

UML-BASED FUNCTIONAL SYSTEM TESTING

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERHAD SARICA

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2011

Approval of the thesis:

**UML-BASED FUNCTIONAL SYSTEM TESTING**

Submitted by **SERHAD SARICA** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan ÖZGEN

Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet ERKMEN

Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Semih BİLGEN

Supervisor, **Electrical and Electronics Engineering Dept., METU**

**Examining Committee Members:**

Assoc. Prof. Dr. Cüneyt BAZLAMAÇCI

Electrical and Electronics Engineering Dept., METU

Prof. Dr. Semih BİLGEN

Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. İlkey ULUSOY

Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Ece Güran SCHMIDT

Electrical and Electronics Engineering Dept., METU

Gürhan Günce GÜRSEL, M.Sc.

Manager, ASELSAN Inc.

**Date:** 27/01/2011

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name : Serhad SARICA

Signature :

# **ABSTRACT**

## **UML-BASED FUNCTIONAL SYSTEM TESTING**

Sarıca, Serhad

M. Sc., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Semih Bilgen

January 2011, 106 pages

Effectiveness of system testing, in specific phases such as design verification, requirements validation, test planning and generation, system integration and system testing are considered. Software as well as hardware test issues are reviewed. Metrics related to system testing are specified. The current system testing processes in a large Turkish military electronic systems manufacturer are reviewed, specific problems are identified and UML-based behavioral testing is proposed as an improved process. The current process and the proposed process are compared in terms of test coverage, test effectiveness and test effort metrics.

Keywords: System testing, test effectiveness, test process.

# ÖZ

## UML TABANLI FONKSİYONEL SİSTEM TESTİ

Sarıca, Serhad

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Semih Bilgen

Ocak 2011, 106 sayfa

Sistem testlerinin etkinliği dizayn ve ister doğrulaması, test planlama ve test senaryolarının hazırlanması sistem entegrasyon ve sistem testleri safhalarında değerlendirilmelidir. Literatürde bulunan yazılım ve donanım sistem testlerine yönelik kaynaklar taranarak, yazılım sistem testlerine yönelik UML tabanlı test senaryoları hazırlama metotları incelenmiş, donanım bazlı sistemlere adapte edilmeye çalışılmış ve bu çalışma sonucunda UML tabanlı fonksiyonel sistem test prosedürü ortaya çıkmıştır. Bir savunma sanayi şirketinde geliştirilip üretilen iki adet sisteme yönelik sistem testlerinde gözlemlenen problemler ortaya konmuş, geçmişe yönelik metrik hesapları yapılmıştır. İlgili sistemlere uygulanan UML tabanlı fonksiyonel sistem test prosedürü sonuçları, daha önceki test süreçleriyle, test kapsamı, test etkinliği ve test eforu metrikleri göz önünde bulundurularak karşılaştırılmıştır.

Anahtar sözcükler: Sistem sına, test etkililiği, sına süreci.

To My Family

## **ACKNOWLEDGEMENTS**

I would like to express my deepest gratitude to my supervisor Prof. Dr. Semih BİLGEN for his encouragement, guidance, advice and support throughout this study.

I would like to thank to my colleagues and friends for their continuous encouragement and valuable advices during this thesis. I am also grateful to ASELSAN Inc. for the facilities provided for the completion of this thesis.

Finally, I would like to thank to my family, Gizem MESTAV and Ahmed CAMCI for their encouragement and helps throughout my study.

# TABLE OF CONTENTS

ABSTRACT .....	iv
ÖZ .....	v
ACKNOWLEDGEMENTS .....	vii
TABLE OF CONTENTS .....	viii
LIST OF TABLES .....	x
LIST OF FIGURES.....	xii
LIST OF ACRONYMS AND ABBREVIATIONS.....	xiii
CHAPTERS	
1. INTRODUCTION.....	1
2. LITERATURE REVIEW.....	6
2.1. SYSTEM TESTING AND SYSTEM ACCEPTANCE TESTING.....	6
2.2. CONSTRUCTING TEST CASES.....	11
2.3. TESTING EFFECTIVENESS .....	18
3. UML-BASED BEHAVIORAL SYSTEM TESTING PROCEDURE .....	28
3.1. PRELIMINARY WORK BEFORE TEST CASE GENERATION ...	30
3.2. TEST CASE GENERATION PROCESS.....	31
4. SYSTEM TESTING PROBLEMS AND PREVIOUS SYSTEM TESTING RESULTS .....	40
4.1 .ACCEPTANCE TESTING/SYSTEM TESTING ACTIVITIES AND PROBLEMS.....	42
4.2. TEST PLANNING ACTIVITIES.....	50
4.3. ORIGINAL TEST PROCEDURES OF SYSTEMS.....	52
5. APPLICATION OF UML-BASED BEHAVIORAL SYSTEM TESTING PROCEDURE.....	56



5.1. INVESTIGATION OF ORIGINAL SYSTEM REQUIREMENTS DOCUMENT .....	57
5.2. REVIEW OF SYSTEM REQUIREMENTS DOCUMENT .....	59
5.3. IDENTIFICATION OF VERIFICATION METHODS .....	61
5.4. TEST CASE GENERATION PROCESS .....	63
5.5. TESTING OF SYSTEMS .....	82
5.6. TESTING RESULTS .....	85
5.7. COMPARISON OF PREVIOUS TESTING PROCESS AND UBST PROCESS .....	88
6. CONCLUSION .....	94
6.1. CONCLUSIONS .....	94
6.2. FUTURE WORK .....	100
REFERENCES .....	101
APPENDIX: USE CASES, BEHAVIORAL DIAGRAMS AND TEST SCENARIOS .....	106

## LIST OF TABLES

### TABLES

Table 2-1: System Testing and System Testing Approaches and Advises .....	11
Table 2-2: Test Case Generation Approaches and Their Applicability .....	16
Table 2-3: Test Complexity Measures .....	21
Table 2-4: Fault Finding Performances of Functional and Environmental Tests ..	22
Table 3-1: Example of RTM .....	31
Table 3-2: Activity Diagram Semantics .....	35
Table 3-3: IFD Semantics .....	36
Table 3-4: Example Test Plan Table .....	35
Table 4-1: Number of Requirements by Verification Status.....	53
Table 4-2: Number of Defects Found .....	53
Table 5-1 : Number of Requirements for Each Unit.....	57
Table 5-2: Number of Problematic Requirements .....	59
Table 5-3: Examples of Requirements Review Step.....	60
Table 5-4: Reviewed System Requirements .....	61
Table 5-5: Number of Requirements by Verification Methods .....	62
Table 5-6: V/UHF T/Rx SYS Opening Scenario Use Case (Textual).....	66
Table 5-7: GUARD Receiver Opening Scenario Use Case (Textual).....	68
Table 5-8: A Test Scenario for V/UHF T/Rx System Opening Scenario Use Case .....	80
Table 5-9: A Test Scenario for GUARD Receiver Opening Scenario Use Case...	81

Table 5-10: Requirements by Verification Methods.....	85
Table 5-11: Number of Defects Found by UBST and Prediction of Future Faults .....	87
Table 5-12: Number of Test Cases, Links and Requirements and Their Ratios....	88
Table 5-13: Previous Test Metrics vs. UBST Process Test Metrics .....	89

## LIST OF FIGURES

### FIGURES

Figure 2-1: Scenarios and requirements.....	13
Figure 5-2: GUARD Receiver Use Case Diagram.....	68
Figure 5-3: V/UHF T/Rx Opening Scenario Activity Diagram.....	72
Figure 5-4: GUARD Receiver Opening Scenario Activity Diagram.....	73
Figure 5-5: V/UHF T/Rx System Opening Scenario IFD.....	74
Figure 5-6: GUARD Receiver Opening Scenario IFD .....	75
Figure 5-7: IFG of V/UHF T/Rx System Opening Scenario Use Case .....	77
Figure 5-8: IFG of GUARD Receiver Opening Scenario Use Case .....	78

## **LIST OF ACRONYMS AND ABBREVIATIONS**

AT&E	: Acceptance Test and Evaluation
COTS	: Commercial Off-the-Shelf (Software or Systems)
Dz.K.K.lıgı	: Navy Forces Headquarters
FAT	: Factory Acceptance Tests
FSM	: Finite State Machine
HAT	: Harbor Acceptance Tests
HF	: High Frequency (frequencies between 3 MHz and 30 MHz)
IFD	: Interaction Flow Diagram
IFG	: Interaction Flow Graph
M&S	: Modeling and Simulation
OT&E	: Operational Test and Evaluation
PTT	: Push-to-Talk
RTM	: Requirements Trace Matrix
SAT	: Sea Acceptance Tests
SEDNC	: Systems Engineering Department of Naval Communications
SPL	: Software Product Lines
SRS	: System Requirements Specification

SSM	: Undersecretariat of Defense Industry
T&E	: Test and Evaluation
TCP	: Test Case Prioritization
TCR	: Test Case Reduction
TEMP	: Test and Evaluation Master Plan
T/Rx	: Transmit/Receive
UBST	: UML-Based Behavioral System Testing
UHF	: Ultra-High-Frequency (frequencies between 300 MHz and 3 GHz)
UML	: Unified Modeling Language
VHF	: Very-High-Frequency (frequencies between 30 MHz and 300 MHz)
V/UHF	: Very and Ultra-High-Frequency (frequencies between 30 MHz and 3 GHz)

# CHAPTER 1

## INTRODUCTION

System testing verifies that an integrated system meets the system requirements [1] whereas acceptance testing verifies that it meets user requirements [2]. Another definition is that system testing and acceptance testing can also be named as system verification and system validation respectively.

These two testing processes are important phases of the system development lifecycle. Acceptance testing is the first step of system delivery and a step that the developer gets paid and users get a new system which meets their needs if acceptance testing is fully or partially successful. System testing is not the last step for finding system faults, but it is the last step for developer to overcome the problems by paying lesser costs.

The effectiveness of system testing and system acceptance testing processes are directly related to fault finding capability of test scenarios and this capability has become significantly more important for the project lifecycle, considering prevention of late realization of faults [3] that it provides.

The first prerequisite for having effective system testing and system acceptance testing processes is having clear and understandable requirements [2][4][5][6]. Furthermore, the importance of communication between user and developer for solving the problem of ambiguous requirements [2] and for defining acceptance

criteria [5] should also be taken seriously during project lifecycle.

For hardware-based system testing, acceptance test and evaluation (AT&E) [7][8][9], operational test and evaluation (OT&E) [8][9][10] and scenario-driven testing [11] approaches are investigated. These are all useful information for conducting a whole system testing process from its planning to its execution, but the literature is relatively insufficient for defining methods for creation of test cases and scenarios for hardware-based systems when compared to software systems.

Lack of a standardized methodology and clearly defined process for creating test cases brings great risks into system testing and acceptance testing of hardware-based systems. Lack of a methodology obliges engineers to generate test cases for only the verification of the system requirements, an approach which can be summarized as “Create a test scenario to verify a requirement”. But, generating test cases simply to verify their intended requirements can be risky. In such an approach, no matter how much the engineer is qualified and experienced, there will be a high probability of missing some hidden faults that cannot be detected by only executing scenarios that verify the requirements.

The software systems and hardware-based “system of systems” such as communications systems of a naval ship are very similar when their project lifecycles are compared. Test case generation methods such as Finite State Machine (FSM) based methods [12][13] and Unified Modeling Language (UML) based methods [14][15][16][17][18][19] are important methods considering their methodological and systematic approaches to test case generation.

The most common hardware-based system testing method is behavioral/functional system testing. UML-based test case generation approaches for system testing processes cover the functional behavior of software systems. The test case generation method which is based on UML sequence diagrams [18][19] covers both the functional and structural testing sections. Besides, the method which is



based on UML use cases and activity diagrams [15] only covers the functional / behavioral testing section of system testing which includes only the verification of behavioral characteristics of the systems, mainly system responses to user actions.

In this study, first software system and acceptance testing approaches, hardware-based system testing approaches, test planning approaches and testing effectiveness issues are investigated. Then a selected methodological test case generation method for software system testing that is proposed in the literature is studied. This method is then improved and adopted to hardware-based systems by integrating some preliminary requirements management and test planning steps. Suitable systems from the products of department in which the author is currently employed are chosen. These are the Very and Ultra-High-Frequency (V/UHF) Transmit/Receive (T/Rx) System and GUARD Receiver, which have previous testing data for the calculation of testing metrics of test coverage, test effectiveness and test effort which are used for measuring testing effectiveness. Moreover, the problems related to previous system testing effort and system testing and acceptance testing processes in the company where author is employed are investigated. The most visible problems can be mentioned as misunderstanding of requirements, overlooked requirements and operational scenarios during testing process, as well as insufficient level of system testing and having no methodological approach for test case generation.

After identifying problems and gathering previous testing effort metric results, the improved testing procedure is applied to corresponding systems and testing scenarios are generated by forming UML models and gathering the test cases systematically from the UML models. The test scenarios which are generated by the developed procedure are executed by considering the rules which are formulated for increasing the testing effectiveness. The data gathered during testing are used for calculation of metrics. The metrics of test coverage, test effectiveness, test effort and calculations made for measuring test complexity are

evaluated. These measurements show that testing effectiveness is improved by the application of UML-Based Behavioral System Testing (UBST) Procedure.

Beyond this introductory chapter, this thesis document is organized as follows:

In Chapter 2, software system testing and hardware-based system testing approaches are investigated. The propositions of these approaches are analyzed and their applicability is discussed. Moreover, studies on test case generation methods which are developed for software system testing and test effectiveness issues are also reviewed. For the sake of brevity, concepts and approaches considered especially significant within the scope of the present study are presented and compared in a tabular form in different sections of this chapter.

In Chapter 3, the proposed system testing procedure UBST procedure is introduced. The chosen test case generation method and its advantages over the other methods are presented in detail. Test case generation with the chosen method is presented step by step. The additions which are made to this method for adapting the method to hardware-based systems for handling requirements and planning test sequence are also discussed.

In Chapter 4, information about the department of the company where author is employed and its products are presented. The chosen subsystem of communications system for the application of developed system testing procedure is described. Furthermore, system acceptance testing approach applied in the author's department is presented and problems faced in acceptance testing processes and system testing processes are determined by interviewing colleagues and superiors. Furthermore, the previous testing effort is presented in detail, giving its data and calculation of metric values for the comparison

Chapter 5 includes the information about the application of UBST procedure to V/UHF T/Rx System and GUARD Receiver. The application of the procedure includes the reviewing of system requirements specification (SRS) document of the systems under test, identification of verification methods of requirements and

generation of test cases and formation of test scenarios. In addition, the rules which are created for organizing the execution sequence of test cases and test scenarios are presented. At the end, test results of the applied procedure are gathered; metric calculations and calculations which are made to analyze the complexity of testing process are presented. The chapter concludes with the metric comparisons and analyses which show that testing effectiveness of system testing processes of V/UHF T/Rx System and GUARD Receiver are improved compared to the previous testing efforts.

