

'M' *represents number of classification*

Classification tree based testing is a graphical way of showing test cases which can be applied at any level of testing, can be used in any organization, might complement with other testing techniques. It increases the visibility of test cases and enhances maintainability.

5 INTERVIEW DESIGN

Qualitative study is conducted to find out answers of research questions by literature review and interviewing persons related to payment card industry. Interviews are performed as a data compilation tool to extract information associated with black box testing techniques which are used for functional testing in payment card industry. For in depth study, interviews are selected in place of survey. According to Creswell [21], interview is used as a tool to perform face-to-face, one-to-one, in person interview or perform interview from telephone and group. The study design of the interview has the following parameters:

- 5.1 Interview organization
- 5.2 Semi-Structure Interviews
- 5.3 Selection of Interview Topics
- 5.4 Interview Questions
- 5.5 Process
  - 5.5.1 Ground Work
  - 5.5.2 Implementation
  - 5.5.3 Data Validation
- 5.6 Association between Interview Questionnaires and Research Questions

5.1 INTERVIEWS

It is typically believed that interview is easy technique to extract necessitate information from interviewee. On the other hand, interviews are awkward and time consuming technique when getting accurate information is required [68]. Interviews sometimes produce end result in unsatisfactory, inappropriate and insignificant information. In fact, it is concentrate of the interview to poses questions on its precise position when performing interview. Hence, it is significant point is to plans the interview in a suitable mode for obtaining the necessitate information from the issue. Researchers has selected to design the semi structured interview for discovering information on black box testing strategies for functional testing exercised in dissimilar institutes.

5.2 SEMI-STRUCTURED INTERVIEW

A semi-structured interview is relatively a casual and contented conversation that is foundation on early planned subject [69]. The main principle is to get information on black box testing techniques from the concerned person (e.g. Testing Team Lead) who is doing job in software industry. A number of questions of different type are asked and there is no limitation of conversation [69].

Semi organized interview method is performed to get information about important black box testing strategies for functional testing from software industry. This method of interview is so simple and easy to use. It exercises open ended questions; throughout the interview some question occurs as unexpected. While authors are two group partners, one partner raises questions for the period of interview and other partner writes are necessity questions. If there are some questions are missing throughout the interview then one partner can inquired with other interviewee on the closing stages of the interview.

A group of open ended questions is arranged ahead of the interview for authorizing interviewee, to express his observations during the discussion. Though homework, authors maintain questions straightforward and produce rational cycle connecting them for handling the current information. The interview is a qualitative approach but authors have also get information from quantitative data. The interview is initiated to pose general question and then deeply highlight the practical facts associated to black box testing strategies. The advantage of this interview technique is to difficult questions are simply purified throughout the interview, and researchers can goes deeply to extract information.

5.3 INTERVIEWEES SELECTION

This research testimony is based on information obtained from academia, literature review and practiced in software organization. The interviews are taken with related contact person fitted specifically in software testing within software industry. The interviewee selection criteria are based on methodology as described in section 2.2 in phase 3 study, as follows.

1. The contact person must be engaged within quality assurance department, specifically in software testing.
2. The contact person must be in a higher post e.g. while quality assurance manager, testing team leader and have sound experience.
3. The contact person must be part of software quality process assessment team.
4. The contact person must be part of level 1 or level 2 teams (which handles runtime errors and fixe those errors with in minimum time).

5. The contact person must have experience of testing of business critical modules (e.g. Commission distribution, fund transferring from bank to bank, transactional processing etc)
6. The roles and responsibilities of the contact person must be the same as explained above in criteria 2 for all companies.

5.4 INTERVIEW QUESTIONS

Writers planed interview questions with the assistance from literature surveys, also previous interview experiences of the software industries. The writers initiated work from brainstorming when designing the interview questions that are directly connected to the black box testing strategies and acknowledged maximum ways to cover for black box strategies research questions. Later than writers creates relevant question for interviews. These questions are structured with assistance of literature study, group meeting, also discussion between concerned persons and removed those questions that are not related. At the end, opinions and interpretation from advisor and faculty reviewer, interview questionnaire is created. Authors used the rehearsal interviews to improve the interview questionnaire.

After interview exercise, the questionnaires were improved but time is required to finalize them. Writers also include their friends for these interview processes that are familiar with software testing and have industry experience, particularly with black box testing strategies. Researchers gathered data for both quantitative and qualitative from the interviews. After that all purified answers are placed in the Microsoft Excel sheets; tables and figures are gathered from the answers. Some questions are redundant and were removed. This method was replicated in anticipation of interview questions were planed for research questions. Many questions created positive decisions for the purpose of getting hundred percent results. Lastly, researchers forward interview questionnaires (Appendix 1) to the advisor for last sanction.

5.5 PROCESS

After finalization of interview questionnaire, authors started to contact with companies in Pakistan. The subsequent parts shows the sequence of the interview practices are taken. Further grounding the interview processes and validity of the data is analyzed in the coming sections.

5.5.1 GROUND WORK

The authors contacted the related persons (quality assurance managers, project managers and testing team leaders) in the desired organizations via telephones and emails, these contacted persons functioning in different software organizations, intended for the reason to performing interviews. Through this negotiation via telephone and emails with interviewee, authors provide them general thought about the interview. Also gave them particulars methods and description of the interview questionnaires. Actually, software organizations is applying these methods and actions, however they rigidly distinguish the authentic names of these methodologies, point out the mistakes in material which authors provided to them. That shows understanding, strong grip and experience of intellectuals in their field. Authors accomplish interview in three manners.

1. Interviews with Pakistani software organizations via Skype discussion.
2. Interviews with Pakistani software organizations via telephonic interview.
3. Interviews with Pakistani software organizations via email communication.

5.5.2 IMPLEMENTATION

By performing authentic interview, researchers have communicated every aspect of the questions that are specified in the questionnaires for the reason of research and explanation, when going to perform authentic interview. As performing interviews through dissimilar software organizations, researchers did not obstruct in the conversation of the contacted person except he diverge from inquired questions. When authors unintentionally asked questions from wrong angles, the interviewee responds quickly to correct them which showed contacted person interest and grip on method or technique or domain. Writers performed interviews in that way one writer contributed to prepare all notes for the purpose of that there are no data is missed in the interview. Interview is not recorded may be interviewee feel hesitation within interview this some extent not provide exact information.

5.5.3 DATA VALIDATION

Data validation is very important to remove confusions and answers verification collected as a result of interview. Answers are verified from the interviewee it helped authors to understand and verify the answers. In case of ambiguity left in gathered information, then this is cleared through asking by telephone and email. The ultimate aim is to find answers of research questions from software companies of Pakistan. Multiple people were involved in interview process; interviewee is not sure about the exact answers, so authors conduct multiple interviews to know the final answers.

5.6 ASSOCIATION BETWEEN RESEARCH QUESTIONS AND QUESTIONNAIRE

Q1 What are the most important black box testing strategies***?***

Authors have done a literature study to identify important black box testing techniques for functional testing that can be used in software industry. This question is directly considered for interview to find its importance from industrial point of view. This research question is directly asking about black box testing strategies used in software industry because there are several testing techniques, proposed in literature but not practice in industry. So, authors mentioned an interview question for that is directly linked to this research question. E.g.:

Which black box testing strategies you perform for your projects?

Do you use equivalence partitioning as black box testing strategy for your projects?

Do you use boundary value analysis as black box testing strategy for your projects?

Do you use decision tables as black box testing strategy for your projects?

Do you employ use cases as black box testing strategy for your projects?

Do you use classification tree method as black box testing strategy for your projects?

Do you use state diagrams as black box testing strategy for your projects?

Why this technique is important for your organization as black box testing strategy?

Q2 What is the primary purpose of each black box testing strategies***?***

Authors have done a literature study to know about the primary purpose of each black box testing strategy which is presented in chapter 4. This question is directly asked to interviewee to know their view points, how they perceive from different angles. So authors directly linked to this research question. E.g.:

What is primary purpose of this technique?

Q3 What are the advantages and disadvantages of important black box testing strategies***?***

This research question is discussing about advantages and disadvantages of black box testing strategies which shows that there are advantages and disadvantages which are associated to it. Further if any organization use or do not use these techniques then its benefits and limitations are attached to that organization. In order to get organizations view points about this question, authors have directly developed a question which is associated to this research question. Authors asked about advantages and disadvantages of each of the important black box testing strategy which is used in software industry. This question is directly mapped to our interview questionnaire, e.g.:

Can you provide two advantages and two disadvantages of decision table as black box testing strategy?

Q4 How complementary are various combinations of important black box testing strategies? Is there any best possible combination?

This research question is about to find out how various techniques are complement to each other. If any organization is using one or more techniques combinations then how much benefit software organizations are getting in terms of most defect guessing, defect catching, high quality test cases, normal or robust testing etc. Authors develop an interview question to find out view points of organizations. Authors asked questions about complementary nature and best combination of techniques. This question is directly mapped to our interview questionnaire, e.g.:

Is this technique is complementary with combination to other technique?

Authors have analyzed the results after conducting interviews, extracted information about important black box testing strategies, their primary purpose about different organization point of view, the benefits an organization is getting by using particular or combination of these techniques. Authors analyzed the perceived advantages and disadvantages of the black box testing techniques.

6 INTERVIEW FINDINGS

In this chapter researchers have presented academic passage from the interview findings, which are performed in five companies in Pakistan. The tables as well as texts are used to show the related data of the interview findings for black box testing strategies exercised in the subjected software companies. The researchers were taken the interview to gather and properly saved all information regarding black box testing strategies. The brief introduction of all five companies is presented in the following coming sections of this chapter.

6.1 INTRODUCTION OF ORGANIZATIONS

This research thesis contains the analysis of five organizations of Pakistan. These five companies are located in different cities in Pakistan. This report is suitable to the person who reads the information in context of analysis and comparison of black box testing strategies in the desired companies. The researchers are not allowed to mention the selected software company's names due to privacy factors of organizations. The brief introductions about organizations are collected from their websites as well from the contacted persons.

6.1.1 ORGANIZATION A

This organization offers facilities of geo technology and information technology to governments and businesses. They have over 600 highly skilled staff for clients to improve their outfitted competence to all over the world. This organization declared as a CMMI level 3. They present automated and manual testing strategies like, functional testing, load/stress testing, performance testing, unit testing, API testing etc.

The contacted person of this organization is functioning in software testing department as a senior Quality Assurance engineer from the last five years. He is engaged within software quality assurance department. His responsibilities are to construct and run specialized tests to check product functionality. He build test plans, test cases and detailed description of software defects and participate in manual tests projects that accomplish the responsibility of senior quality assurance engineer. The selection of the contacted person was taken from the procedure that is described in (5.3).

6.1.2 ORGANIZATION B

This organization provides self services solutions and complete maintenance for retail, finance, travel, healthcare and hospitality to all over the world. This organization is CMMI level 5 certified. They are working in different software testing techniques.

Our contact person in this organization is working as a project manager in quality assurance department. This person is functioning in this organization from the last six years and also have more than five years background experience. The interviewee is managed the software testing activities in this organization.

6.1.3 ORGANIZATION C

This organization expertise and provides financial solution for trusted payments cards. They have more than 300 employees worldwide. They are certified by CMMI level 3. This organization provides complete and suitable solution to facilitate clients for the purpose of increase revenues, recover competence and reduced costs.

The interviewee in this desired organization is working as level 1 (as discussed in section 5.3) team lead. This person is responsible to guarantee the testing events that are properly defined for the purpose of quality. He is engaged in software testing procedures of business critical applications.

6.1.4 ORGANIZATION D

This organization was established in 1977 in Pakistan. They are playing role in some largest IT projects in the country. They are providing services and products to a growing list of corporate clients. They have more than 700 professionals. They offer financial solution to banks and government sectors organizations. They are successfully appraised for certification of CMMI Level 3.

The interviewee 'D' is functioning as a test engineer in quality assurance department at software organization D. He is functioning in this organization since 2004. The interviewee is involved to build, design, test and deploy effective manual test solutions.

6.1.5 ORGANIZATION E

This organization is the global business services and enterprise application solutions provider to all over the world. This organization was come into being in 1995. They are committed to quality demonstrated by their achievements CMMI maturity Level 5 assessment. They have more than

200 employees in Pakistan office. They have separate quality assurance department that are providing testing support to all the projects.

Interviewee 'E' is working as a testing team lead in quality assurance department at software organization E. He is working in software organizations since 2005. The contacted interviewee is involved in the test scope finalization activity, test planning and release test plan. Interviewee 'Y' is fulfilling the selection requirements for performing interviews for discovering black box testing strategies exercised in software industries.

6.2 IMPORTANT BBTS EXCERCISED IN ORGANIZATIONS

Black box testing strategies (BBTS) which confirms the word 'Importance' dimension given in figure 1.2 and performed in software organizations are accessible in the table 1 below. In the columns of the table shows the organization names and rows presents black box testing strategies respectively.

COUNTRY	PAKISTAN				
Organizations ***BBTS***	Organization A	Organization B	Organization C	Organization D	Organization E
Equivalence Partitioning	Yes	Yes	Yes	Yes	Yes
Boundary Value Analysis	Yes	Yes	Yes	Yes	Yes
Decision Tables		Yes	Yes	Yes	
Classification Tree Method					
Use Case Based Testing	Yes	Yes		Yes	Yes
State Diagram			Yes	Yes	

Table 1 Shows most important BBTS practiced in Pakistan based Organizations

Table 1, shows the most important black box testing strategies are applying in the selected organizations. The most important strategies in terms of most defect finding and frequently used black box testing strategies (BBTS) in their organizations.

The most defect finding and frequently used techniques in the organization A is that equivalence partitioning, boundary value analysis and use case based testing. These three black box testing strategies are the most important techniques are applying in this organization.

The most frequently used techniques in terms of defect finding techniques in organization B is illustrates as; equivalence partitioning, boundary value analysis, decision tables based testing and use case based testing.
The most important black box testing strategies for the purpose of most defect finding and frequently used techniques in the organization C is described as; equivalence partitioning, boundary value analysis, decision table based testing and state diagram based testing.

The most important black box testing strategies in the organization D is exercised as; equivalence partitioning, boundary value analysis, decision table based testing, use case based testing and state diagram based testing.
The most important black box testing strategies to the intention of defect finding and frequently used techniques in the organization E is presented as; equivalence partitioning, boundary value analysis and use case based testing.

6.3 PURPOSE OF EQUIVALENCE PARTITIONING AS BBTS

All five organizations are exercising the equivalence partitioning for the purpose of black box testing strategies. The information regarding equivalence partitioning is described below:

Organization A is performed equivalence partitioning for the purpose to define the system into equal partitions of similar inputs data to minimize the testing efforts and to reduce test cases for a finite set of possible test cases.

Organization B is exercised equivalence partitioning to generate minimum set of data to test the valid and invalid inputs and to check that software works according to the type of data is entered.

Organization C is also practicing equivalence partitioning as black box testing strategy. According to interviewee 'C' the primary purpose of this technique is to decrease test cases reasonably as well as to find errors from small test cases.

Organization D is using equivalence partitioning technique as a black box testing strategy for functional point of view. The primary purpose of equivalence partitioning according to interviewee 'D' is that it reduce the test cases by combing tests and increase performance to detect errors from nominal test cases.

Organization E is employed this black box testing strategy to generate test cases for findings errors. They exercised this strategy for some projects toward outputs of software mechanism.

They used equivalence partitioning as to check that software acts as required and outcomes from test cases are valid or not.

6.3.1 PROS AND CONS OF EQUIVALENCE PARTITIONING

All selected five organizations are applying equivalence partitioning strategy as a black box testing strategy. There are number of precise pros and cons which are written as follows; the pros from equivalence partitioning as BBTS in organization ‘A’ are illustrated as; this strategy helps the organization to find defects and also reduced the test cases for increasing the performance of the test suits. The cons of equivalence partitioning as BBTS from the organization ‘A’ are explained here as; this techniques not provide them guideline for selection of the input and also this strategy is not carry out test for all inputs.

The pros of equivalence partitioning as BBTS from the organization ‘B’ are listed here as; this strategy is very helpful for saving their time for replication of testing efforts and also suitable for minimize test cases to verify the system working properly. The cons of equivalence partitioning as BBTS from the organization ‘B’ are is illustrates as; According to contacted person this strategy is heuristic type and sometimes it does not assure the subsistence of the best tests.