Chapter 6 provides the summary and the concluding remarks of this study as well as presenting suggestions for future study.

## **CHAPTER 2**

### **LITERATURE REVIEW**

The acceptance and system testing phases are complex stages for both hardware and software system projects. The whole process should be examined by studying technical and planning sides of the testing process for successful verification and validation of the system under test. The related work on software system acceptance tests, software system tests, hardware-based system acceptance tests, system tests and system level test planning issues are reviewed in the following parts of this chapter.

#### **2.1. SYSTEM TESTING AND SYSTEM ACCEPTANCE TESTING**

In this section, first the definition of system acceptance testing and system testing for software systems, hardware-based systems and system of systems will be given from literature. Then, some approaches on system testing and system acceptance testing will be presented and discussed.

##### **2.1.1. Definition**

The object of acceptance testing is generally an integrated software system, and for the successful acceptance of the software system, it should comply with all user requirements and system specifications [2]. In the behavior-based acceptance

testing approach, the major objective of acceptance testing is defined as demonstrating how well the software system satisfies the customer's requirements [13]. Another definition made in [5] is that acceptance testing is a formal testing conducted to determine whether a system satisfies its acceptance criteria, the criteria that the system must satisfy to be accepted by the customer. In [20], the aim of software testing is stated as to minimize the cost of software failures and defects for the entire lifecycle of the software product. In another source, system testing is defined a testing process in which tester proves that the system meets all objectives and system requirements [1]. An interesting definition of system testing and acceptance testing, emphasizing their difference, is given in [21]. System testing verifies that the system meets all system specifications which are gathered from user requirements while system acceptance testing also verifies the correspondence between the system and user's expectations [21]. In [1], the difference of system testing and acceptance testing is also discussed such that system testing can be referred as verification testing, while acceptance testing can be renamed as validation testing. Verification testing can be described as requirements-based testing, while validation is referred as capability-based testing [22]. A commercial definition is made stating that acceptance testing is a significant stage in the contractual process, which is likely to operate as a payment milestone and it will affect the application of any warranty provisions available to customer [23].

For military project lifecycles, for complex systems, the concept of AT&E is mentioned in [7]. AT&E is an activity which is shared between project owner and project developer. AT&E focuses on mainly the system-level requirements (functional, performance, etc.) contained in such documents as system specification document and the contract. AT&E is designed to provide confirmation that the system meets the original user requirements and it is ready to be accepted by the customer for operational use [7][8][9].

Various definitions are mentioned above, but they can all be summarized as follows: system testing is the verification of system requirements by developer, whereas acceptance testing is the verification and validation of system requirements and user requirements (operational requirements) by user and developer. There are some methods and approaches that can be useful in system testing and acceptance testing which will be discussed in proceeding parts of this thesis work.

### **2.1.2. Understanding Requirements to Test Correctly**

One of the challenging problems of system testing is observed as the correlation between requirements and test cases. When the requirements are clear, easily understandable and sufficiently explain the system, the test cases can be mapped to each and every requirement that should be verified before system delivery.

In [2], efficient knowledge accumulation in acceptance test process is proposed. The user requirements should be clearly specified in user requirements documents and software should meet those requirements. But in practice, success rate is very low in determining user requirements accurately and completely. This situation ends with the lack of knowledge or misunderstanding of user requirements by developer and misunderstanding of characteristics of software developed by user until both developer and user meet at accepting test phase.

The acceptance criteria (or verification criteria) is based on the measure of whether a requirement is satisfied or not [5]. It defines what the system should achieve in order to meet the requirement [6]. Hence, it is proposed in [5] that acceptance criteria of system requirements should be measurable; it shouldn't be understood differently from person to person. Documenting acceptance criteria during the requirement analysis can help to confirm that the requirements are verifiable, and also help to reduce the ambiguity [6]. Moreover, the acceptance criteria must be defined and agreed upon the negotiations between developer and customer in the contract meetings, and finally an acceptance criteria document should be prepared as a part of contract where two sides are both agreed on.

Actually, both methods discussed in this section propose similar ways to ensure that there are no misunderstood requirements or acceptance criteria. Successful application of the approaches absolutely lead to the ability of making right decisions in the test design process and render the testing process more effective and efficient.

### **2.1.3. Operational Test and Evaluation**

Another testing category, namely Operational Testing, gains importance within the scope of AT&E. The aim of OT&E is operational functionality of the system [8][9][10]. The design issues are out of consideration in this type of testing activity [10]. The testing activities are conducted by customer or an organization which customer has an agreement with. Customer conducts the tests and decides if the system functionally meets their needs [7]. Tests are conducted to verify that system meets its specification when subjected to the actual operational environment [9].

#### *Modeling and Simulation*

There are some interesting approaches in OT&E, such as using Modeling & Simulation (M&S) in the test cases that cannot be tested operationally due to constraints, such as safety [24][25] and the scenario-driven approach in generation of test cases in system level testing activities [11].

The OT&E approach of M&S is defined as they are tools that can potentially augment or complement actual field tests and provide decision makers necessary information to assess the progress of a system toward fulfilling the operational needs [24].

M&S helps OT&E process in a way such that it gives important information about the cause of real world instances of test cases which results in early identification of critical issues, and gives a chance to make a more informed and efficient OT&E plan. Moreover, it provides early operational feedback to system designers and decreases the number of needed field tests [25].

#### **2.1.4. Scenario-Driven Approach**

According to [11] in which scenario-driven approach is discussed, system testing's primary objective is to evaluate the capability and dependability of a system rather than to detect uncovered bugs. The other objectives of system testing are determining the validity of the final system with respect to user needs and requirements, examining and evaluating the system behavior by executing a set of sample data and reducing the risk of unexpected outcomes. The benefits of scenario-driven approach are listed in [11] as follows:

- It enables an understanding of system's behavior through sequences of system operations.
- It allows identifying which scenarios result in the same system behavior. So that, it helps test case generation by giving an idea about test case coverage and it decreases cost of testing activities by preventing duplication of similar testing scenarios.

In [11], some design considerations are offered in creating test cases for functional behaviors, interoperability, integrity, availability and performance of systems. For functional testing of the systems, it is offered that test cases should be real use scenarios of the system as much as possible. The goal of generated test cases is to cause sequences of interactions and behaviors in the actual user environments in order to make the validation of the functional components if they are truly meaningful in terms of functional user requirements [11].

#### **2.1.5. Evaluation of the Approaches**

In this part of the thesis work, the approaches and advices about system testing and system acceptance testing and its wellness which are discussed above will be discussed in summary. The concepts of knowledge accumulation, agreed-on acceptance criteria, scenario-driven testing approach, OT&E and M&S and their applicability are presented in Table 2-1:



Table 2-1: System Testing and System Testing Approaches and Advises

	<b>System Testing</b>	<b>System Acceptance Testing</b>
<b>Verifies</b>	System Requirements	User requirements
<b>Aim</b>	Demonstrate and verify that system is working right according to system requirements.	Demonstrate and verify that developer develop the right system according to user requirements and contract.
<b>Knowledge accumulation [2]</b>	<ul style="list-style-type: none"> <li>Information exchange between developer and user, developers and systems in operation.</li> <li>The investigation of systems in operation.</li> </ul>	
<b>Having clear requirements [2][5][6]</b>	<ul style="list-style-type: none"> <li>Need for deciding acceptance criteria</li> <li>Need for generation of correct test scenarios</li> <li>Need for effective testing</li> </ul>	<ul style="list-style-type: none"> <li>Need for deciding acceptance criteria</li> <li>Prevents misunderstandings</li> <li>Prevents disagreements in acceptance (Developer-User)</li> <li>Need for generation of correct test scenarios</li> <li>Need for effective testing</li> </ul>
<b>Agreed-on acceptance criteria [5][6]</b>	<ul style="list-style-type: none"> <li>Prevents disagreements in acceptance testing process.</li> <li>Need for generation of necessary operational test scenarios.</li> <li>Increases the effectiveness of testing process indirectly.</li> </ul>	
<b>OT&amp;E [8][9][10]</b>	-	<ul style="list-style-type: none"> <li>Verifies operational functionality.</li> <li>Conducted by user in operational environment.</li> </ul>
<b>M&amp;S [24][25]</b>	<ul style="list-style-type: none"> <li>For operational tests which cannot be executed due to constraints. (Safety, etc.)</li> <li>Gives valuable information about real world instances of test cases.</li> <li>Have a development cost and time.</li> <li>Can be irrelevant for time-intensive projects.</li> </ul>	
<b>Scenario-driven approach [11]</b>	<ul style="list-style-type: none"> <li>System testing is not intended to find defects → is not meaningful.</li> <li>Advises real operational scenarios as test cases → increases the fault finding probability.</li> </ul>	

## 2.2. CONSTRUCTING TEST CASES

In this part of this thesis work, the test case generation methods, approaches and advises for software systems given in literature will be presented. FSM-based test case generation methods, UML-based test case generation methods based on behavioral diagrams of state machines, sequence diagrams and activity diagrams

and the place of software product lines (SPL) in test case generation issue will be discussed.

### **2.2.1. FSM-Based Test Case Generation Approaches**

An approach to software acceptance testing is behavior-based acceptance testing [13]. A formal scenario-based acceptance test model for testing the external behaviors of software systems is proposed.

The scope of the approach in [13] is to build an FSM which includes all scenarios for a selected test case. All scenarios that are possible to eventuate are covered in the test plan for corresponding test case. By covering all these scenarios, more test cases can be generated and fault detection can be more efficient, which makes system testing and acceptance testing more efficient.

There are some interesting methodological approaches to software testing such as Black-Box testing, behavioral testing or functional testing [12] which can be applied to systems. In [12] varying software testing techniques and test case generation techniques are proposed for software systems.

The method in [12] appears to be similar to the FSM method proposed in [13]. The unique specifications are named as nodes that should be verified with the generated test case scenarios [12]. Basically, by evaluating the necessary paths (i.e. scenarios) in the graph that consists of nodes (requirements), the evaluation of the system can be made. These paths and nodes are illustrated in Figure 2-1.

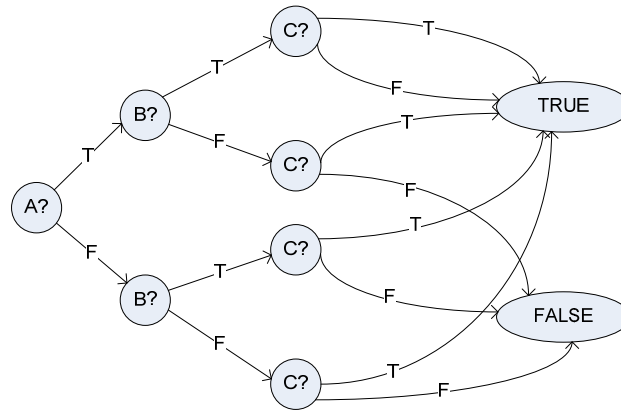


Figure 2-1: Scenarios and requirements

### 2.2.2. UML-based Test Case Generation Methods

A popular approach, use of UML models for test case generation is discussed in [14], [15], [16], [17], [18] and [19]. Most of the approaches of UML-based methods for test case generation are originally developed for automatic generation of test cases, but they also help to think in a way of applying these methods to hardware-based system test case generation and come up with a methodological approach.

The automation of tests requires some standardized models for the generation of test cases. Scenarios and use cases which are the elements of UML do not only feed requirements, but they also build the bases for testing [17]. The use case models are transferred to behavioral diagrams, these diagrams are refined according to method specifications and these refined diagrams are used to generate system-level test cases. These methods mainly generate test cases for black-box testing that is functional behavior of the system [14][17][18][19].

#### Test Case Generation by using State Machines

In [14], use cases are generated by considering their preconditions, post-conditions, extensions and variations. The scenarios that the use cases contain are transferred to state diagrams (or state machines). These state machines have

transitions specifying pre and post conditions and message transfers between states and test cases gathered from these state diagrams. In [17] a similar approach to system testing is advised. Again, use case diagrams and use cases are generated and these diagrams are converted to state diagrams. The state diagrams are converted to a defined diagram named as usage graph. A usage graph is a directed graph with a start node and an end node and usage states between them. Usage states are connected to each other with transitions which are actually user actions. Usage models are used in generating test cases by executing the transitions (user actions) between the usage states. Furthermore, in [17] white-box testing issues are discussed which are concentrated on structural behavior of software systems which is out of scope of this thesis work.

#### *Test Case Generation by using Activity Diagrams*

The approach described in [15] tests the system from the user's viewpoint. First, use cases are gathered from requirements, which are then going to be used to build activity diagrams. Then, activity diagrams are converted to interaction flow diagrams (IFD), which can be defined as an intermediate step to generate test plans. IFD, as it can be understood from its name, reduce the activity diagrams by subtracting intermediate steps (for example interaction between system modules) between user interaction steps. At the end, IFDs are converted to interaction flow graphs (IFG) which are based on a tree structure with no loops in order to have distinct scenarios for the corresponding use case which can be also defined as test cases. This test tree is executed based on a Depth-First-Search algorithm which ensures that each cycle is executed once.

#### *Test Case Generation by using Sequence Diagrams*

The approaches in [18] and [19] use UML sequence diagrams, which include the information of interaction of system with actors, in order to generate test cases. The approach defines a special diagram gathered from sequence diagram to clearly define the scenario paths and uses the diagram to cover the user-system

interactions. The approaches in [18] and [19] includes the black-box testing and white-box testing issues together, considering the structural behavior and functional behavior of the system under test and does not advice a method to reduce the whole testing issue to a functional system testing process.

#### *Test Case Generation by using Use Cases*

Another approach for test case generation from use cases is discussed in [16]. The approach is based on the rule for test cases that each scenario or instance of a use case should correspond to a test case and this approach brings the advantage of preventing the consequences of incomplete, incorrect and missing test cases as other approaches also provide.

The approach in [16] offers first building a system boundary diagram which depicts the interfaces between the software being tested and the individuals, systems and other interfaces; secondly for all actors defined in system boundary diagram, the use cases are generated. At the end, test cases are generated in a way that there exist at least two test cases for one use case which are successful execution of test case and unsuccessful execution of test case [16]. Clearly, much more test cases can be generated for a use case for exceptions and alternative courses.

Furthermore, there are also approaches for functional testing for SPL. An SPL is a set of software intensive systems sharing a common and managed set of features that satisfy the specific needs of a particular market segment or mission. A large number of studies have been done for SPL testing which are covering all the testing levels in the lifecycle of software and unit testing to functional testing. The most important ones of those works about this thesis work's scope are functional testing approaches. A lot of methods are discussed in [26], but mostly studies have been done using UML diagrams which have already been discussed above.