The pros of equivalence partitioning as BBTS from the organization ‘C’ are explained; testing attempts are made smaller and treatment is also guaranteed. It is helpful to find errors. The cons of equivalence partitioning as BBTS from the organization ‘C’ are express as; this strategy is not to test all inputs and in some situations it is hard to trace errors.

The pros of equivalence partitioning as BBTS from the organization ‘D’; it save their time to shrink the test cases and also decrease test risks that tell early removing bugs. The cons of equivalence partitioning as BBTS in the organization ‘D’; According to the concerned person logical mixtures are not defined properly in this technique and also this technique creates problem when implementation of the program is describe dissimilar as tester thinks from other way.

The pros of equivalence partitioning as BBTS from the organization ‘E’ are described; in some projects data types and conditions are provided then equivalence partitioning is very simple and find it as easy testing strategy. This reduces the cost in terms of test maintenance and identifies faults as much as possible. The cons of equivalence partitioning as BBTS according organization ‘E’; input selection instructions are not given in this technique and not test all inputs.

6.4 PURPOSE OF BOUNDARY VALUE ANALYSIS AS BBTS

Boundary value analysis is another black box testing strategy, that is exercised in all subjected organizations and therefore authors have performed interviews. In this section interview results related to purpose of boundary value analysis is illustrated as:

The primary purpose of boundary value analysis in the organization 'A' is described as; it is used to grasp common user input faults that can disturb functionality of the program; it is used for the purpose of to discover errors at boundaries. This technique is performed to cover boundary, external boundary, distinctive values and faulty values.

The purpose of boundary value analysis in the organization 'B' is illustrated as; it is used to recognize defects on boundaries very easily, the primary purpose of this approach is to select data type to test inputs that comprise maximum value, minimum value, error value and abnormal values.

The purpose of boundary value analysis in the organization 'C' is presented as; it is used to reduce test cases by using with equivalence partitioning and the aim of this technique is to select the test cases for those boundary values that fail in equivalence class partitioning values. The boundary value analysis offers organized process for describing quality of software and comprehensiveness.

Organization 'D' is performing boundary value analysis as to classify test cases at boundaries. This organization uses both techniques because boundary value analysis is the next step of equivalence partitioning for test case selection on the edges of the equivalence class. They use boundary value analysis to minimize time for test efforts as well as cost.

Organization 'E' is applying boundary value analysis technique for black box testing in order to find defects as well as minimize test cases. They easily established defects at boundaries by using boundary value analysis.

6.4.1 PROS AND CONS OF BOUNDARY VALUE ANALYSIS

All subjected organizations using boundary value analysis as a black box testing strategy. There are number of particular advantages and disadvantages; the advantages from boundary value analysis as BBTS in organization 'A' are given as; the capability of boundary value analysis to find defects is high as compared to others. Boundary value analysis authenticates that program works properly for any condition input or output. It can be used on equivalence class for user input on screen as well as on time ranges or table ranges. Boundary value may also be used for test data selection. The disadvantages of boundary value analysis as BBTS from the organization A are explained as; it is not test all likely inputs as well as it is not test dependencies within pairs of inputs.

The advantages of boundary value analysis as BBTS from the organization ‘B’ are listed here as; boundary value analysis finds errors better than equivalence partitioning. Boundary value analysis values focuses on outside the limits and also helpful for exception handling. The disadvantage of boundary value as BBTS from the organization ‘B’ are illustrates as; sometimes costly technique for projects. Boundary value analysis technique with equivalence are together easy to generate test cases but still not cover for selection the existing data.

The advantages of boundary value analysis as BBTS from the organization ‘C’; boundary value analysis is very helpful if input variables pass on to physical quantities. The minimum human effort is required for setting up test suites. The disadvantages of boundary value analysis as BBTS from the organization ‘C’ are; boundary value analysis does not work well for Boolean variables. By using boundary value analysis experience is needed to build up high quality test cases.

The advantages of boundary value analysis as BBTS from the organization ‘D’ point of view; boundary value analysis is the most suitable technique to select the unsuccessful points in the data. Boundary value analysis is good technique for robustness testing because it focuses on outside the boundary limits. The disadvantages of boundary value analysis as BBTS from the organization ‘D’; test cases are obtained exclusive of judgments of function. The combinations of additional features make difficult things.

The advantages of boundary value analysis as BBTS from the organization ‘E’ is; this technique helpful for them to discover maximum possible faults from small set of tests. They performed boundary value analysis with equivalence partitioning technique to produce test cases. The disadvantages of boundary value analysis as BBTS from the organization ‘E’; Boundary value analysis does not test all likely inputs and it does not test dependent inputs each other.

6.5 PURPOSE OF DECISION TABLE AS BBTS

Organization ‘A’ is not performing decision table based testing as test case design technique for black box testing because their testing requirements are fulfilled with equivalence partitioning and boundary value analysis.

Organization ‘B’ is applying this as software testing design technique in black box testing for functional behavior. The main purpose of this strategy according to the desired organization: it is used to detect test cases, discover decision variables and conditions. According interviewee this strategy focuses on logical dependencies that enhance efforts to identify test cases. When they are applying this strategy it reduces test combination as compared to other techniques.

Organization ‘C’ is using decision table as black box testing strategy for functional point of view. The primary purpose of this strategy in their organization is to identify test cases that easily cover all test cases. Decision table based testing is ideal for them in some situations like numbers of combinations of events are selected for group of conditions. By using decision table they generate test cases. All information regarding complex business rules sometimes are not documented properly then their testers collects information from decision tables.

Organization ‘D’ perform decision table as black box testing strategy for function testing. Because decision table based testing does not require too much efforts to generate test cases. Hence it is less time consuming technique for them.

Organization ‘E’ does not exercise decision table based testing strategy due to time constraints. According to them it is not helpful to find more defects as others. It is not useful for them because when there are large numbers of defects occurs.

6.5.1 PROS AND CONS OF DECISION TABLE

The pros of decision table as BBTS from the organization ‘B’ is discussed as; due to logical dependencies this strategy increase performance of test cases. According to the interviewee ‘D’ decision table testing discover testing directions that other testing strategy avoids. The cons of decision table as BBTS from the organization ‘A’ is discussed as; too much effort and domain knowledge is requires to generate test cases.

The pros of decision table as BBTS from the organization ‘B’ are explained as; according to interviewee ‘B’ decision table are sufficient covers test cases identified, Also decision tables allows them to detect potential error in their specification. The cons of decision table as BBTS from the organization ‘C’ are explained as; over head test cases are not applicable and not minimize test cases.

6.6 PURPOSE OF CLASSIFICTION TREE AS BBTS

All subjected five organizations are not exercising classification tree method as BBTS for functional testing and the reason why they did not use this technique is illustrated as;

Organization 'A' does not perform classification tree method because this strategy is not applicable for large systems. The classification tree method is not suitable for their projects and as well as for their domain.

Organization 'B' is also not exercise classification tree method as BBTS in their organization; because testers experienced is required classification tree method.

Organization 'C' did not apply classification tree method as BBTS for their organization; because their projects requirements are achieved with other black box testing strategies. Hence it is not applicable for them.

Organization 'D' did not use classification tree method as BBTS for their organization; because some test case are linked with white box testing like path coverage, so when they apply classification tree method it removes some infeasible test cases this leads to hurdles for generation test cases.

Organization 'E' did not implement classification tree method as BBTS for their organization; when different people use this strategy and they get different results from classification tree and test suits this is more complicated for them.

6.7 PURPOSE OF USE CASE AS BBTS

Use cases based testing is another black box testing strategy, that is exercised in some organization and one organization do not perform this strategy, therefore authors have performed interviews. In this section of interview results related to use cases based testing are illustrated as:

Organization 'A' is performing use cases as BBTS. The primary purpose of use cases based testing according to interviewee 'A' is that; this technique is most near to real world usage of the software and it can reveal most critical bugs in software products.

Organization 'B' is exercising use cases as BBTS. The primary purpose of this technique according to interviewee 'B' such as; it extract the functional requirements from users. Use cases are used as an input for test case improvement.

Organization ‘C’ does not use, use cases as BBTS. Because use cases are not used to specify defect finding, it is not easy to test the whole product because many actors/behaviors depends upon the other actors/behaviors. Non fictional requirements are difficult to capture using use case testing.

Organization ‘D’ is applying use cases as BBTS. According to interviewee ‘D’ the primary purpose of use cases as black box testing strategy is that: use cases are used to validate the user requirements, traceability of requirements and early detection of requirements defects. Because it helps to easily map the test coverage of requirements.

Organization ‘E’ is utilizing use cases as BBTS. According to the interviewee ‘E’ the primary purpose and importance of this technique is illustrated as; it is used to uncover defects in the process flow during real world use of the system. Use cases are also exercised to manual black box testing. A use case based testing technique provides prototype test cases. It is the most defect finding technique.

6.7.1 PROS AND CONS OF USE CASE TESTING

Use Case process as BBTS practiced in five organizations; it has some precise pros and cons described as:

The pros of use cases as BBTS in organization ‘A’ discussed as; it reveals most bugs in the application and better test coverage. The cons of use cases as BBTS discussed as; increases work effort and complexity.

The pros of use cases as BBTS in organization ‘B’ are explained as; it is useful in uncovering defects in the process flow during real world use of the system and designing acceptance test with user participation. The cons of use cases as BBTS in organization ‘B’ are explained as; it is not helpful for component testing and not used to specify quality in terms of defect finding.

The pros of use cases as BBTS in organization ‘D’ are illustrated as; widely used to test the right thing and gives complete picture of the software functionality. The cons of use cases as BBTS in organization ‘D’ are; it is time consuming and not includes detail regarding user interface or screens.

The pros of use cases as BBTS in organization ‘D’ are; easy to understand and language independent. The cons of use cases as BBTS in organization ‘D’ are; non functional requirements are difficult to detain and level of abstraction for use case is complex.

6.8 PURPOSE OF STATE DIAGRAM AS BBTS

State diagrams are practiced as black box testing strategies for functional testing only in two organizations. The information related to state diagrams as BBTS is illustrates as following:

The state diagrams as BBTS in organization 'A' are not performed because; It can not test every possible condition of application in which defects occurs. To design state diagram from a program is difficult and it is complex technique because it increases states and events.

The state diagrams as BBTS in organization 'B' is not applied because; time consuming, too much effort required.

The purpose and importance of state diagram as BBTS from organization 'C' are explained as; it identify finite number of tests and infinite test scenarios. State diagram are used as design specification for test case generation. It is important when test cases are designed to check the failure of database and system behavior.

The purpose and importance of state diagram as BBTS exercising in organization 'D' is discussed as; it is used to determine a sequence of events and all transition in state diagram should be tested at least once. State diagram identify test cases from invalid and unrelated events. It is used to create test cases for detect defects that are not covered.

The state diagrams as BBTS in organization 'E' are; it is not useful for describing the collaboration between objects that cause the transition. Hence it is complex and time consuming.

6.8.1 PROS AND CONS OF STATE DIAGRAM

State diagrams are exercised as black box testing strategies for functional testing only in two organizations. The pros and cons related to state diagrams as BBTS is described as:

The pros of state diagram as BBTS in organization 'C' are; easy to understand the testing flow and to detect defects in implementation that facilitate invalid paths from one state to another. The cons of state diagram as BBTS in organization 'C' is; takes too much time and not applicable for real time events like ATM transaction.

The pros of state diagram as BBTS in organization 'D' are discussed as: easy to use and to useful for testing user interface. The cons of state diagram as BBTS in organization 'D' are discussed as; in high risks system too much effort requires and not best when system has no state.

7 INTERVIEW FINDINGS ANALYSIS

Researchers have analysed the necessary actions to organizing this thesis report from interview design, performing interview, and data analysis and data validity. Researchers gathered a complete data from semi structured interviews. The researchers are providing only the answers of directly or indirectly related research questions. The important information regarding research question was not discarded by the researchers. After data gathering from interview, researchers have analysed the data from many perceptions and organized different issues and connect to each other. The different issues are connects each other and analyzed the data from different point of view to get appropriate results. The group meeting was arranged both thesis partners to discuss important factors from interview findings.

7.1 DATA ANALYSIS WITH RESEARCH QUESTIONS

This section describes to satisfy research question from interview results. The relationship between research questions and interview questionnaires provides helps to analyse the data from different perceptions of black box testing strategies. The data analysis is done by using literature review, contribution done in testing techniques in terms of robust, exploratory, number of test cases as in chapter number 4, views of experienced people in financial field and interview questionnaires to answers the whole research questions. Interview findings shows only the results from interview questionnaires but this section presents complete analysis and answers that satisfy research questions. The followings sections showing data analysis with each research questions. The research questions are presents as RQ1, RQ2, RQ3, and RQ4 explicitly.

7.2 DATA ANALYSIS WITH QUESTION 1

What are the most important black box testing strategies?

The answer of this research question according to word 'important' dimensions as given in figure 1.2 is provided into chapter number 6 of interview findings. The interviewee answers questions according to established dimension. Researchers have analyzed that selected organizations exercised different black box testing strategies and procedures according to their projects requirements. Later than performing interviews then researchers discovered the following are the mostly used black box testing strategies and most defect finding techniques in the software organizations:

1. **Equivalence Partitioning:** All five organizations are performing Equivalence Partitioning.
2. **Boundary Value Analysis:** All five organizations are performing Boundary Value Analysis.
3. **Decision Table Based Testing:** Three out of five organizations are using decision table as BBTS.
4. **Classification Tree Method:** No one organization is using classification tree method.
5. **Use Cases Based Testing:** Four out of five organizations are using use cases BBTS.
6. **State Diagram Based Testing:** Two out of five organizations are using state diagram BBTS.

The authors analysed the black box testing practices of all five organizations, in which they are performing equivalence partitioning, boundary value analysis, decision table based testing, classification tree method, use cases based testing and state diagram based testing as their black box testing strategies (BBTS). These are the essentials and frequently used BBTS in software organizations now days. These black box testing strategies for functional testing are important in terms of most defects finding and frequently used according to the desired organizations of Pakistan. There are five organizations; organization A, B, C, D and organization E are practicing equivalence partitioning as BBTS. Boundary value analysis as BBTS are also used in all five organizations such as organization A, B, C, D and organization E. Decision table as BBTS are used in only three organization that are; organization B, C and organization C. Classification tree method as BBTS are not used in any organization. Use cases as BBTS are exercising in four organizations such as; organization A, B, D and organization E. State diagram as BBTS are performed in two organizations that are; organization C and organization D.

7.3 DATA ANALYSIS WITH QUESTION 2

What is the primary purpose of each of the black box testing strategy?

The authors raised question related to primary purpose of each BBTS from different angles like quality of test cases, most defect finding etc. The contacted person gave answers and their answer of this research question according to the organizations point of view is provided into chapter number 6 of interview findings. But here authors analysed the purpose of each black box testing strategies according to the research question.

7.3.1 PURPOSE OF EQUIVALENCE PARTITIONING AS BBTS

Equivalence partitioning as black box testing strategies (BBTS) is exercised in all five organizations of Pakistan, in which authors performed interview studies. The outcomes of questions related to primary purpose of equivalence partitioning as black box testing strategies are presented in section 6.3 of interview findings. The analysis and comparison to primary purpose of equivalence partitioning as BBTS are followings:

The purpose of equivalence partitioning in all five organizations are more or less the same and are listed below.

- Authors analysed that the purpose of equivalence partitioning is used to reduce test cases for a finite set of possible test cases. It is defect finding technique so it requires minimum test cases to find errors in the whole system as well as when test cases are nominal then automatically testing efforts will be low. Hence it is most defect finding and easy technique for all organizations.
- Authors examined another purpose according to organizations perspectives it is used to testing the valid and invalid inputs for detecting minimum set of data and also to verify software works according to the type of data is entered. It is basically used to getting all possible test cases and categories into classes of valid invalid and select one input value from each class for testing. These input values are used for testing the software and it verifies that this software working according to values that are entered. If is not working properly for these value then errors identified by this technique.
- Authors observed the purpose of this technique; it is used to decrease test cases and detect errors. It requires reasonable test cases to testing the software for finding the defects. Too much test cases need extra efforts and performance but in equivalence partitioning; it needs smallest amount of test cases as well as it increase performance to defects finding.
- The authors monitored that most of the organization purposes equivalence partitioning to used for checking that software acts as required and outcomes from test cases are valid or invalid means defects detection technique that requires nominal test cases from each partitions classes to find most errors and minimize testing efforts.