### 2.2.3. Evaluation of the Approaches

In this part of the thesis work, the approaches and methods about test case generation of software system testing process and their specifications will be compared. The approaches of FSM-based and UML-based test case generation methods using state machines, activity diagrams, sequence diagrams and use cases and their applicability to system testing of hardware-based systems or system of systems are presented and compared in Table 2-2:

Table 2-2: Test Case Generation Approaches and Their Applicability

Approach	Concentration & Benefits & Advantages	Disadvantages
<b>FSM-Based Behavior based approach [13]</b>	<ul style="list-style-type: none"> <li>Covers all possible scenarios.</li> <li>Increases fault finding probability.</li> </ul>	<ul style="list-style-type: none"> <li>Not a standardized method.</li> <li>Hard to apply preliminary steps before converting to FSM.</li> <li>Can be applied to systems that are produced in a production line.</li> </ul>
<b>FSM-Based Black-Box approach [12]</b>	<ul style="list-style-type: none"> <li>Requirements → Nodes</li> <li>Sequences of nodes verified.</li> <li>Prevents missing requirements.</li> </ul>	<ul style="list-style-type: none"> <li>Modeling, documentation and application will be time consuming.</li> <li>Becomes inefficient for system testing of hardware-based systems.</li> </ul>
<b>UML-Based using State Machines [14][17]</b>	<ul style="list-style-type: none"> <li>Use Cases → State Diagrams</li> <li>State Diagrams also contain pre and post-conditions and extensions</li> <li>State Diagrams → Usage Graphs</li> <li>Usage Graphs → Test Cases</li> <li>Models functional behavior</li> </ul>	<ul style="list-style-type: none"> <li>Not as efficient as the other behavioral diagrams in modeling sequences of actions and functional behavior.</li> <li>Complex transitions</li> <li>Manual application becomes time consuming.</li> </ul>
<b>UML-Based using Activity Diagrams [15]</b>	<ul style="list-style-type: none"> <li>Activity Diagrams → IFD</li> <li>IFD includes only user-system interactions.</li> <li>IFD → IFG (A directed tree)</li> <li>IFG → Test Cases</li> <li>Models functional behavior</li> <li>Easy to apply manually</li> <li>Includes all operational scenarios</li> </ul>	All actions should be handled manually for system testing of hardware-based systems

Table 2-2 Continued

Approach	Concentration & Benefits & Advantages	Disadvantages
<b>UML-Based using Sequence Diagrams [18][19]</b>	<ul style="list-style-type: none"> <li>• Sequence Diagrams and Use Case Diagrams → Intermediate Diagrams</li> <li>• Intermediate Diagrams → Test Cases</li> <li>• Covers Black-Box testing</li> </ul>	<ul style="list-style-type: none"> <li>• Many intermediate steps</li> <li>• Complex manual steps</li> <li>• Covers also white-box testing</li> <li>• Not only covers functional behavior but covers also structural behavior.</li> </ul>
<b>UML-Based using Use Cases [16]</b>	<ul style="list-style-type: none"> <li>• Use Cases → Test Cases</li> <li>• Use cases are defined on system interfaces</li> </ul>	<ul style="list-style-type: none"> <li>• Only covers main success scenario and failure scenario</li> <li>• Operational scenarios are not discussed</li> <li>• Not a standardized method considering other UML-based methods</li> </ul>

In addition to the information which is given in Table 2-2, the use of UML models [14][16][17][18][19] (including use cases) for some system testing and system acceptance testing activities' test case generation process can be not as efficient as it is in software testing issues because of limited test scenario number and limited input to the scenarios for system testing case. This low number of obvious scenarios can be obtained without the help of behavioral diagrams. But, having a systematic and defined way of system developing language, which is understandable and clear, will help the system developers to develop and specify systems in graphical models where there is no chance of misunderstanding.

Furthermore, SPL is a very different issue and methodology from the ones that will be used in the scope of this thesis work. The methods discussed in [26] are similar to the methods discussed in [14], [15], [16], [17], [18] and [19], but SPL testing is out of the scope of this thesis study.

## 2.3. TESTING EFFECTIVENESS

Until now, some testing techniques of software and hardware based systems presented in the literature are discussed. But, the point of how we can enhance the effectiveness of this whole testing issue is the topic of testing effectiveness assessment. This section discusses the problems that create ineffectiveness of testing processes, discusses some techniques used in order to assess testing effectiveness and presents some metrics to measure and compare the effectiveness of testing activities.

### 2.3.1. Testing Metrics

A metric is a measure [27]. Test metrics accomplish in analyzing the current level of maturity in testing and give a projection on how to go about testing activities by allowing us to see what is wrong and what makes testing process ineffective.

Test metrics that are determined to collect for a system or software should be measurable, easy to collect, simple and meaningful in order to be beneficial for improving the testing process.

There are several types of test metrics in software engineering, such as test coverage, test effectiveness, test effort, test span, test complexity, Mean-Time-To-Failure, defect density, customer problem metric, customer satisfaction metric and so on. In following sections, only the metrics that are suitable to scope of this thesis work will be discussed.

#### 2.3.1.1. Test Coverage

In [28], the first objective of test coverage metric is defined as that it is used to measure that how many of the requirements is tested. Another definition made in [27] such that given a set of things to be tested, test coverage is the portion that was actually tested and given following equation:

$$\text{Test Coverage} = \frac{\text{Tests Conducted}}{\text{Total Tests}} \times 100 \quad (2-1)$$



According to [28], to have a satisfactory test coverage value, it should be determined that each requirement is linked to at least one test case. Besides in [27], it is stated that it may be impossible to reach 100 % test coverage in system level test cases if “Total Tests” value in above equation is not limited by making a “Test Inventory”. Actually, “Test Inventory” is the total number of tests that should be executed to verify the system, which can be redefined as total number of test cases to verify all the requirements. So, the definition given in [28] can be substituted to the equation given above and the following equation for test coverage can be obtained:

$$Test\ Coverage = \frac{Requirements\ linked\ to\ at\ least\ one\ test\ case}{All\ requirements} \times 100 \quad (2-2)$$

### 2.3.1.2. Test Effectiveness

The definition of test effectiveness is made as the ability of a test to accurately emulate the expected mission environment, and reliably detect failure causing defects before launch in [29]. In [30], it is stated that high test effectiveness is achieved when more deficiencies of system found in early testing procedures such as unit testing, subsystem testing and, when only workmanship problems are occurred during acceptance testing. According to [3], test effectiveness refers to the ability of the test environment, at operational conditions or in factory, to cost effectively detect and isolate actual faults in the unit or system. Test effectiveness is defined simply as bug-finding ability of the test set in [27]. The following equation is given in [31] for test effectiveness measurement:

$$Test\ Effectiveness = \frac{Defects\ found\ by\ testing}{Defects\ found\ by\ testing + Defects\ reported\ by\ user} \times 100 \quad (2-3)$$

Evaluating Test Coverage and Test Effectiveness metrics together will give developer a great measure about the testing effort of system under test. As it is stated at [27], having 100 % of test coverage will not lead to a high percent of test effectiveness. Giving an example, a test set may have 50 % test coverage whereas

it has a 95 % test effectiveness measure. This means the test designer chooses the right 50 % of tests from test inventory and makes the whole testing process more efficient by reducing time and cost.

Test effectiveness with above equation only measures the percentage of the defects that are found. But, all found defects are not same if the solutions of the defects are considered. So, a more precise measurement can be done if a variable value for each defect found will assigned. This variable value can be between 1 and 10 and the developer can grade defects by multiplying them with this variable value. The above measurement method can be illustrated with the following equation:

$$Test\ Effectiveness = \frac{\sum_i^m k_i Defect_i}{\sum_i^m k_i Defect_i + \sum_{j=m+1}^n k_j Defect_j} \times 100 \quad (2-4)$$

Here k is the variable that is defined above. This measurement will give us the idea of test effectiveness by considering solution efforts of the defects.

### **2.3.1.3. Test Effort**

The aim of test effort is defined as minimizing the number of defects that the users find in the project [27]. This goal is achieved by eliminating defects and bugs before system delivery. So, the test effort metric can be described as the ratio of defects eliminated to all defects found and can be represented with the following equation:

$$Test\ Effort = \frac{Defects\ Eliminated}{All\ Defects\ Found} \times 100 \quad (2-5)$$

This metric is useful in projects which have multiple delivery of the system under test in different times. So, while getting the measurements from first system

delivery, the developer can improve the test procedures for upcoming deliveries by eliminating the defects that are found by user in first system delivery.

#### 2.3.1.4. Test Complexity

In large projects which have many more requirements than small projects, traceability of requirements to tests should be investigated for excessive and insufficient testing. In this metric, ratio of links to requirements (L/R), ratio of links to test cases (L/T) and ratio of requirements to test cases (R/T) are investigated where a link is a connection between a requirement and a test case [28]. The measures of L/R, L/T and R/T and the expressions of these measures are given in Table 2-3:

Table 2-3: Test Complexity Measures

Measure	Limit	Meaning
R/T	>1:1	If close to 1:1: <ul style="list-style-type: none"> <li>• Long testing process</li> <li>• Increases testing cost</li> </ul> If less than 1:1: <ul style="list-style-type: none"> <li>• Risk of over testing</li> </ul>
L/R	>1:1	If less than 1:1: <ul style="list-style-type: none"> <li>• Existence of requirements not linked to a test case</li> <li>• End up with missing verification steps</li> <li>• Decreases testing effectiveness</li> </ul>
L/T	-	If too high: <ul style="list-style-type: none"> <li>• Test process is too complex</li> <li>• Traceability becomes hard,</li> <li>• Probability of excessive testing of requirements</li> </ul>

#### 2.3.2. Improving Testing Effectiveness

The above metrics can be used in hardware systems to measure the suitability and efficiency of testing process. In following sections, the ways of improving testing effectiveness, efficiency and metrics introduced in previous sections will be discussed.

### 2.3.2.1. *Role of Functional and Environmental Tests in Testing Effectiveness*

In [27] and [29], some measures are given about the ratio of faults found during functional testing in ambient conditions and environmental tests. These ratios are given in terms of percentages in Table 2-4:

Table 2-4: Fault Finding Performances of Functional and Environmental Tests

Testing Step	Fault Finding Percentage
Early Integration Tests [27]	10-25 %
Functional Tests [27]	~ 40 %
Integration Tests at System Level [29]	15-25 %
Environmental Tests including Thermal and Dynamic Tests [29]	50-60 %
Dynamic Tests [29]	10-25 %
Thermal Tests [29]	20-35 %

Investigating Table 2-4, it is seen that functional and environmental tests are very important in improving testing effectiveness considering whole testing process of the system, from unit testing to system acceptance testing. Considering acceptance testing and system testing, system level functional tests can be also an effective step of whole testing process of systems. The importance of environmental tests cannot be underestimated, but the environmental tests are out of scope of this thesis. It is advised in [29] that for effective testing more functional tests at ambient conditions, which come with lower risk ratios and lower cost, should be conducted [29].

### 2.3.2.2. *Late Realization of Deficiencies: Cost Problem*

In [3], it is mentioned that the allover testing effectiveness comes to a cost problem finally. This is because of late detection of deficiencies, problems or failures in acceptance testing, system-level testing or operational testing levels

which bring high risks to system projects and incur cost due to delayed project finalization.

It is stated in [3] that the problem of testing effectiveness depends on late realization of deficiencies. That is the concern of engineers is the design problems that are occurred in acceptance testing level or operational testing level. But, what are of even greater concern are the faults which cannot be detected in either acceptance test, operational test or environmental test, and yet may cause serious loss of function [3]. The problem of hidden failures or deficiencies is hard to solve because of fragmented solutions that may work on paper or during limited, controlled demonstrations. But actually, these solutions only postpone the appearance of deficiencies to operational life of system [3]. For overcoming this problem, preventions should be taken into account to increase testing effectiveness of system level testing processes, which is the last step for detecting and fixing problems without higher costs.

### ***2.3.2.3. Enhancing Testing Effectiveness***

#### ***Decision Making Systems***

In order to achieve a mature test effectiveness decision system, test data, cost data, performance data and many more test related data of prior testing jobs of similar systems should be gathered and judged by the above given decisions. In [29], some industrial firms are investigated for testing effectiveness systems, but there were no such a full decision system about testing effectiveness. Some recommendations made for firms in achieving testing effectiveness such as:

- Develop a long range testing effectiveness strategy,
- Improve and integrate data gathered from previous systems,
- Investigate innovative test strategies,
- Conduct test substitution studies,

- Develop priority risk assessment to target testing more precisely against the most serious risks.

### Tracing Requirements

For assessment of testing effectiveness, some advices are given in [32]. In the beginning, requirements to be tested should be clearly identified; they should not be ambiguous and vague. After identification of requirements, a “Requirements Trace Matrix” (RTM) should be prepared that is used as a planning tool. This tool helps determining the number of tests required, verification and test types used, whether tests can be automated and if there exists any duplication of tests and reuse capabilities. The second step for assessment process is analyzing existing testing activities of similar software systems. The RTM of these existing systems can be helpful for analysis by giving a chance to investigate previously used test types associated by unique requirements, to investigate equivalence classes of tests. Equivalence classes of tests corresponds to test cases which may be testing the same thing, in which if one test passes all the others are treated as passed or the reverse case. The last step in the assessment process is given as removing the unnecessary and repeating test steps from test plan. In [32], it is stated that after going through these steps a more meaningful collection of test steps will be in hand and tests will have a higher probability of finding undiscovered defects. Again this results in an increased return on investment, so increased cost effectiveness.

### Reviewing Documents

In [33], making whole test issue effective starts with reviewing requirements and test plan. Testing tasks in test plan, or if there is not a plan, applying the following listed questions to requirements can help designers in order to break down testing tasks.

- What should be tested? : Scope of testing

- When should test procedures be developed? : It is suggested to develop test procedures as soon as requirements are available. After deciding what to test, the priorities of tests should be determined. High priority tests should be tested first.
- How should test procedures be designed? : In order to design the appropriate and effective tests, it is necessary to consider the parts that make up the system and how they are integrated.

### Prototypes

Prototypes also give valuable information about inconsistencies or incompleteness of design and provide a basis for developing correct, sufficient and effective test procedures about the system [33].

### Exploratory Testing

Furthermore, for discovering system's functionality and operational boundaries exploratory testing is advised in [33] in projects where there is not much knowledge about the system under test or functional requirements are informal or absent. Exploratory testing identifies test conditions based on an iterative approach. The problems found early in exploratory testing helps focusing the direction of later test efforts [33]. Exploratory testing helps engineers to cover the most important issues in test plans while it is impossible to cover all possible test scenarios, variations and combinations in limited time.

### Software Inspection

Inspection is a process that is used in verification and validation of hardware based systems. Inspection is also tried to be inserted in software development processes [20]. Software inspection is a set of methods which evaluate the user interfaces to find usability problems and is easy to apply and cost-effective. Software inspection allows finding defects earlier, thus reducing rework cost.

Research in [20] reveals that software inspection is as powerful as software testing in finding defects after changes or new versions of the software. Combining software inspection and software testing to cost-effectively find and eliminate defects, and provide reliability. Combination of these processes can also increase testing effectiveness by decreasing time and cost of testing process and increasing the test effectiveness metric by finding much more defects that software testing can miss.