7.3.2 PURPOSE OF BOUNDARY VALUE ANALYSIS AS BBTS

Boundary Value Analysis (BVA) is practiced in all five organizations, in which interview studies has been conducted. The answer of the questions associated to 'Purpose' of BVA is described in section 6.4. The purpose of BVA has been analyzed as follows:

- Authors analyzed the purpose of BVA that it is a way of grasping common user input faults which can disturb functionality, to discover errors at input boundaries. This technique is performed with perspective of normal, strong and robust testing.

- Authors observed that it is an easy way of recognizing defects on input boundaries. It is a way of selecting data according to Max+, Max, Max-, Nominal, Min-, Min, Min+ baseline. It is a way of producing test cases by using it with equivalence partitioning which it misses. It is a way of software invention and comprehensiveness. It is a way of minimizing defects because it considers values on boundaries.

7.3.3 PURPOSE OF DECISION TABLE AS BBTS

Decision table (DT) as BBTS is exercised in three out of five companies. The outcomes of questions related to primary purpose of Decision Table as BBTS are presented in section 6.5 of interview findings. The purpose of Decision table as BBTS has been analyzed as follows:

- Authors examined that the purpose of DT to perform functional tests, to design test cases, discover decision variables and conditions and focus on logical dependencies.

- Authors observed that it is a way of finding defects and it handles situations in which numbers of combinations of events are selected for group of conditions or verification and validation of complex business rules is required.

7.3.4 PURPOSE OF CLASSIFICTION TREE AS BBTS

Classification tree method is not exercised as BBTS in any organization out of five. The reasons why companies do not prefer classification tree is presented in section 6.6. The authors analyzed the purpose of Classification Tree method as follows:

- Authors analyzed that its main purpose is to identify important aspects from specification for purpose of testing and use these aspects to divide input domain into classes.

- Authors examined that its purpose is identify valid and invalid combinations of classes and perform business, functional and component testing. Further it can be applied with requirement specification, user stories.

7.3.5 PURPOSE OF USE CASE BASED TESTING AS BBTS

Use cases are used as BBTS in four organizations. The results related to primary purpose of use cases as BBTS are presented in section 6.7. Its purpose is analyzed as follows:

- Authors observed that its purpose is to capture requirements related to system functionality. It is used as way of capturing inputs, outputs and functionality matters. Its purpose is to use it as a way of communication, understanding requirements.

- Authors examined that its purpose is to hear needs of users, identify features, write use cases, identify scenarios and finally write test cases. It is way of expanding vision about requirements, design and implementation. It provides prototype test cases and it can be viewed as most defect finding techniques

7.3.6 PURPOSE OF STATE DIAGRAM BASED TESTING AS BBTS

State diagram is practiced as BBTS in only two organizations. The outcome related to primary purpose of it, is presented in section 6.8. The authors analyze its purpose as follows:

- Authors examined that its purpose is to determine a sequence of events and all transitions in state diagram, to identify test cases for valid or invalid events. Its soul purpose is to identify finite number of tests and generate infinite test scenarios.

- Authors observed that its soul purpose is to promote understanding and provide a reusable framework for product development, to capture knowledge about a system and then using this knowledge as the system grows. It is used for structural analysis, test plan developed and test coverage is examined.

7.4 DATA ANALYSIS WITH QUESTION 3

What are the advantages and disadvantages of important black box testing strategies?

The questions regarding advantages and disadvantage of black box testing techniques was directly asked. The companies answer this question from their own angle.

7.4.1 ADVANTAGES AND DISADVANTAGES OF EQUIVALENCE PARTITIONING AS BBTS

Equivalence partitioning (EP) is used in all five companies. The results of interview questions related to advantages and disadvantages of equivalence partitioning are listed in section 6.3.1 and further discussed and analyze as follows:

- EP is easy to learn, simple and most defect finding technique. It is like brainstorming activity. The test

- EP is sophisticated generate test cases which are helpful to tester. The test case sophistication depends on tester choice of selecting type of equivalence class such as weak normal, strong normal, weak robust or strong robust equivalence class.

- EP requires moderate level of effort to generate test cases. Authors observed that by putting moderate level effort, EP produces equivalence relations which are help full for testing purpose. The test case generation activity is not challenging.

- EP is versatile in the sense that controlled changes, such increasing and decreasing of number of test case. Author noticed that EP has versatility; the number of test cases can be controlled on base of single fault and multiple fault assumptions.

- EP facilitates early test case generation and helps identifying any problem before coding starts. As project in its initial phases but test generation activity can proceed parallel with software process phases

- EP is independent from program logic. A tester does not need to know internal working, testing paths or program structure to generate test cases. They only need the type of data.

The disadvantages are:

- EP is like heuristic type and does not ensure best test cases. How much your heuristic is strong for identification of equivalence classes determines test case efficiency for finding most defects.

- EP does not provide any guide line for input selection criteria. The domain knowledge is required.

- EP ensures good test cases if equivalence relation is good. The equivalence relation depends on selection of equivalence class types like weak, strong and robust etc.

- Equivalence relation validity depends on time and effort. Quality of test cases is directly proportional to time and effort.

7.4.2 ADVANTAGES AND DISADVANTAGES OF BOUNDARY VALUE ANALYSIS AS BBTS

Boundary Value Analysis (BVA) is practiced all five companies, in which authors conducted interview studies. The results related to advantages and disadvantages of the BVA are presented in section 6.4.1 and analysis and discussion on BVA advantages and disadvantages are given as follows:

- BVA is systematic way of evaluating the quality and completeness of program under test. It has symmetry and mechanical nature to make test procedure easier to remember and apply in practice.

- It is a good way of learning other activities like equivalence partitioning. It can focus on values above than boundaries and a good way of exception handling.

- It has quality of sophistication and simplicity. Test cases generated by it can cover discover errors at and beyond the boundaries.

- BVA can be done in uniform manner by maintaining one of variables at their nominal or average level and allows remaining variables to take on its extreme values.

- The reliability theory like baseline or robust and fault models like single fault or multi fault determines the number of test cases.

The disadvantages are:

- The test case generation using boundary value analysis introduces redundancy. Due to its simplicity it can be misused and do not use it with full potential. The effectiveness of it depends on its correct use.

- The variable dependencies needs for foresight into the system's functionality, in this case BVA can be restrictive. BVA is expensive practically.

7.4.3 ADVANTAGES AND DISADVANTAGES OF DECISION TABLE AS BBTS

The decision table based testing is practiced in three out of five companies in which authors conducted interview studies. The results of questions related to advantages and disadvantages are given in section 6.5.1. The analysis and discussion on advantages and disadvantages of is as follows:

- Decision table based testing discovers other directions that other testing strategies avoid. BVA does not discuss the data and logical dependencies but decision table do.

- The program logic, business rules, testing paths are exercised by using it. The potential and critical errors are discovered.

- It is more sophisticated technique as compared to BVA and EP. The test cases have potential to discover critical paths, large number of error are discovered.

The disadvantages are:

- It requires domain knowledge of program under test, any confusion in understanding of program understating decreases the chance of discovering a lot of errors.

- The quality of good test cases depend on investment of time and effort that's why it require an effort and time. It is not necessary that test cases generated by it are all useful, for removing un-useful understanding of functionality is required.

7.4.4 ADVANTAGES AND DISADVANTAGES OF CLASSIFICATION TREE AS BBTS

The classification tree method (CTM) is not practiced in all five companies. Authors make discussions to know that why they do not prefer these techniques, the reasons authors find out as a result of interview studies is present in section 6.6. In discussion advantages and disadvantages of this technique discussed. The advantages and disadvantages presented in literature and discussed with companies, authors identified following advantages and disadvantages of this technique:

- Classification tree method has well defined steps and deliverables. The steps and deliverables have been discussed in section 4.5.

- It has less reliance on automated tools and there is no restriction on type of specification and it also has less reliance on the experience of large software development.

- The test cases generated by classification tree can be calculated easily which helps to tester to estimate the required testing resources and other planning activities.

- It is an early test case generation strategy and helps to identify any problems before coding starts. It covers the business, functional and component testing as discussed in section 4.5.

- The test suite generated by using this method is independent of the program logic and therefore it can be reused for different implementations of the specifications.

But authors perceived following disadvantages:

- The test cases derived from CTM may be infeasible such as test cases not suitable for path coverage and if so, they will have to be removed.

- If the specifications in case of large projects has not been decomposed into smaller units prior to tree construction then resulting classification tree may be complicated.

- If CTM is used by different people then resulting classification tree, and hence the test suite, may be different.

7.4.5 ADVANTAGES AND DISADVANTAGES OF USE CASE BASED TESTING AS BBTS

Use Case Based Testing (UCBT) is practiced in four companies. The advantages and disadvantages as a result of interview studies have been discussed in section 6.7.1. Authors did analysis and discussed various pros and cons of it from various angles which are given as follows:

- The design methodologies like object oriented analysis and design place it in inherent parts. It provides a ground for both software developments as well as for testing provides uniform notations.

- Test cases development using UCBT enables us an early validation of requirements and it reveals most bugs in the application and provides better test coverage.

- Author observed that it is beneficial to uncover defects by using it when process flow during real world use of the system. It is easy to understand and language independent. It covers the complete picture of the software functionality.

Besides its advantages, disadvantages are:

- Authors analyzed that coverage of defects classes such as completeness defects, Input/output defects, Calculation defects, Data handling defects, Control flow and sequencing defects, concurrency defects, GUI defects and Non function requirement defects is not easy task. Along with it, various other strategies need to be use such as state diagram for complete path coverage, to uncover calculation defects extended use case notation need to be use etc.

- For component testing, it is not helpful and not uses to specify quality in terms of defect finding. It does not contain information regarding user interface. High quality test cases demands time and effort.

7.4.6 ADVANTAGES AND DISADVANTAGES OF STATE DIAGRAM BASED TESTING AS BBTS

State diagram based testing (SDBT) is practiced in two out of five companies. The pros and cons collected as a result of interview study are given in section 6.8.1. Authors discuss and analyze these factors. The advantages and disadvantages are given as follows:

- State diagram based testing is most frequently used model based testing. Modelling is an economic way of capturing knowledge about a system and then reusing this knowledge as the system evolves. Models provide way for structural analysis test plans are developed.

- State diagram has incremental nature. If requirement changes then it is easy to incorporate at further stages of software process.

- State diagram based testing has ability to uncover defects like: incorrect results, change have no effect, incorrect or missing events, sneak path etc.

- Test cases can be designed for all paths, all events or all actions or all transitions and finally n-transitions sequences.

The disadvantages are:

- Exhaustive testing by using it is not possible and impractical.

- Automated tool can be helpful for test case design, test case execution, monitoring and making a benchmark for validation of implementations.

- Model creation is time taking and effort hungry task.

7.5 DATA ANALYSIS WITH RQ4

How complementary are various combinations of black box testing techniques? Is there any best possible combination?

The black box testing techniques for functional testing point of view are complement to each other. Authors raised this question during interview, tried to get information from various angles like is combination is best to reduce test cases, produce quality of test cases, defect guessing power of test cases increased or not, what satisfactory level of functional testing is achieved etc.

- Authors analyzed that for functional testing, Boundary value analysis (BVA), equivalence partitioning, and decision table are important and complement to each other. If any one of these techniques is missing then functional testing satisfactory level cannot be achieved. Decision table is evolutionary form of equivalence class testing. Decision table groups the input and output behaviours into an ‘equivalence’ rule and test logical dependencies of these rules. Boundary value analysis and Equivalence partitioning are used generate test data in use case testing which enhance its power to find more defects and make it most defect guessing technique.

- Authors examined that State based testing is used to model the functionality, requirements documented in form of use cases, scenarios etc and it shows behaviour of the system at particular state, for specific set of inputs. Classification tree is like equivalence partitioning, it contains classes. To generate data for each disjoint class boundary value technique can be used. It is way of visualization of equivalence classes, their combination to generate valid set of test cases. From application, sophistication and effort point of view all techniques are different.

- Authors observed that Boundary Value Analysis does not discuss logical or data dependencies, it is domain based testing, less sophisticated, generated numbers of test cases are high and minimum effort is required. Equivalence partitioning discusses the data dependencies, generates less test cases as compared to boundary value analysis, it is more sophisticated than BVA and demands effort more than required to BVA.

- Authors examined that Decision table based testing discusses the logical dependencies, it is more sophisticated than aforementioned testing techniques and more effort is required to generate quality of test cases. Use case testing discusses the data dependencies by incorporating equivalence partitioning in test case data generation.

- Authors analyzed that Classification tree discusses the data dependencies by visualization of test classes and generating test cases for valid combinations. It is sophisticated and demands effort to identify disjoint classes, to visualize, identification of valid combinations and generating quality of test cases. State diagram based testing discusses the behavioural dependencies between different states of system, generates minimum number of test cases and demands effort to convert requirements into different states, transitions.

8 DATA VALIDITY

The validity of data collected from quantitative or qualitative research is vital, Harison [70]. The work in this thesis is completed with the help of qualitative research approach along with quantitative strategies. The validity of qualitative research is seen as strength to suggest whether the findings are accurate from participant's point of view, Creswell [21]. The criteria to validate interview results suggested by Trochim [70] are used in this report. The criteria consists on four different validity assessments to judge qualitative research approaches. The validity assessment approaches are as follows:

8.1 Credibility

8.2 Transferability

8.3 Dependability

8.4 Confirm ability

8.1 CREDIBILITY

The interview results of this thesis report should be believable or credible from the perspective of the participants involved in this qualitative research, William et al [69]. A multi-phased research approach including qualitative research along with quantitative strategies has been followed to achieve credibility of this thesis report. The first phase started from literature survey to find out black box testing techniques for functional testing presented in academia. To get information about testing strategies used for functional testing in software organization, a questionnaire is prepared.

Five software companies of Pakistan related to finance industry are selected to conduct interviews regarding black box testing techniques for functional testing. For conducting interviews legitimate participants from selected companies are contacted. Before conducting interview meetings, a proper correspondence through email taken place for deciding interview meetings with all the subjects on their available time. To get credible information, interviews through video conferencing (e.g. Involvement in on going testing process for clarification, on board elaboration of testing process etc) and telephone (e.g. Two way discussion to show the understanding and perception about topic under discussion) are conducted to get believable information. Authors are confident about the credibility of the study, after pursuing this validation process.

8.2 TRANSFERABILITY

To generalize the results and make concrete conclusions, transferability is concerned e.g. in this thesis report, the results of interviews are very helpful to identify the appropriate black box testing techniques for functional testing practiced in software industry related to finance domain. This thesis report can be very helpful for software industry to improve their functional testing process by giving information about suitable black box testing techniques according to their project needs. The interview results, the issues identified in this thesis report and suitable combination of black box testing techniques is a step toward removing these issues.

The interview subjects involved in the interview process can be a threat. All persons are working on different domain, having different working experience, my have an effect on the results. E.g. Company A is specialized in accounting domain, company B is specialized in transaction handling domain and company C has name in payment card applications. So, all the contacted individuals from their respective organization may have their own background, experience and knowledge, which can be different. But, subjects having same level of criteria (as discussed in section 5.3) are used for minimizing this threat. Other possible threat can be educational system of subjects. It is possible that their educational background is different, which may have an effect on the findings of the study. The natural settings of environment help to generalize the finding of the study by conducting interviews through video conferencing and telephone.

8.3 DEPENDABILITY

Emphasize of dependability is: “the ever-changing context within which research occurs”, Jacobson [43]. The authors contacted with participants by telephone and email for taking time for the interviews. Some of the people were busy with their release deadlines and they gave time to authors for interview after fifteen days and sometimes after one month. Before interview, authors send a general overview of the topic for which interview was going to be held. The questionnaire was not sent by the authors before interview. Authors conducted telephonic interviews and sometimes use video conferencing for understanding the phenomenon.

During the process of interview, authors realized that participants did not know the academic names and terminologies of the activities and processes but they are working with those activities and processes. The questioning process was so flexible that subjects describe their whole processes and activities from which required information can be extracted. To find out the exact and relevant answers, authors conducted more than one interviews in the same organization.