#### Test Case Prioritization & Reduction

Test Case Prioritization (TCP) introduces the planning of the execution order of the test cases in order to increase the effectiveness of testing activities by improving the rate of detection of deficiencies [31]. Another similar definition made in [34] such that TCP techniques schedule test cases for execution in an order that attempts to maximize objective function (Fault detection, test coverage, reliability, etc.). TCP techniques discussed in [34] are mainly code based techniques. Some metric collection techniques in [31] such as following can be applied to hardware based systems when applying TCP techniques:

- Test Coverage,
- System failure data for a component by previous test efforts,
- System failure data for a component reported by customers in the field.

Test Case Reduction (TCR) techniques are also discussed in [34]. TCR techniques help to find out effective subset of test cases during maintenance phase and are helpful in reducing the testing cost [34]. TCR techniques give the advantage of spending less time for executing test cases. However, there may be reduction in fault detection rate while applying TCR techniques [35]. TCR techniques are generally applied as follows:

- A test suite is selected,



- Faults detected by original test suite is written down,
- Test suite is reduced by applying TCR technique,
- Faults detected by reduced test suite is written down,
- Percentage size of reduction and percentage of fault rate detection is calculated and compared [36].

Another method described in [37] is Cumulative Test Analysis (CTA) technique which reduces time to find defects by prioritizing and minimizing testing. In this method, test areas are chosen to target product areas having the highest risk of defects. By using CTA, it is predicted that test team will run “as few tests as necessary”, rather than the traditional “as many tests as possible”.

Concluding the effectiveness assessment issue, it is understood that the whole testing issue from its planning to documentation should be done carefully, systematically and in a clever way to accomplish efficient use of time and manpower; and to cost effectively end testing processes by finding all critical deficiencies that can cause loss of money and time during operational life of system under consideration because of high maintenance costs.

## **CHAPTER 3**

### **UML-BASED BEHAVIORAL SYSTEM TESTING PROCEDURE**

Systems engineering is about seeing the big picture, specifying and developing large, complex systems and system of systems [38]. It has been claimed that SysML which is based on UML 2.0 has all the necessary features for systems engineering [38][39]. SysML reuses a subset of UML and provides additional extensions, specifications and rules to satisfy the requirements of system engineering processes [39].

Using modeling languages like SysML or UML helps systems engineers in various ways. The most important benefit of using SysML/UML in system engineering processes is having a standardized and comprehensive system specification models. This brings consistency between syntax and sub-elements (requirements, diagrams, models) [38]. Moreover, graphical symbols used in modeling diagrams have unambiguous meaning considering textual statements used in conventional system development processes. This will bring a noteworthy decrease in miscommunication between developers and others (testers, buyers and users).

This thesis work only covers the system verification issue of the whole system engineering process. In Chapter 2, literature about the system testing approach based on UML is investigated. There are methods developed for just structural

testing, methods developed for both structural and behavioral testing and methods developed for only behavioral testing. Structural testing covers the interaction of inner system modules opposing to the functionality of system whereas behavioral testing verifies the response of system against the interaction between user and system.

The objective of this study is to propose an effective process for system verification and user acceptance testing, based on applying testing methods and advices that are given in Chapter 2. As it is understood from above paragraphs, testing methods introducing UML into testing phase are chosen because of their adaptability to system thinking.

In summary, UBST Procedure, the procedure that is presented throughout this chapter, is actually an improved test case generation process for hardware-based systems' system verification tests. UBST sums up the applicable parts of useful methods in software testing and system testing processes that are discussed in Chapter 2 of this thesis work. The systems under tests that are discussed in this thesis are investigated in their system verification and validation period. Because of this situation, a planning and extra documentation effort for whole project lifecycle is considered as unnecessary. So, only planning of testing and necessary documentation for the system verification and validation (system testing) era are going to be handled. UBST sums up the applicable parts of useful methods in software testing and system testing processes that are discussed in Chapter 2 of this thesis work.

The first step of UBST will be the investigation of requirements and specify their testability. The next step is the identification of the verification methods of requirements and handling of necessary test planning and test case generation processes in order to have a test plan and procedure.

In this chapter, the refinement of requirements and planning issues will be discussed in summary. But, the chosen UML-based test case generation technique

will be given in detail and information will be given about how it will be applied to hardware-based systems.

### **3.1. PRELIMINARY WORK BEFORE TEST CASE GENERATION**

In the beginning of Chapter 2, the definitions of acceptance testing and system testing are given from different sources. The basic definition of the scope of system testing and acceptance testing that can be gathered from [1], [2], [13] and [21] is the verification and validation of requirements. The requirements can successfully be verified if and only if they are clear, unambiguous and verifiable (measurable, testable) [5]. System requirements should be refined and finalized until all requirements made unambiguous, verifiable and clear by exchanging knowledge between developer and user [2].

After requirement refinement process, the verification methods for each requirement that are going to be verified must be defined with the agreement of developer and user. Then again with the agreement of developer and user, the acceptance criteria should be defined for each requirement which the corresponding verification step should satisfy [5][6].

Whole information discussed above can be put into together in a table which can also be named as the RTM. RTM contains the information of number of tests required, verification steps and types of verification steps [32], so does the table that we advice above. This table can be built as the one given in Table 3-1.

Constructing RTM is the last step before going on with test case generation process. The requirements are refined, requirements that are going to be tested are defined and acceptance criteria of requirements are defined. All the information for building models that are going to be used in test case generation process will be in hands of developer after building RTM adequately.

Table 3-1: Example of RTM

Requirement No	Requirement	Verification Technique	Acceptance Criteria	Verification Step
7.2.1	V/UHF Transceiver System will consist of 1 CU, 1 RFSU, 1 V/UHF Filter and 1 UHF Amplifier unit.	Inspection	All units should be included in configuration.	xyz
....	.....	.....		.....

### 3.2. TEST CASE GENERATION PROCESS

The use of UML models in systems engineering process is a popular research area. A recently published standard likewise UML specification named as SysML [39] is prepared by the contributions of many large industrial companies and will be used as a reference document for managing diagrams in a systems engineering viewpoint. SysML does not advise a way to generate test cases or any other processes but standardizes the diagrams and elements that are used in system modeling and development. SysML uses most of UML models, makes some extensions on the used one or does not make any change and just use them. In this thesis work, the UML-based approach to system testing is used as the test case generation step of UBST. While using a hybrid of UML-based approaches described in [14], [15], [16], [17], [18] and [19] for system test case generation process, SysML models will be tried to be integrated into this hybrid approach.

#### 3.2.1. Use Cases and Use Case Diagrams

The common base of all the approaches discussed in [14], [15], [16], [17], [18] and [19] is use cases. In all test case generation processes, first use case diagrams and corresponding textual use cases are constructed before going on with sequence, activity or state diagrams.

Use case diagrams actually describe the usage of the system from the viewpoint of users (actors). Use cases can be also viewed as functionality that is accomplished through the interaction between system and actors [39]. There are three different relationship commonly used in use case diagrams which are:

- Communication relationship (Straight Line): Actors are connected to use cases via communication paths,
- Include relationship: Provides a mechanism for describing common functionality or scenario that is shared between use cases,
- Extend relationship: Provides optional functionality to overcome a use case's scenarios other than main success scenario [39].

Textual use cases are generated from use case diagrams keeping its structural connections and relationships. Textual use case specifies the pre and post conditions, main success scenario, extensions and variations of the use case [14][39]. So, it includes all the possible interactions between user (actor) and system under test. Because of this extensive information included in textual use cases, they are used in generation of sequence, activity and state diagrams. In those diagrams, system's functionality and behavior corresponding to actors' inputs to the system are covered. Moreover, as it is mentioned before, these diagrams are used to generate test cases for system testing processes.

The first step in test case generation is to understand the system, its working scenarios, its specifications, its functional characteristics and its performance issues. After understanding the system clearly, use case diagram for executing behavioral system testing should be modeled, the use cases that the actor is in interaction should be specified and the common use cases included in other use cases should be revealed for ease of modeling.

### **3.2.2. Choosing the Model**

The UML-based test case generation methods are introduced in Chapter 2, and their applicability is also discussed. But remembering again, generating test cases from state diagrams [14][17], sequence diagrams [18] and activity diagrams [15] can be possible.

In general, all the approaches use "use cases" and generate a behavioral diagram, than make changes and derivations on those behavioral diagram, and finally they generate test cases from the modified or generated diagrams. Model will be chosen by investigating the details of the methods.

In Chapter 2, applicability of these models is also discussed. The most important criterion is the suitability of the method to the system under test in elimination of the methods. Since in this thesis work, the system under test is a hardware-based system and scope of this thesis is to make the system testing (system acceptance testing, functional system testing, behavioral system testing) effective, the test case generation method should specifically be interested in black-box testing instead of white-box (structural) testing.

The approaches in [18] and [19] are usable when considering sequence diagrams' nature. The messaging between modules and system and actor is apparent and sequentially available. But, in system testing or acceptance testing, the interactions between modules and units form the system is out of consideration.

Approaches using state diagrams described in [14] and [17] can be both considerable as successful in generating high-level test cases. But, state diagrams are more formal diagrams than sequence and activity diagrams. They integrate complex semantics, which makes them hard to generate and handle compared to activity and sequence diagrams. Moreover, sequential actions are hard to be defined and resolved in state diagrams which make it harder to realize the interaction flow during use case scenarios. Another drawback of approaches discussed in [14] and [17] is that for both, there are too many manual work and

additional effort that test designer should handle which can make generating test cases for hardware-based systems with a formal approach not as efficient as generating them without a formal and planned approach.

The most suitable approach for the system of interest in this thesis work is the UML-based approach by using activity diagrams which is discussed in [15]. The approach excludes inner system interactions and structural behavior and considers only functional behavior of system that is responses to actor inputs. This feature of the approach and the simplicity of application push it one step forward from other approaches discussed in this section.

### **3.2.3. Transforming Use Cases to Activity Diagrams**

It is advised to build an activity diagram which provides a functional system model instead of a technical model [15]. Activity diagrams start with a starting node, which is the initial state of the use case. The below list which are included in textual use cases should be regenerated as “action nodes” in activity diagrams:

- All the actor action steps,
- System actions which run processes and generate outputs which force the actor to make a decision.

Moreover, decision nodes are should also be marked. Decision nodes show the necessary decisions that the actor or system should perform by considering the outputs of previous actions or inputs. Furthermore some other requirements given in [15] for building activity diagrams are:

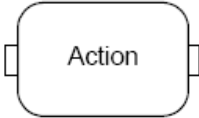
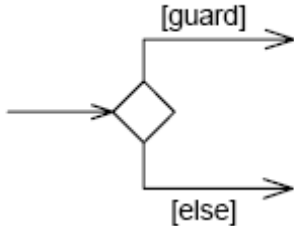


- All objects providing information to a manually executed function are modeled as input pins of actions,
- Objects, which are changed are modeled as input and output pins of actions,



- Objects created during the execution of an action are modeled as output pins of actions.

Here, objects can be referred as system responses in our case. The semantics of activity diagram that are going to be used in this thesis work are given below in Table 3-2:

Table 3-2: Activity Diagram Semantics [39]

Node Name	Semantic (SysML)
Action Node (with input and output pins)	
Decision Node	
Initial Node	
Final Node	

### 3.2.4. Interaction Flow Diagram – Interaction Flow Graph: The Intermediate Steps

The activity diagrams become very detailed when the processes which are executed by systems are added to the diagram. In [15], the viewpoint of user or tester is considered only in generating test cases, the approach considers the system as black-box, and is not interested in inner system processes or inter-module interactions in the system. Hence, these processes should be identified and eliminated from activity diagram in order to generate a test procedure which will

test the behavioral responses of system as a black-box, accepting inputs only from user or tester and giving outputs only to them.

Interaction Flow Diagrams (IFD) represents the control flow of inputs and outputs that a tester has to state to the system and expect from the system [15]. Internal actions defined in activity diagrams are subtracted in IFD. Moreover, IFD is considered as the starting point to generate test cases as it contains only user interactions. IFD is simpler than activity diagrams with its reduced syntax and increases the simplicity of processing the information in it. Elements of IFD are given in Table 3-3 below:

Table 3-3: IFD Semantics [15]

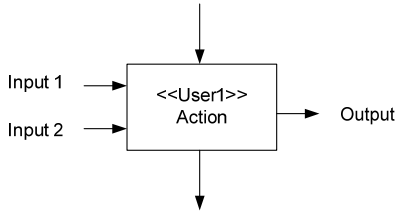
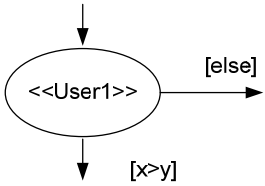
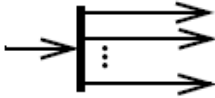



Node Name	Semantic
Action Node (with input and output pins)	
Decision Node	
Fork Node	
Join Node	
Start Node	
Stop Node	

Figure 3-1 shows the mapping between activity diagram and IFD. It is seen that the action which is not assigned as a user action is not mapped to IFD. This is because of IFD's viewpoint. An end user cannot verify the internal functions of the system.

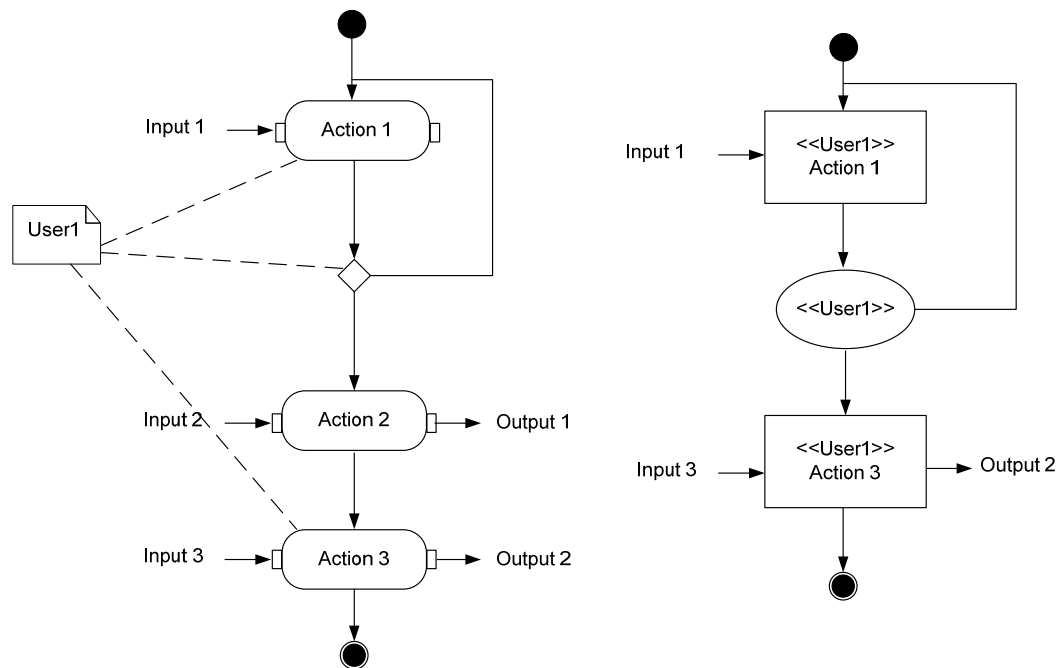


Figure 3-1: Mapping Activity Diagram (Left) to IFD (Right)

The last step before test case generation process is converting the IFD to IFG. In spite of IFD, IFG cannot contain any cycles in it. This is achieved by applying a Depth-First-Search algorithm, which ensures that each cycle and each combination of cycles are executed only once [15].

The IFG is a directed graph and each path of IFG corresponds to a test scenario. In our case, these scenarios can be main success scenario, scenarios corresponding to extensions and scenarios corresponding to variations. So, if IFD has a cycle in it, IFG will have multiple occurrences of nodes that the cycle contains. A simple example converting the IFD given in Figure 3-1 is given below in Figure 3-2:

As it is seen from the Figure 3-2, decision nodes are not a member of IFG. There are two distinct paths in IFG, where each path corresponds to a scenario. It can be seen that, the only cycle that IFD has been included in IFG and executed only once in a path. Moreover by applying the approach in [15], it can be seen that all paths coverage is attained, that is all possible scenarios of the corresponding use case is covered.

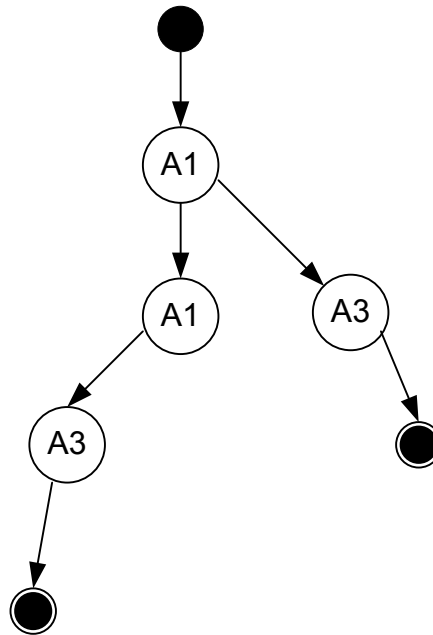


Figure 3-2: An example of IFG

### 3.2.5. Generating Test Cases and Constructing Test Plans

As it is mentioned above, each path of IFG corresponds to a test scenario. Since the graph has no cycles, these paths can be analyzed and extracted with a Depth First Algorithm [15].

In hardware-based systems, generally, at least one of those distinct paths corresponds to the main success scenario and the others to scenarios of extensions and variations. Moreover, for the case in this thesis work, pre and post conditions of the action nodes should be specified and this information should be transferred firstly to IFD, then to IFG and at the end to test plan. Because, execution of some

states can only be possible if the needed preconditions are supplied. Table 3-4 gives a format for a test case scenario test plan.

This simple example gives the tester simple instructions to overcome the test procedure, and verify the system considering the possible outputs. By the approach in [15], the tester will have a full coverage of operational scenario of the system, which also helps the tester to decide to verify if the system behaves as specified in test plan.

Table 3-4: Example Test Plan Table

<b>Use Case Name</b>	.....			
<b>Scenario No</b>	TS1			
<b>Test Case #</b>	<b>Pre-Conditions</b>	<b>Input</b>	<b>Summary</b>	<b>Output</b>
TC1	-	Input-1	Action 1	-
TC2	-	Input-1	Action 1	-
TC3	-	Input-3	Action 3	Output 2

## **CHAPTER 4**

### **SYSTEM TESTING PROBLEMS AND PREVIOUS SYSTEM TESTING RESULTS**

In this chapter, system testing of naval communication systems which are designed, produced and integrated in the company that the author is employed will be investigated. Positive and negative properties of whole testing process will be discussed in terms of testing effectiveness, and metrics calculated from previous testing activities will be shared. The assessment of current testing procedures performed will be done, by introducing some senior engineers' and managers' ideas, as obtained in the interviews conducted within the scope of this study. Specifically, 1 systems engineering manager, 2 senior systems engineers, 2 expert engineers and 2 engineers have been interviewed and the material in the rest of this chapter mainly reflects their opinions, as synthesized by the author. Moreover, the testing procedures of previous systems testing activities will be investigated and metrics from these testing activities will be gathered.

Naval Communication Systems produced for Turkish Navy consist of Internal Communication Systems such as Sound Powered Telephone System, Telephone System and Announcing System, and external communication systems such as HF Subsystems, V/UHF Subsystems and a master switching system which manages internal and external communication systems.

Naval Communication Systems do not have any interface to the other electronic systems of the ship, except LINK, which is a system that shares information between ships in a fleet by using V/UHF or HF Transceivers. But this system interface is kept out of the scope of this chapter.

Naval Communication Systems differ from other systems such as radar systems in a way that they include many subsystems; some are Commercial Off-the-Shelf (COTS) products, some are produced by subcontractors and some are designed and produced in the company that the author is employed. Because of the variety of subsystems, the probability of finding defects in testing due of integration problems is very high.

Moreover, another difference of naval communications systems from other systems is that communication systems are built uniquely for each and every project, since all different kinds of ships have different system requirements and each project need development process. It can be understood that mass-production of naval communication systems is impossible.

Testing of V/UHF Radio System is the main system that will be investigated in present chapter. The system has 4 configurations namely V/UHF Transceiver AC and DC configurations, Ultra-High-Frequency (UHF) Transceiver and Guard Receiver. All configurations have the same radio transceiver, but this transceiver is adjusted to different modes, in order to meet the related configuration's requirements. V/UHF Transceiver configuration consists of a Central Unit which includes transceiver and controls, a COTS V/UHF Filter, a COTS UHF Amplifier and a RF switching unit to integrate the COTS products with the transceiver. The previous version of V/UHF ship configuration mentioned above is still used in the Turkish Navy in a class of frigates and some design and testing experience was gathered from the modernization projects of related class of ships. Also, all four configurations of V/UHF Radio Systems are integrated to a corvette, to a class of patrol boats and are going to be integrated to some export ships and new ships planned to be built.

#### **4.1. ACCEPTANCE TESTING/SYSTEM TESTING ACTIVITIES AND PROBLEMS**

In the company that the author is employed, in the department of system engineering, integrated communication systems for naval ships are designed. Considering these communication systems, they are designed and built specially for different kinds of ship platforms. That is, a mass production of the communications systems is not possible. In the acceptance of those systems, there are 3 steps: First step is Factory Acceptance Tests (FAT), the second step is Harbor Acceptance Tests (HAT) and the last step is Sea Acceptance Tests (SAT). After completing the last step successfully, the system is fully accepted by project owner or it is temporarily accepted. The acceptance type of system changes in different projects because of contract issues and unsatisfied system requirements.

The acceptance viewpoint of the company's responsible divisions interested in naval systems is meeting and satisfying all the system and customer requirements in the three steps of acceptance tests mentioned above, that are FAT, HAT and SAT. The functional, non-functional (reliability, maintainability, etc.), performance and operational requirements are all tested and evaluated with the contribution of developers, customer and user of the project.

The verification and validation issues planned by systems engineering departments and confirmed by the user and the project owner are conducted by the department charged (which changes for different projects) with testing by the supervision of systems engineering department in FAT, HAT and SAT.

In FAT the end user of system meets for the first time with system and their role in this step doesn't go forward from observation for not decreasing the efficiency of test steps. In this step of acceptance testing, a lot of knowledge accumulation is performed between the end user and the developers.



In HAT and SAT, the end user is encouraged to contribute the test in a role of operator of system. In HAT and SAT steps, the end user has more knowledge about the system and its use, because in all integration processes, they are supervising the integration team of company and before HAT and SAT, developer organizes a user training which is generally a contract must. From FAT to HAT, there made some integration tests in ship platforms to be sure that system is ready for full or temporary acceptance, but those tests are conducted by the company that the author is employed, so this test steps are not a formal activity between developer and project owner, and this test steps are controlled only by developer's own quality divisions. But these tests give brief information to the user about the operational and functional use of system that can be a usable knowledge for them in HAT and SAT if they contribute to tests as observers.

The acceptance testing aim of companies, for different projects and for different platforms is mainly verifying the system requirements and validating original user requirements. The verification and testing methods, testing periods, milestones for testing are all variable issues which are decided differently in various projects for different naval platforms and for different requirements.

OT&E, at least with this name, is not applied for communications systems in naval projects of Undersecretariat of Defense Industry (SSM), Navy Force Headquarters (Dz.K.K.lığı) and National Defense Ministry in Türkiye. But, of course operational evaluation of communications systems is made by users of systems that are operator noncommissioned officers and officers that are expert in communications systems. But, the nature of communications systems makes it unnecessary to execute long periods of operational testing and evaluation unnecessary. The communication scenarios that the system offers are operationally executed a lot of times in test scenarios, from time to time by contribution of users as operators, to satisfy user during AT&E.

Considering the system testing processes in the department that the author is employed, it should be noted that there is not a clearly defined procedure and

method neither for system engineering process nor quality assurance process. The sub-systems are generally verified by verifying their system requirements one by one or the verification procedure that is prepared by the responsible engineer of sub-system based on his/her experience system and system requirements. After completion of integration process, both in factory and in original environment, a pre-acceptance test which is conducted to verify system and user requirements by the developer department in order to be sure that there is no fault left in the system before acceptance tests.

#### **4.1.1. Problems**

So far, in the preceding sections, some methods for conducting system tests have been discussed. But, the most important element of the system test and evaluation (T&E) process is always human beings who are engineers generating test cases and planning the whole testing process. An experienced engineer can create more efficient test scenarios and test plans than an inexperienced engineer. Forming a methodology for test case generation and test planning activities can decrease the level difference between an experienced engineer and inexperienced one. Lack of a methodology for testing process can be the source of some problems given below:

##### ***4.1.1.1. Missing Requirements and Operational Scenarios***

An important problem in acceptance testing and system testing is missing some test scenarios that should be covered for successful verification and validation of system under test. The test cases are designed for verifying at least one system requirement. But, while focusing on single requirement verification, some prerequisites and post-conditions that should also be provided while verifying that single requirement can be missed.

For example, in a requirement for V/UHF radios, it is written that the radio should give an output RF power of 10W in Very-High-Frequency (VHF) band in AM mode. V/UHF radios designed as a system which consists of transceiver, filter, RF

switching and amplifier units which are all separate units. So, it is a mistake to measure output power from the output of transceiver because the RF signal is switched to appropriate RF ways by RF switching unit for filtering and amplification. The appropriate measurement should be performed from the output of RF switching unit, because the RF signal traveling through different devices and it is subjected to some attenuation.

The above mistake has been come out due to not considering the requirement as a system requirement and trying to meet it at an incorrect level thereby missing an important test scenario. This mistake costs documenting the detailed test plan again, repeating the test with a true scenario and consequently leading to losses of effort and time.

#### ***4.1.1.2. Insufficient system level tests***

As mentioned above, the system testing approach of systems engineering department is basically “Verify all system requirements”. System level tests are executed for verification of the systems that will be subject to acceptance tests at last. But, there is not a clear procedure or technique for determining the test plan or test methods of system testing. Generally, the responsible engineer of the system under test writes a test procedure in order to verify system functionality. This non-methodological way of practicing system level tests sometimes causes missing some defects that cannot be determined by executing the verification scenarios. Some defects can appear in operational life of system where every scenario is possible. This situation reduces the systems reliability and decreases system testing effectiveness.

#### ***4.1.1.3. Who should be responsible for preparing test documents?***

One of the engineers in systems engineering department stated that system requirements, test plan and test cases should be prepared by the same person and also verified by the same person. This argument was rejected by other colleagues. It is very dangerous in projects to let one person to carry out all the work

mentioned above alone considering lack of a methodological approach for determining test scenarios of system level testing process. Because, a single engineer, whoever he is, a manager or a new engineer, may not detect his own fault. So, all the documents and planning discussed above should be prepared by a team and should be controlled again and again in order not to miss any requirements or verification steps.

Another senior engineer mentioned that whole test plan and test procedure should be prepared and executed by a quality assurance division apart from design department. In order to support his comment he has also stated that designers can prepare the test plan and test cases in a way that the system can certainly pass them. This should be prevented by independent verification divisions or firms who should control every design stage and should have deep knowledge on system design and functionality. In this way, a system that meet specifically all needs of users can be supplied while having a high test effectiveness value. This argument of one of the colleagues can be considered as meaningful for acceptance testing process which is verification and validation step between customer and developer where developer is issued to demonstrate and prove that the system supplied meets all the system and user requirements. But considering system testing process which is a verification process conducted for verifying the system requirements that are derived by developer, this argument becomes partially meaningless. The test plan and test scenarios should be generated by a test team, but it is unnecessary to make them generated by a quality assurance division which cannot determine all the functionality of the system. Besides, quality assurance divisions should control and approve system testing processes for procedural suitability and completeness.

#### ***4.1.1.4. Duplication of test cases***

Here an important problem, duplication of tests which increases testing cost and testing time is stated by one of the colleagues. An example is given such that two distinct test scenarios for measuring the output power value and voltage standing

wave ratio (VSWR) value of High Frequency (HF) transceivers with a test setup which measures the two values in the same time. In this example, both test cases include the same functional and performance characteristics. So, there is no need to run the same functional verification steps twice which is only waste of time and manpower.

#### ***4.1.1.5. Multiple understandings of requirements***

According to a senior engineer working in systems engineering department, the most important problem in communication systems system level tests is that user and developer don't have the same understanding of requirements and acceptance-criteria. This problem occurs when customer does not force the developer to follow formal project development documents and milestones, such as SRS, critical design report, test and evaluation master plan and other engineering documents that should be shared with customer and also that should be approved by customer. This situation is mentioned in previous chapters and it can result in reconstructing test cases, rebuilding test cases and even can force developers to make design modifications on product.

#### ***4.1.1.6. Requirements should be in detailed format***

One of the colleagues argued that our system requirements documents are not prepared in a very detailed format. All operational scenarios or operational requirements which are not specified in contract are not covered in SRS documents. Because of unprofessional personnel that are recruited by customer and end user, generally verification and approval of these documents are done in a way that the developer wants. This situation results in problems in acceptance tests and after project delivery between end user and developer. End user recognize that they should have been covered more functionality than they wrote years ago in contract and force developer to make necessary modifications and additions to cover functional needs. So, in order to prevent these problems during acceptance tests and after delivery, the developer should be in tight

communication with end user whilst preparing systems requirements specification document and they should be both agreed on the document.