Authors tried to select high profile and high CMM (Capability Maturity Model) level companies to generalize the results. One threat is that all companies are not of same level and high profile. Some of them are market project driven organizations and others are providing solutions to Banking sectors or textile industry. The other threat is that different companies are using different black box testing techniques for functional testing, some are using two techniques and some are using combination of these techniques. The finding of the study may be affected by difference in number of companies (e.g. the most frequently used testing technique is Decision table instead of BVA). There is no validity threat regarding companies, if all of the companies are using same black box testing techniques for functional testing.

8.4 CONFIRM-ABILITY

A number of strategies exist to enhance the confirm-ability of this thesis report. E.g. semi-structured interviews are conducted in which open-ended questions asked for getting required information. The interview process is divided in a way that one person asked questions from the subject and the other person was dedicated to document the answers of those questions. Author discussed the confusions in the answers after the interviews. The misunderstanding changed to understanding through email to validate the answers.

To stay focus on the topic of black box testing techniques for functional testing and extract necessary information, authors have refined the interview questionnaire. Initially interview results were documented in the form of notes. On the very next day, after conducting the interview, authors coded the answers into Microsoft excel sheets to keep the data safe. Authors conducted telephonic interviews which have been recorded. The data is validated repeatedly hearing those recorded interviews.

9 EPILOGUE

9.1 CONCLUSIONS

This thesis comprise on an interview study of analysis and comparison of black box testing techniques practiced in five companies of Pakistan which providing financial solutions.

After this study, six important black box testing techniques have been identified that are available in academia and given as follows: Boundary Value Analysis, Equivalence Partitioning, Decision Table, Use Case Based Testing, State Diagram Based Testing and Classification tree based method. The data and information of interview related to these techniques is presented in form of tables.

The interview study identify black box testing techniques which are practiced in software companies under study e.g. Boundary Value Analysis, Use Case Based Testing etc. The advantages and disadvantages of each technique are discussed and analyzed in this report. The comparison of these techniques is also part of the report. Boundary Value Analysis and Equivalence Partitioning are comparatively catching more defects. State based testing is expensive one as compared to other techniques. Classification tree is time consuming. Decision table require sufficient knowledge of code to develop more effective defect catching test cases.

Boundary Value Analysis (BVA) and Equivalence Partitioning (EP) are most widely practiced techniques in subjected companies whereas use case based testing, decision table and state based testing techniques are at second, third and fourth level respectively. Classification tree method is not used by any subjective company. This study discusses, analyzes and compares the black box testing techniques used in companies of Pakistan. All subjective companies are using BVA and EP as most defect finding testing techniques.

This study provides an overview of important black box testing techniques, discusses problems associated with them along with their advantages. The study analyzes and compares different black box testing techniques on based of most defect finding, less time consuming, moderate effort and cost. These techniques are differentiated on the basis of aforementioned criteria.

9.2 FUTURE WORK

This thesis focuses specifically on black box testing techniques (BBTT) for functional testing. In this area further research may be more beneficial. The areas in which future work for BBTT can be done are given:

This thesis consists on the study of five companies of Pakistan which are providing financial solutions. It might be beneficial and interesting for further research in the area of BBTT, to conduct interview studies in the same area from other geographical locations with extended number of companies. The interview study focus on senior personnel working in this area like: project managers, quality assurance managers, software engineers of more than two year experience. But it might be more beneficial to include stakeholders like developers, testers etc.

REFERENCES

[1] Turing. A. Checking a Large Routine. Report of a Conference on High Speed Automatic Calculating-Machines (Jan. 1950). 67-69.

[2] Turing. A. Computing Machinery and Intelligence. Mind 59, (Oct.1950). 433-460.

[3] Baker, C. Review of D.D. McCracken's "Digital Computer Programming". Mathematical Tables and Other Aids to Computation 1 I, 60 (Oct. 1957). 298-305.

[4] Meyers, G.J. The Art of Software Testing. John Wiley & Sons, New York, 1979.

[5] Deutsch. M.S. Soffuare Verification and Validation: Realistic Project Approaches. Prentice-Hall, Englewood Cliffs, N.J... 1982.

[6] Miller. E. and Howden. W.E. Eds. Tutorial: Software Testing and Validation Techniques. IEEE Computer Society Press. New York. 1981.

[7] Deutsch, M.S. Software Project Verification and Validation. Computer 14. 4 (Apr. 1981], 54-70.

[8] Howden. W.E. Life-Cycle Software Validation. Computer 25. 2 (Feb.1982), 71-78.

[9] Guideline for Lifecycle Validation, Verification, and Testing of Computer Software. National Bureau of Standards Report NBS FIPS 101. Washington, D.C... 1983.

[10] Laurie Williams, Testing Overview and Black Box Testing Techniques, 2006, Available on internet: http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf, Last Visited 20th October 2008

[11] Huo Yan Chen , T. H. Tse, F. T. Chan, T. Y. Chen, "In black and white: an integrated approach to class-level testing of object-oriented programs", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Volume 7 , Issue 3 , July 1998, pp 250 -295 .

[12] T.Ryber, "Essential Software Test Design", *Fearless consulting*, 2007

[13] Fevzi Belli, Axel Hollmann, "Test generation and minimization with "basic" state charts", *Proceedings of the 2008 ACM symposium on Applied computing*, ACM, 2008.

[14] IEEE standard 610.12-1990. "IEEE Standard glossary of software engineering terminology." The Institute of Electrical and Electronics Engineers, 345 East 47: TH Street, New Your, NY 10017, USA, September 1990.

[15] ISO/IEC FDIS 9126-1. Information technology- software product quality, 2000. ISO/IEC FDIS 9126-1: 2000 (E).

[16] Robson, C. Real world research, second ed. Blackwell publishing, 2002

[17] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., And Wesslen, A. Experimentation in software engineering, first ed. The Kluwer international series in software engineering. Kluwer Academic Publishers, 2000.

[18] Adrion, R. Research methodology in software engineering. In First

[19] Pfleeger, S. L. and Kitchenham, B. A. 2001. Principles of survey research: part 1: turning lemons into lemonade. *SIGSOFT Softw. Eng. Notes* 26, 6 (Nov. 2001), 16-18.

[20] Juristo, N., And Moreno, A. M. Basics of Software Engineering Experimentation, second ed. Kluwer Academic Publishers, 2001.

[21] Creswell, J. *Research Design Qualitative, Quantitative* and Mixed Method Approaches. Sage Publications Ltd.2002.

[22] Owen, S., Brereton, P., and Budgen, D. 2006. Protocol analysis: a neglected practice. *Commun. ACM* 49, 2 (Feb. 2006), 117-122.

[23] Boehm, B.,Software Engineering Economics, Prentice-Hall, 1981.

[24] Miller, E., "The Philosophy of Testing," in Program Testing Techniques,IEEEComputer Society Press, 1977, pp. 1–3.

[25] Wallace, D.R. and R.U. Fujii, "Software Verification and Validation: AnOverview," IEEE Software, May 1989, pp. 10–17.

[26] Deutsch, M., "Verification and Validation," in Software Engineering (R. Jensenand C. Tonies, eds.), Prentice-Hall, 1979, pp. 329–408.

[27] Myers, G.,The Art of Software Testing,Wiley, 1979.

[28] Davis, A., 201 Principles of Software Development,McGraw-Hill, 1995.

[29] Sommerville, I., Software Engineering, 5th ed., Addison-Wesley, 1996.

[30] Boris Beizer, Software Testing Techniques, Van Nostrand Reinhold, 2nd edition, 1990.

[31] Pressman, Roger S, Software engineering: a practitioner's approach, 5th ed., McGraw-Hill series in computer science. ISBN: 0-07-365578-31. pp. 631-652

[32] Jorgensen, Paul C. *Software Testing: A Craftman's Approach*. 2nd ed. CRC Press, 2002.

[33] Luca Cardelli, Typeful Programming, Digital Equipment Corporation, Systems Research Center 130 Lytton Avenue, Palo Alto, CA 94301 [Online]. Available: ftp://gatekeeper.research.compaq.com/pub/DEC/SRC/research-reports/SRC-045.pdf

[34] Beyer, William H. *CRC Standard Mathematical Tables*. 25th ed. CRC Press, 1981.

[35] Naryan C Debnath, Mark Burgin, Haesun K. Lee, Eric Thiemann, A Testing and analysis tool for Certain 3-Variable functions, Winona State University.

[36] Karl Reed, Software Reliability, Testing and Security Class Lecture Notes. CSE31STM, subject of the Department of Computer Science and Computer Engineering, La Trobe University, Australia, 1998.