Moreover, the manager of system engineering department makes a self-criticism, stating another problem as not preparing systems requirements in a very detailed format which are derived from contract. All possible operational and functional needs, which are not specified in contract, should be covered in SRS document in order to overcome problems that can occur between user and developer in acceptance tests.

#### ***4.1.1.7. Traceability Matrix***

Another problem mentioned by a colleague that, not in our department but, in other firms, traceability of requirements for successful verification of system cannot be done very effectively. There can be a missed requirement that is not verified in acceptance tests, which is a sign of the same requirement never verified for the system, which brings a great risk to project delivery. This problem can be solved by using requirement management software tools such as DOORS, where requirements and their relational links to test documents and other documents can be easily observed.

#### ***4.1.1.8. Lack of supervision on testing process***

The manager of systems engineering department finds the company's testing concept not cost-effective due to having no formal definition of testing time, manpower spent for testing, testing schedule, verification standards and there is no supervision of testing processes. The only important case of testing process is successful delivery of the system. If the system is successfully delivered to user, there is no problem and no one examines the project cycles, development process or testing process deeply. Absence of a formal system development and verification methodology, a formal documentation and quality management process bring the problems stated above.

The most visible problem reported in the research in recent literature and information gathered from colleagues is that there is no methodology, a format or a source that developer can benefit for generation of test cases for user requirements in hardware-based large systems. It is observed that there are many test case generation techniques for software systems while there are none for hardware systems. The methods and techniques surveyed in preceding sections can be rearranged for use in hardware systems. But, it should be noted that before test case generation, user requirements should be grouped and sub-system requirements should be clearly defined which relate to another research area.

#### **4.1.2. Parallel Testing**

It is mentioned above that OT&E is not a formal step in naval communication projects. But, the absence of OT&E in naval projects does not mean that the testing techniques cannot be used for AT&E processes. One interesting approach for optimizing operational tests is parallel testing.

Briefly defining parallel testing, it is the ability of performing multiple simultaneous measurements [40]. In system level, this is the ability of performing simultaneous test cases.

Communications subsystems are not closely integrated with each other. That is, advance testing of different subsystems is not considered necessary. So, some subsystems can be tested parallel to each other. Neither in author's employer company nor in the other companies investigated, parallel testing is not considered in test plans. In the preceding sections, parallel testing was also discussed from another viewpoint. Here, the problem of lack of evaluators (users, project owner and others) is discussed. But, a different viewpoint in parallel testing of separate systems can be obtained in communications systems T&E activities.

Communications systems consist of some similar subsystems and some distinct subsystems. For example, HF Transceiver and HF Receiver subsystems are

considered as separate systems in contract and in systems engineering documents. Parallel testing of these separate systems can be achieved by efficiently planning test cases. Giving a practical example, an HF Receiver's functional test case of receiving at 29,999999 MHz in CW mode can be combined with the test case of an HF Transceiver's functional test of transmitting at 29,999999 MHz in CW mode. If the transmitted audio from transceiver can be heard from the receiver, the two test cases are passed in one step. This kind of approach when developing test cases for AT&E can shorten AT&E period and improve efficiency, and decrease the cost of testing activities in manpower.

#### **4.1.3. Applicability of M&S**

In the system testing and AT&E processes of naval communications systems, use of M&S will not be so effective. Firstly, systems engineering design phase mostly consists of integration work. Secondly, the developed systems vary in many points but the structure of whole communications systems is not varying much. So, system level developers of communication systems don't expect too much surprises while developing and integrating the system. But, when a totally new subsystem enters the system, it can be useful to use M&S techniques for demonstration in system testing process and testing to find probable deficiencies of subsystem from user requirements. But, the developed M&S should not cost much and its developing process should not be very long, in order to preserve test efficiency and cost. Here, a simulation for communication system is diverging a lot from a weapon system simulation or flight simulator.

## **4.2. TEST PLANNING ACTIVITIES**

Test planning activities start with requirements analysis. Briefly, whole system decomposed to subsystems, requirements decomposed to logical sub-levels, testability and measurability of system requirements are determined and documented.



In the company that author is employed, there is not a standard Test and Evaluation Master Plan (TEMP) document format. Despite this, there are document formats for detailed test procedures and test plans for separate AT&E activities such as FAT, HAT and SAT but they give no information about planning or test case generation, just formats. Test cases are planned and information about how the test case should be operated, which measurements should be done and how they should be done are written in a very detailed format by system engineering department and it is presented to project owner and user to confirmation. But, of course the planning of testing activities is done at system level at the beginning of project lifecycle by forming a verification matrix and stating milestones. This verification matrix is presented to project owner for confirmation. Then, calendar for acceptance tests are prepared with project owner and developer.

Aside from the verification matrix discussed in the previous paragraph, in all stages of acceptance tests, which are FAT, HAT and SAT, an “Acceptance and Inspection” document is prepared which consists of detailed information about application of test cases to test subjects, use of test equipment, the connections of test equipment, duration of test case, the limits of performance characteristics of test subjects and many other necessary information in order to conduct the test cases.

TEMP document can be introduced to author’s employer company for naval projects; there is no obstacle for accomplishing this except possible administrative rules. Besides, this document is becoming a project must in SSM projects, but the status of the document cannot be considered as obligatory on project lifecycle. The context of the document should be reworked and document should be made useful and obligatory on project lifecycle. Absolutely, the content of document should be changed from what is offered in [41] which is not suitable for Turkish naval projects’ operation. The testing activities can be defined, test equipments, equipments under test, grouping of these equipments, test methods

(functional test, performance test, analysis, inspection and other kinds of test methods that are changeable for different platforms) can be also defined. Already formed verification matrix can be incorporated with TEMP document, giving references to system requirements and detailed testing plans that are discussed above.

Moreover, by not modifying the TEMP, a new document format can be generated by using TEMP knowledge and suggestions made in [8] and [42] and using experiences of senior engineers who work in the area of system engineering for several years. A document format which covers all planning activities for not missing any important test activity for successful verification and validation issues can be very helpful in acceptance stage, especially for acceptance of system by user without a problem and decreasing workmanship costs.

### **4.3. ORIGINAL TEST PROCEDURES OF SYSTEMS**

This part will discuss the test procedure carried out by systems engineering department to V/UHF-2 Transceiver System and Guard Receiver. There is not a written document that specifies how to test the corresponding equipment. So, the verification of design is done by verifying the requirements in SRS document one by one. As it is discussed before, this procedure is inadequate. The probability of missing system defects which can only be encountered by running real use case scenarios is very high in this type of verification and testing procedure.

Table 4-1 gives the data collected during testing process that is based on verification of requirements in SRS document.

The column of “# of Req. Not Tested” in Table 4-1 corresponds to ambiguous and not testable, immeasurable, unverifiable requirements. It is seen in Table 4-1 that verification of design cannot be made completely because of problematic requirements. Furthermore, every requirement is connected to a unique test case

in original testing process. That is the original testing process is not efficient because of one-to-one match between requirements and tests. This situation creates an inefficient testing process which includes duplication of test cases, loss of manpower and time, and also money. In Table 4-2 the numbers of faults and defects found during and after testing process of V/UHF-2 Transceiver System and Guard Receiver are given in order to understand the status of testing process in a more detailed way.

Table 4-1: Number of Requirements by Verification Status

	<b># of Requirements Verified</b>	<b># of Requirements Not Verified</b>	<b># of Requirements Verified by Testing</b>	<b># of Total Requirements</b>
<b>V/UHF-2 T/Rx System General Requirements</b>	26	3	13	29
<b>CU Functional Requirements</b>	42	5	13	47
<b>RFSU Functional Requirements</b>	17	1	9	18
<b>RCU Functional Requirements</b>	7	3	3	10
<b>Guard Receiver General Requirements</b>	13	3	9	16
<b>Guard Receiver Functional Requirements</b>	70	3	48	73

Table 4-2: Number of Defects Found

	<b># Defects/Faults Found During Testing</b>	<b># Defects/Faults Found After Testing</b>	<b># of Total Fault/Defects Found</b>
<b>V/UHF-2 Transceiver System</b>	5	5	10
<b>Guard Receiver</b>	5	5	10

Table 4-2 shows that nearly half of the defects and faults were found after testing process. This situation cannot be acceptable. Because, this data shows that our testing process is not effective that so many problems cannot be detected by running it. The faults found after testing process are occurred when real use case scenarios run on systems. Because of this fact, real use case scenarios will be considered seriously for verification of requirements in the new version of test procedure that will be prepared in this thesis work.

Table 4-1 and Table 4-2 gave the necessary data to calculate the testing metrics given in part 2.3.1. The calculations of corresponding metrics which are calculated by substituting data gathered in equations (2-1) to (2-3) are given below:

For V/UHF-2 Transceiver System,

$$Test\ Coverage = \frac{13 + 13 + 9 + 3}{29 + 47 + 18 + 10} \times 100 = 37\% \quad (4-1)$$

$$Test\ Effectiveness = \frac{8}{13} \times 100 = 61\% \quad (4-2)$$

$$Test\ Effort = \frac{8}{10} \times 100 = 80\% \quad (4-3)$$

For Guard Receiver,

$$Test\ Coverage = \frac{9 + 48}{16 + 73} \times 100 = 64\% \quad (4-4)$$

$$Test\ Effectiveness = \frac{5}{10} \times 100 = 50\% \quad (4-5)$$

$$Test\ Effort = \frac{10}{10} \times 100 = 100\% \quad (4-6)$$

The metric results for V/UHF Transceiver System show that while test coverage is adequate, it is understood that the low value of test effectiveness metric shows that tests are not adequate to locate the defects and faults of system. On the contrary, the test coverage metric of Guard Receiver shows that the tests are not satisfactory to cover and verify all the requirements. Moreover, the test process which is applied to Guard Receiver is very unsuccessful in finding faults and defects, more than half of the total faults found after testing process. Both V/UHF Transceiver System and Guard Receiver test process can be classified as successful in eliminating defects during test processes when test effort and test effectiveness values are investigated. But, test effort metric is very low because of low value of test effectiveness value. The defects are eliminated but the user found defects which were not detected during testing process.

## **CHAPTER 5**

### **APPLICATION OF UML-BASED BEHAVIORAL SYSTEM TESTING PROCEDURE**

UBST procedure is proposed within the scope of the present study as an improved testing process for hardware-based systems developed in Systems Engineering Department of Naval Communications (SEDNC) in the company that the author is employed. Lack of a standardized testing approach creates problems in system testing processes, reduces testing effectiveness and increases cost of testing as discussed in previous chapters. UBST is developed in order to fulfill the standardized system testing approach necessity of SEDNC to overcome the problems faced during system testing and system verification processes. In this study UBST will be applied to V/UHF Transceiver System's and Guard Receiver's system testing processes and the metrics gathered from this new approach will be compared to the system testing process' metrics that was applied to the systems in May, 2009.

In this chapter, first, SRS document of the items mentioned above will be investigated. The problems of SRS document will be detected and written down. Afterwards, some test planning activity and determination of verification methods will be handled. Later, the improved "UML-Based System Testing Procedure (UBST)" process will be implemented, the test generation procedure introduced in Chapter 3 will be presented in a detailed format for the reader. After applying the generated test suite, metrics discussed in Chapter 2 will be obtained to be

compared and contrasted with those measured earlier. Hence, the benefits of the proposed approach will be quantitatively demonstrated.

## **5.1. INVESTIGATION OF ORIGINAL SYSTEM REQUIREMENTS DOCUMENT**

### **5.1.1. Overview of System Requirements Document**

The SRS document of V/UHF Radio Systems consists of four main parts which describes the general, functional and electrical requirements of unique systems which are V/UHF Transceiver System Configuration-1 (DC Configuration), V/UHF Transceiver System Configuration-2 (AC Configuration), UHF Transceiver System (Configuration-3) and Guard Receiver (Configuration-4). In this thesis, testing process of V/UHF Transceiver System Configuration-2 and Guard Receiver will be investigated. V/UHF Transceiver System consists of one Central Unit (CU), one RF Switching Unit (RFSU), one Remote Control Unit (RCU), one COTS V/UHF RF Filter and one COTS UHF Amplifier. On the contrary, GUARD Receiver is a standalone unit. The CU, RFCU and RCU are designed and produced in author's employer company and their system requirements will be considered separately in the following parts of this chapter. The properties of requirements document is given in Table 5-1 below:

Table 5-1 : Number of Requirements for Each Unit

<b>Requirement Type</b>	<b>V/UHF-2 Transceiver System</b>	<b>V/UHF-2 CU</b>	<b>V/UHF-2 RFSU</b>	<b>V/UHF-2 RCU</b>	<b>GUARD Receiver</b>
<b>General Requirements</b>	29	-	-	-	16
<b>Functional Requirements</b>	-	47	18	10	73
<b>Electrical Requirements</b>	-	6	5	2	3

In Table 5-1 “General Requirements” corresponds to general characteristics that should be met while designing every single unit in the system. In proceeding parts, these requirements will be closely investigated to determine whether they are testable and measurable or not. By the help of this investigation, clues for preparation of better SRS document can be gathered.

### **5.1.2. Detailed Investigation of V/UHF-2 Transceiver System and Guard Receiver Requirements**

In this part, the requirements of V/UHF-2 Transceiver System and Guard Receiver will be investigated to determine whether they are measurable, testable or not. The ambiguous and unclear requirements are determined and marked for later revisions in order to create a more precise and understandable SRS document. Below in Table 5-2, the numbers of problematic requirements for every single unit and systems’ general requirements are given. The requirements that include statements containing words like “may” are defined not verifiable or not measurable. Moreover, requirements that do not have clear meaning and which are not measurable are also marked as not measurable or not verifiable. Also requirements that have definitions that are not explained anywhere in the SRS document and which the designer and customer can understand differently are marked as ambiguous and an example for this situation can be seen by comparing the old requirement and reviewed requirement as follows:

- Old Requirement: RFSU is going to consist of RF relays, a switching control card, and a frequency serial-to-parallel converter unit in order to integrate filter and amplifier for the switching of RF signal of transceiver.

The frequency serial-to-parallel converter unit which is mentioned in above requirement is not a separate design or product from switching control card. So, the verification of the requirement will be problematic if it is not revised. The requirement is re-specified as:



- Reviewed Requirement: RFSU is going to consist of RF relays and a switching control card in order to integrate filter and amplifier for the switching of RF signal of transceiver.

Table 5-2: Number of Problematic Requirements

	<b>Not Verifiable/ Measurable</b>	<b>Ambiguous</b>	<b>Total</b>
<b>V/UHF-2 T/Rx System General Requirements</b>	2	3	29
<b>CU Functional Requirements</b>	1	2	47
<b>RFSU Functional Requirements</b>	0	0	18
<b>RCU Functional Requirements</b>	1	0	10
<b>Guard Receiver General Requirements</b>	2	0	16
<b>Guard Receiver Functional Requirements</b>	1	0	73

It is seen in Table 5-2 that not many problematic requirements are found in the original SRS document, but misunderstanding of any one of the problematic requirements can lead to challenging design problems that may occur in the acceptance period or after system delivery. By using the methods and suggestions mentioned in [2] and [5], the problematic requirements found in SRS document will be revised in order to reach a measurable and clear acceptance criteria between designer and user.

## **5.2. REVIEW OF SYSTEM REQUIREMENTS DOCUMENT**

The problematic requirements of SRS are detected and classified in section 5.1. This document is reviewed with a colleague engineer from the systems engineering department who is familiar with V/UHF Systems and their field use

but has not contributed in the development process of these systems. Knowledge exchange is applied as it is discussed in [41] to overcome the ambiguity and problems of requirements. For V/UHF Transceiver System's "General Requirements", three "ambiguous" and two "not verifiable" requirements were reviewed and rewritten and have become verifiable. Similarly, two of Guard Receiver's "General Requirements", five of V/UHF Transceiver System's "Functional Requirements" and four of Guard Receiver's "Functional Requirements" are revised to make them testable and/or verifiable. Two of V/UHF RCU's "Functional Requirements" are eliminated because of updated design considerations. Moreover, a requirement of V/UHF CU's "Functional Requirements" is removed because it is included indirectly in other requirements. Some examples of these changed, revised and eliminated requirements are given in Table 5-3:

Table 5-3: Examples of Requirements Review Step

<b>Requirement</b>	<b>Reviewed Requirement</b>	<b>Reason of Change</b>
The transceiver of V/UHF T/Rx System can be controlled from only one RCU or from computer.	The transceiver of V/UHF T/Rx System can be controlled from only one of the following units at the same time: <ul style="list-style-type: none"> <li>• RCU on CU</li> <li>• From a remote RCU</li> <li>• From computer</li> </ul>	Ambiguous
The connectors of external interfaces of GUARD Receiver should have spare connections for possible development.	10 % of the external interface connector pins must be reserved for spare for each connector considering possible future improvements in GUARD Receiver.	How much? Not verifiable
The "HAZIR LED" is going to be lid when Filter is adjusted.	Erased. It is an old requirement and contained in other functional requirements such as: The "HAZIR LED" is going to be lid when V/UHF T/Rx System is ready for transmission.	Updated design considerations.

Table 5-3 Continued

Requirement	Reviewed Requirement	Reason of Change
GUARD Receiver's open voice interfaces will be switched to remote units of Communications Systems and the attenuation of these signal levels will be prevented.	GUARD Receiver's open voice interfaces will be switched to remote units, which can be connected with a 100 m cable of 0.25 mm <sup>2</sup> . Necessary design considerations must be taken into account for the prevention of the dropping of the signal level below - 15 dBm.	Not measurable, Not verifiable.

Considering the above examples and other changes, Table 5-1 can be updated as follows:

Table 5-4: Reviewed System Requirements

Requirement Type	V/UHF-2 Transceiver System	V/UHF-2 CU	V/UHF-2 RFSU	V/UHF-2 RCU	GUARD Receiver
General Requirements	29	-	-	-	16
Functional Requirements	-	46	18	8	73
Electrical Requirements	-	6	5	2	3

Only the "Functional Requirements" of V/UHF-2 CU are reduced by 1 as discussed in the previous paragraph.

### 5.3. IDENTIFICATION OF VERIFICATION METHODS

The last step before the application of test case generation steps of UBST will be the identification of the verification methods of requirements and handling of necessary test planning and test case generation processes in order to have a test plan and procedure.

Verification methods should be identified in early lifecycle of the project, a short time after the sides are agreed on the contract. In military contracts, the first versions of the test and evaluation master plan or test plan is requested from developer mostly in 30 to 90 days after contract is signed between both sides.

Requirements in revised version of SRS document which is the subject of this thesis work are investigated in detail and at least one verification method is assigned to each verifiable requirement. In Table 5-5, the number of requirements that are planned to be verified by each verification method is given.

If Table 5-4 is revisited, it is seen that the requirement numbers in that table do not agree with the requirement numbers in Table 5-5. This is because there are some requirements that are verified by more than one method.

Table 5-5: Number of Requirements by Verification Methods

Verification Method	Number of Requirements			
	V/UHF-2 Transceiver System Gen.Req.	V/UHF-2 Transceiver System Func. Req	GUARD Receiver Gen.Req.	GUARD Receiver Func.Req.
<b>Inspection</b>	11	21	6	11
<b>Analysis</b>	10	36	4	21
<b>Test</b>	12	23	7	59

It is unnecessary to run a test case generation method for requirements which are going to be verified by “Inspection” and “Analysis”. Integrating those verification steps to a model or methodology will only increase the complexity of the work and increase the possibility of faults during test case generation processes. Separate verification steps will be added to the test plan for those requirements.

The requirements that should be tested for verification and validation vary in their test types. Most of the test types that will be used for verification and validation of the systems is “functional testing” because of the nature of system testing and acceptance testing. Moreover, some requirements will be verified by performance

testing which verifies the measurements taken during tests if they meet the requirement specifications.

After refining requirements and revising them by exchanging knowledge between customer and developer, the developer will have a clear and approved document to develop the necessary system(s) for successful acceptance of the project. The developer should prepare a test plan and give information about the necessary developmental and operational test steps, identify them, and connect them to specific project milestones in order to achieve successful verification of project from units to system. While preparing detailed test plans, which are much more detailed than documents like TEMP, the developer should supply RTM which specifies at and by which test step, the corresponding requirements will be verified. In this section, we identified the verification methods and in proceeding chapters, after we build the test steps, we will be able to give the resultant RTM.

## **5.4. TEST CASE GENERATION PROCESS**

The test case generation process for V/UHF Transceiver System and Guard Receiver will be started by generation of use case diagrams and use cases in first step as it is described in Chapter 3. Then the process will go on by constructing activity and interaction flow diagrams, which will guide us to construct the IFG that is the last step before test case generation.

### **5.4.1. Use Cases and Use Case Diagrams**

As it is discussed in Chapter 3, use case diagrams and use cases are the fundamental elements of all the UML-based test case generation techniques investigated in Chapter 2. Remembering again, use case diagrams are the most general diagram that is used in system development. Textual use cases are developed from use case diagrams considering the connections between actors with use cases and use cases with use cases.

Preparing use case diagrams and textual use cases requires a very good knowledge of system structure and functional behavior of the system. Developers should be aware of user inputs, expected outputs and erroneous outputs of the system to specify main success scenario, variations and extensions correctly. Generating test cases which have good coverage and are efficient in finding system defects is mainly dependent on well defined textual use cases. Textual use cases should be prepared in a detailed format because of their importance for activity, sequence and state diagrams. Below in Figure 5-1, the use case diagram of V/UHF T/Rx System is given which is the beginning point of test case generation process.

It can be seen that an operator can operate the V/UHF T/Rx in eight different modes and an operator can be in a useless mode where the system is adjusted out of its working frequency band. It is seen from Figure 5-1 that all the nine use cases include the same use case which is the use case describing a successful opening scenario of the system. These use cases can be executed sequentially following one after another, but at least the conditions that are produced at the end of an opening scenario use case should be provided before executing all the use cases. For example, starting "Operate in VHF" use case with a faulty transceiver (means unsuccessful opening scenario) is meaningless, because of a lack of adequate preconditions.

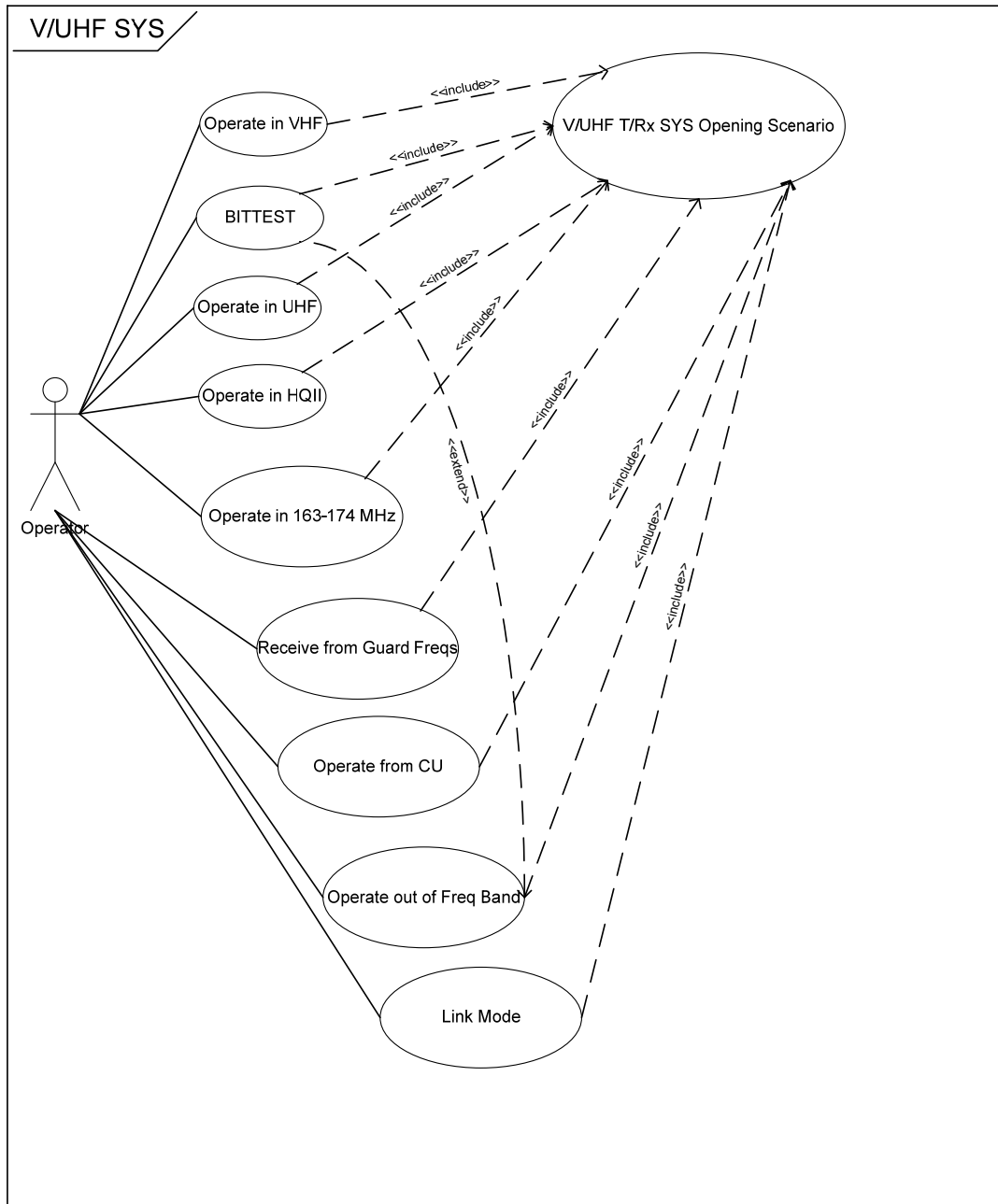


Figure 5-1: Use Case Diagram for V/UHF T/Rx System

For the sake of brevity, only one of corresponding use cases' textual version will be presented in this section. The others can be found in the Appendix. The "V/UHF T/Rx SYS Opening Scenario" use case is presented in Table 5-6 below:

Table 5-6: V/UHF T/Rx SYS Opening Scenario Use Case (Textual)

Name	V/UHF T/Rx SYS Opening Scenario
Goal	This use case describes the opening scenario of the V/UHF T/Rx system and switching to one of operating modes of V/UHF T/Rx System.
Pre-conditions	The system must be connected to power supply appropriately. The MODE switch of RCU in the CU should be in OFF position. The GÜÇ, RF ANAHTAR and GÜÇ YÜKSELTECİ switches of CU must be in OFF position.
Post-conditions	HAZIR LED becomes ON. ARIZA LED is OFF.
Main Success Scenario	<ol style="list-style-type: none"> <li>1. Operator powers the CU by switching on the “Güç” switch.</li> <li>2. Operator sees that the “AÇIK” led in on.</li> <li>3. Operator switches both “SES KONTROL” and “KONTROL” switches to “DAHİLİ” positions.</li> <li>4. Operator switches RCU to T/R mode. RCU opens.</li> <li>5. The CU runs the BIT Test. The “ARIZA” LED blinks..</li> <li>6. System displays “R/T OK” message on RCU. The “ARIZA” led is off.</li> <li>7. Operator adjusts to a frequency between 118-163 MHz.</li> <li>8. Operator powers the RFCU and Filter by switching on “RF Anahtar” switch.</li> <li>9. Operator controls the frequency scale of filter, sees that it suits with the frequency of CU.</li> <li>10. Operator powers the amplifier and sees that amplifier is on.</li> <li>11. “HAZIR” led becomes on.</li> </ol>
Extensions	<p>2a) LED is not on and CU does not start.</p> <p>2a1) Operator switches off “Güç” switch.</p> <p>2a2) Operator controls and corrects input voltage.</p> <p>2a3) Repeat the use case from step 1.</p> <p>4a) RCU does not open.</p> <p>4a1) CU fault, end use case.</p> <p>6a) System displays error message.</p> <p>6a1) Operator switches off “Güç” switch.</p> <p>6a2) Repeat the use case from step 1.</p>



Table 5-6 Continued

	<p>6b) System displays error message twice or more.</p> <p>6b1) Transceiver fault. Cancel Use case.</p> <p>9a) Frequency scale does not match.</p> <p>9a1) Transceiver must be adjusted to 16 BIT Antenna Info. Repeat the use case from step 1 after adjustment.</p> <p>11a) HAZIR LED is OFF.</p> <p>11a1) CU fault, end use case.</p>
Variations	<p>3a) Operator switches “SES KONTROL” and “KONTROL” switches to “DAHİLİ” and “HARİCİ” positions respectively.</p> <p>3b) Operator switches “SES KONTROL” and “KONTROL” switches to “HARİCİ” and “DAHİLİ” positions respectively.</p> <p>3c) Operator switches both “SES KONTROL” and “KONTROL” switches to “HARİCİ” positions.</p>
Included Use Case	None.

The use case in Table 5-6 shows the success scenario and its extensions (unsuccessful scenarios). By specifying unsuccessful scenarios and problematic states up front in textual use cases, it will be hard to miss important points of failure while generating test cases from diagrams that are, in turn, generated using these textual use cases.

The use case diagram of GUARD Receiver which is simpler than V/UHF T/Rx System’s is given in Figure 5-2.

It can be seen that the operator can operate the GUARD Receiver in four different modes. It is seen from Figure 5-2 that all the four use cases again include the same use case which describes the successful opening scenario of the system. Moreover, two use cases share a common use case named BIT. These use cases can be executed sequentially following one after one, but at least the conditions that are produced at the end of the opening scenario use case should be provided before executing all the use cases, for each and every use case’s successful

scenario endings. The textual use case of GUARD Receiver Opening Scenario use case is given in Table 5-7.