[37] D. Hamlet, R Taylor. Partition Testing Does Not Inspire Confidence. IEEE Transactions on Software Engineering, vol.16, no. 12, December 1990, pp. 1402 – 1411

[38] S. Toida. (2009, February). Equivalence Relation. Old Dominion University, Norfolk, VA [Online]. Available: http://www.cs.odu.edu/~toida/nerzic/content/relation/eq_relation/eq_relation.html

[39] E. van Veenendaal (2004), "The Testing Practitioner – 2nd edition", UTN Publishing, ISBN 90-72194-65-9.

[40] Mosley, Daniel J., "The Handbook of MIS Application Software Testing", Yourdon Press, Prentice Hall. 1993.

[41] Jean-Claude Vauthier, "Decision tables: A testing technique using IBM Rational Functional Tester", Software Services, IBM, http://www.ibm.com/developerworks/rational/library/jun06/vauthier

[42] Dr. Roggenbach, Prof. Schlingloff, " A Review Paper on Decision Table-Based Testing", http://www.cs.swan.ac.uk/~csmarkus/CS339/dissertations/FerridayC.pdf

[43] Ivar Jacobson. *Object-Oriented Software Engineering*. Addison Wesley Professional. ISBN 0-201-54435, 1992.

[44] Peter Zielczynski, Traceability from Use Cases to Test Cases, IBM [Online]. Available: http://www.ibm.com/developerworks/rational/library/04/r-3217/

[45] D. Wood and J. Reis, "Use Case Derived Test Cases", [Online]. Available: http://www.stickyminds.com/.

[46] Ivar Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison- Wesley, 1994.

[47] Martin Fowler, *UML Distilled*, Addison-Wesley, 1997.

[48] M. Jarke, "Requirements Tracing", *Communications of the ACM* , vol 41, no. 12,pp. 32-36, 1998.

[49] Y. Wu and J. Offutt, "Modeling and Testing Web-based Applications," GMU ISE Technical ISE- TR- 02-08, 2002.

[50] X. Jia and H. Liu, "Rigorous and Automatic Testing of Web Applications," *Proceedings of the 6th IASTED International Conference on Software Engineering and Applications (SEA 2002),* November, 2002.

[51] Illes,T., Paech, B., 2006, in IFIP International Federation for Information Procesing, Volume 227, Software Engineering Techniques: Design for Quality, ed. K, Sacha, (Boston Springer), pp. 211-222.

[52] Beizer, B.: Bug Taxonomy and Statistics, Appendix, Software Testing Techniques, Second Edition, Van Nostrand Reinhold, New York, (1990).

[53] Tsun S. Chow. Testing design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, 4(3): 178-187, May 1978.

[54] G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5): 1045-1079, 1955.

[55] E. F. Moore. Gedanken experiments on sequential machines. *Automata Studies*, pp. 129-153, Princeton University Press, NJ, USA.

[56] Steven Rosaria and Harry Robinson. Applying models in your testing process. *Information and Software Technology*, 42(12): 815-824, September 2000.

[57] Chang Liu and Debra J. Richardson. Using application states in software testing (poster session). *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*, p. 776, ACM, Cambridge, MA, USA, 2000.

[58] Pascale Thevenod-Fosse and Helene Waeselynck. An investigation of statistical software techniques. Journal of Software Testing, Verification & Reliability, 1(2): 5-25, July/September 1991.

[59] Hyoung Seok Hong, Young Gon Kim, Sung Deok Cha, Doo Hwan Bae and Hasan Ural. A Test Sequence Selection Method for Statecharts.*The Journal of Software Testing, Verification & Reliability,* 10(4): 203-227, December 2000.

[60] J. G. Kemeny and J. L. Snell. *Finite Markov chains*. Springer-Verlag, New York 1976.

[61] Dorothy Graham, Erik Van Veenendaal, Isabel Evans, Rex Black. *Foundations of Software Testing: ISTQB Certification,* 978-1-84480-989-9, page 102.

[62] M. Grochtmann and K. Grimm, "Classification tree for partition testing", *software Testing, Verification and Reliability*, 3:63-82, 1993.

[63] H. Singh, M. Conrad and S. Sadeghipour, "Test data design based on Z and the C1assification Tree Method", *in Proceedings 0f First 1nternational Conference on*

Formal Engineering Methods, November 1997, page5 81-90, and 1EEE Computer soc1ety Press.

[64] Y. Chen and P. L. Poon, “Construction of c1assification trees via the c1assification-hierarchy table”, *1nformation and software Technology*, 39(13): 889-896, December 1997.

[65] Y. Chen, P. L. Poon and T. H. Tse, "An integrated classification-tree methodology for test case generation", International *Journal 0f software Engineering and Knowledge Engineering,* 10(6):647-679, December 2000.

[66] T.Y. Chen and P.-L. Poon, “Experience with teaching black-box testing in a computer science / software engineering curriculum”, *IEEE Transactions on Education*, vol. 47, no. 1, pp. 42−50, 2004.

[67] Tira Cohene and Steve Easterbrook, “Contextual Risk Analysis for Interview Design”, Requirements Engineering 2005, Proceedings 13th IEEE International Conference

[68] Semi-Structured Interviewing, “Participatory Planning Monitoring &Evaluation Resource Portal”, WAGENINGEN UR 2004 – 2006, January 25th, 2008, 02:10 AM (http://portals.wi.wur.nl/ppme/?page=1124)

[69] William M.K. Trochim, “Qualitative Validity”, Research Methods Knowledge Base, 2006, August 14, 2008, 03:50AM, (http://www.socialresearchmethods.net/kb/qualval.php).

[70] L. Harison, “Political Research an Introduction”, British Library Publication, ISBN: 0-415-22655-4.

[71] www.istqb.org

[72] ISTQB course plan version 0.2.

[73] Denzin, N. K., & Lincoln, Y. S. (2000). “*Handbook of qualitative research*” London: Sage Publications.

APPENDIX 1

SEMI-STRUCTURED INTERVIEW QUESTIONNAIRE

1. What is your name?
2. What is your designation in the organization?
3. What is your responsibility in the organization?
4. At which CMM or CMMI level of your organization?
5. Do you employ black box testing strategies?
6. Which black box testing strategies you perform for your projects?
   - **a. Equivalence Partitioning**
     - i. Do you use equivalence partitioning as black box testing strategy for your projects?
     - ii. Why this technique is important for your organization as black box testing strategy?
     - iii. What is primary purpose of this technique?
     - iv. Can you provide two advantages and two disadvantages of equivalence partitioning as black box testing strategy?
     - v. Is this technique is complementary with combination to other technique?
   - **b. Boundary Value Analysis**
     - i. Do you use boundary value analysis as black box testing strategy for your projects?
     - ii. Why this technique is important for your organization as black box testing strategy?
     - iii. What is primary purpose of this technique?
     - iv. Can you provide two advantages and two disadvantages of boundary value analysis as black box testing strategy?
     - v. Is this technique is complementary with combination to other technique?
   - **c. Decision Table based testing**
     - i. Do you use decision tables as black box testing strategy for your projects?

ii. Why this technique is important for your organization as black box testing strategy?
iii. What is primary purpose of this technique?
iv. Can you provide two advantages and two disadvantages of decision table as black box testing strategy?
v. Is this technique is complementary with combination to other technique?

d. Use Cases based testing

i. Do you employ use cases as black box testing strategy for your projects?
ii. Why this technique is important for your organization as black box testing strategy?
iii. What is primary purpose of this technique?
iv. Can you provide two advantages and two disadvantages of use cases as black box testing strategy?
v. Is this technique is complementary with combination to other technique?

e. Classification Tree Method

i. Do you use classification tree method as black box testing strategy for your projects?
ii. Why this technique is important for your organization as black box testing strategy?
iii. What is primary purpose of this technique?
iv. Can you provide two advantages and two disadvantages of classification tree method as black box testing strategy?
v. Is this technique is complementary with combination to other technique?

f. State Diagrams based testing

i. Do you use state diagrams as black box testing strategy for your projects?
ii. Why this technique is important for your organization as black box testing strategy?
iii. What is primary purpose of this technique?
iv. Can you provide two advantages and two disadvantages of state diagrams as black box testing strategy?
v. Is this technique is complementary with combination to other technique?