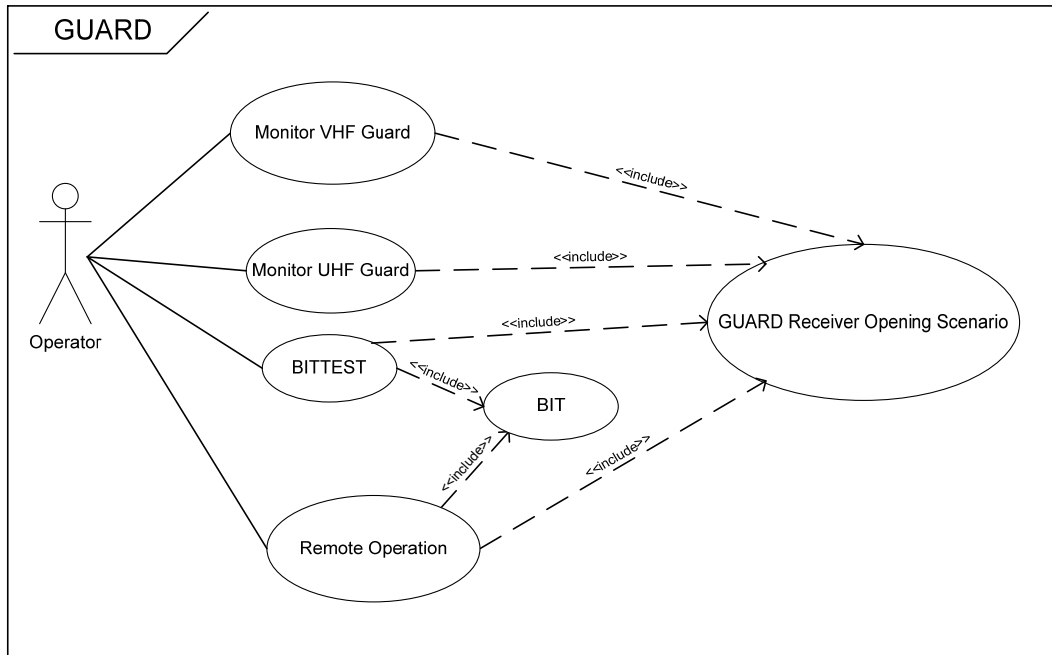


Figure 5-2: GUARD Receiver Use Case Diagram

Table 5-7: GUARD Receiver Opening Scenario Use Case (Textual)

Name	GUARD Receiver Opening Scenario
Goal	This use case describes opening scenario of the GUARD Receiver
Pre-conditions	The system must be shut down. The system must be connected to power supply appropriately. (220 VAC ± 10 % 50 Hz ± 10 % Input Voltage.) The GÜÇ switch of GUARD Receiver must be in OFF position. The BAND switch should be in UHF position.
Post-conditions	ARIZA LED is OFF, selected frequency band's LED is ON.
Main Success Scenario	<ol style="list-style-type: none"> <li>1. Operator switches "KONTROL" switch to "DAHİLİ" position.</li> <li>2. Operator switches on Power switch.</li> <li>3. GÜÇ LED is ON.</li> <li>4. All LEDs on Guard Receiver goes momentarily ON and then OFF again, except GÜÇ LED.</li> </ol>

Table 5-7 Continued

Main Success Scenario	<p>5. System enters BITTEST and blinks ARIZA LED while BITTEST.</p> <p>6. ARIZA LED becomes OFF after BITTEST ends successfully.</p> <p>7. Operator switches BAND switch to VHF.</p> <p>8. System adjusts its working band considering band select switch.</p> <p>9. A BIP sound is heard and selected Band's LED is ON.</p>
Extensions	<p>3a) GÜÇ LED is OFF</p> <p>3a1) GUARD Receiver fault, end use case.</p> <p>4a) All LEDs on Guard Receiver does not go momentarily ON.</p> <p>4a1) GUARD Receiver fault, end use case.</p> <p>5a) ARIZA LED is not blinking.</p> <p>5a1) Guard Receiver fault, end use case.</p> <p>6a) ARIZA LED continues blinking and does not stop.</p> <p>6a1) GUARD Receiver fault, end use case.</p> <p>6b) ARIZA LED is ON.</p> <p>6b1) Repeat the use case from step 1.</p> <p>6b2) GUARD Receiver fault, end use case.</p> <p>9a) No BIP sound or selected band's LED is OFF or deselected one's is ON.</p> <p>9a1) GUARD Receiver fault, end use case.</p> <p>9b) Selected Band's LED is OFF, deselected is OFF.</p> <p>9b1) GUARD Receiver fault, end use case.</p> <p>9c) Both band's LEDs are glowing.</p> <p>9c1) GUARD Receiver fault, end use case.</p>
Variations	<p>1a) Operator switches the "KONTROL" switch to "HARİCİ" position.</p> <p>7a) Operator switches BAND switch to UHF position.</p>
Included Use Case	None.

The use case in Table 5-7 shows the success scenario, its extensions (unsuccessful scenarios) and variations. It is seen that step 7 can be executed in two different ways. For complete verification, these two variations are needed to be covered in test cases. Again, the problematic states described in “extensions” will be used for upcoming test generation issues.

#### **5.4.2. Generating Intermediate Steps: Activity & Interaction Flow Diagrams**

The textual use cases that are created for V/UHF Transceiver System and Guard Receiver given in

Table 5-6 and Table 5-7 are used to produce activity diagrams discussed in Chapter 3. The activity diagrams are the first graphical artifacts that describe the functional behavior of the system considering the UBST.

The activity diagram constructed from “V/UHF T/Rx SYS Opening Scenario” textual use case is given in Figure 5-3. When it is compared with the related use case, the activity diagram includes all the steps to functionally operate the main success scenario. Moreover, it also handles the extensions and variations, by using decision, fork and join nodes. As it is mentioned before, the activity diagram gives an understandable and clear graphical specification of the whole use case.

Here in the activity diagram given in Figure 5-3, the pre and post conditions of the use case are not specified in order to prevent complexity on the diagram. This action prevents the complexity while converting activity diagrams to interaction diagrams. But it also assigns an extra action for developer or tester which is tracking preconditions and post-conditions and integrating them in the final step of UBTS where the test cases are generated.

It can be seen that the actor actions are marked in the activity diagram as described in Chapter 3 for making it easy to transform activity diagram into interaction flow diagram. The V/UHF Transceiver System opening scenario and GUARD Receiver opening scenario given in Figure 5-3 and Figure 5-4 are composed of a series of operator actions such as powering and mode selection

actions. These actions do not force the system to accomplish complicated actions. The system mostly gives momentary responses immediately after user action. Only the “BIT TEST” activity that the system runs is considered as important to specify in activity diagrams.

The next step after generation of activity diagrams is the generation of IFDs. As discussed before, IFD represents the control flow inputs and outputs that the user must enter and expect from system. Internal actions that are not requesting inputs from user and not providing outputs to user for evaluation are kept out of IFDs. This characteristic of IFDs makes it easier to generate test plans by reducing the number of nodes and unnecessary system operations that are represented at activity diagrams. The IFD diagrams of V/UHF T/Rx System and GUARD Receiver are given in Figure 5-5 and Figure 5-6.

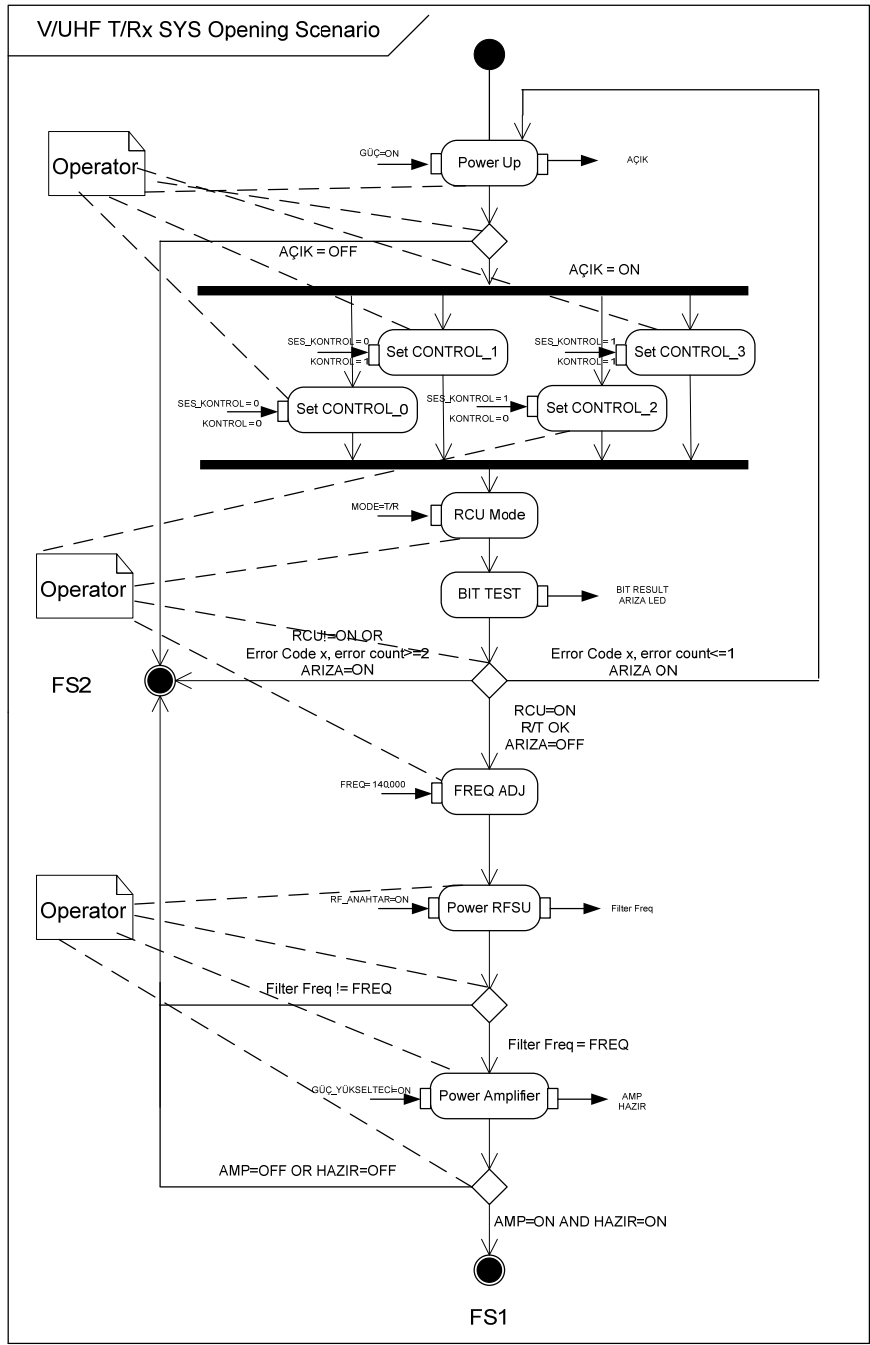


Figure 5-3: V/UHF T/Rx Opening Scenario Activity Diagram

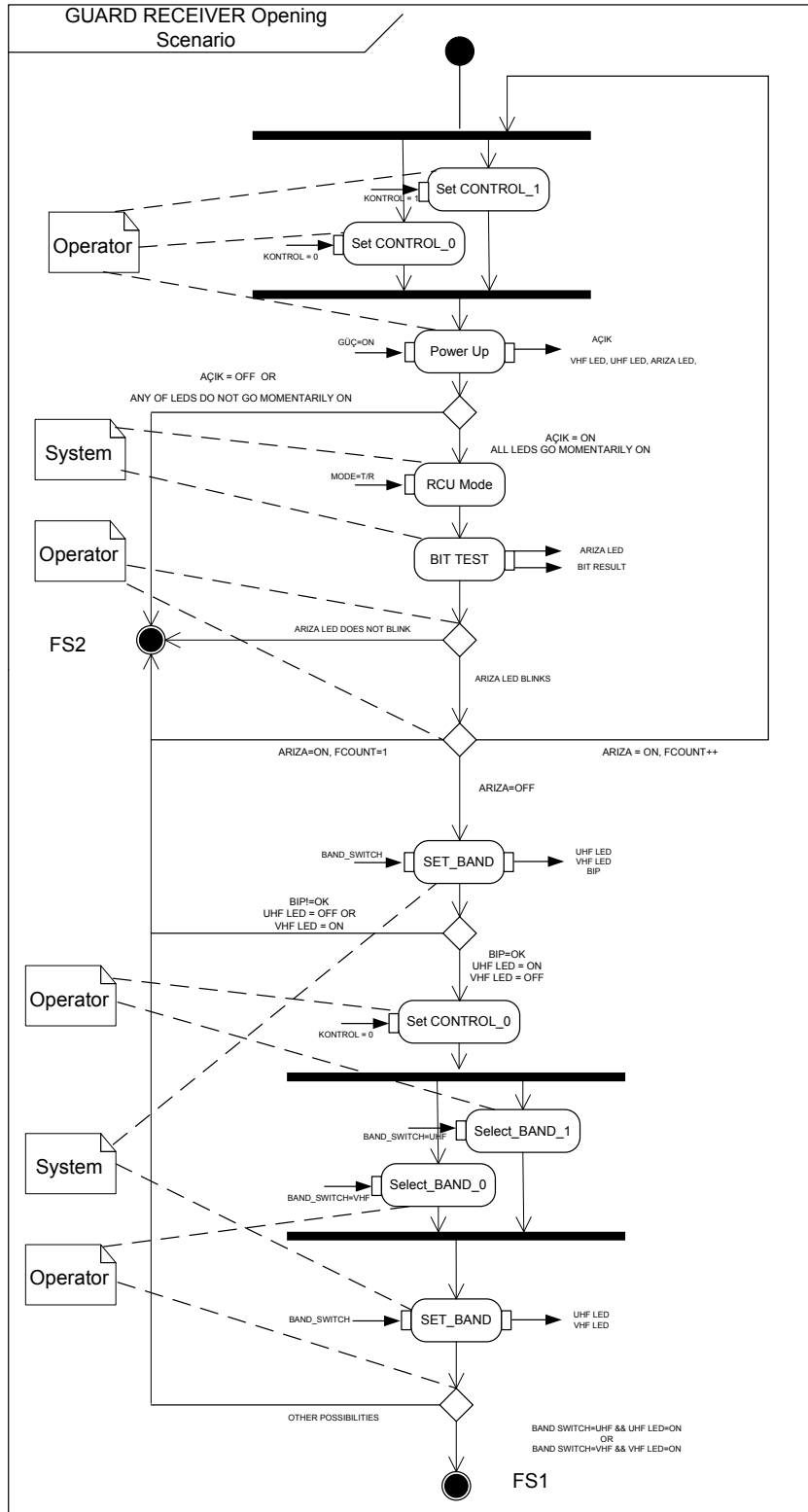


Figure 5-4: GUARD Receiver Opening Scenario Activity Diagram

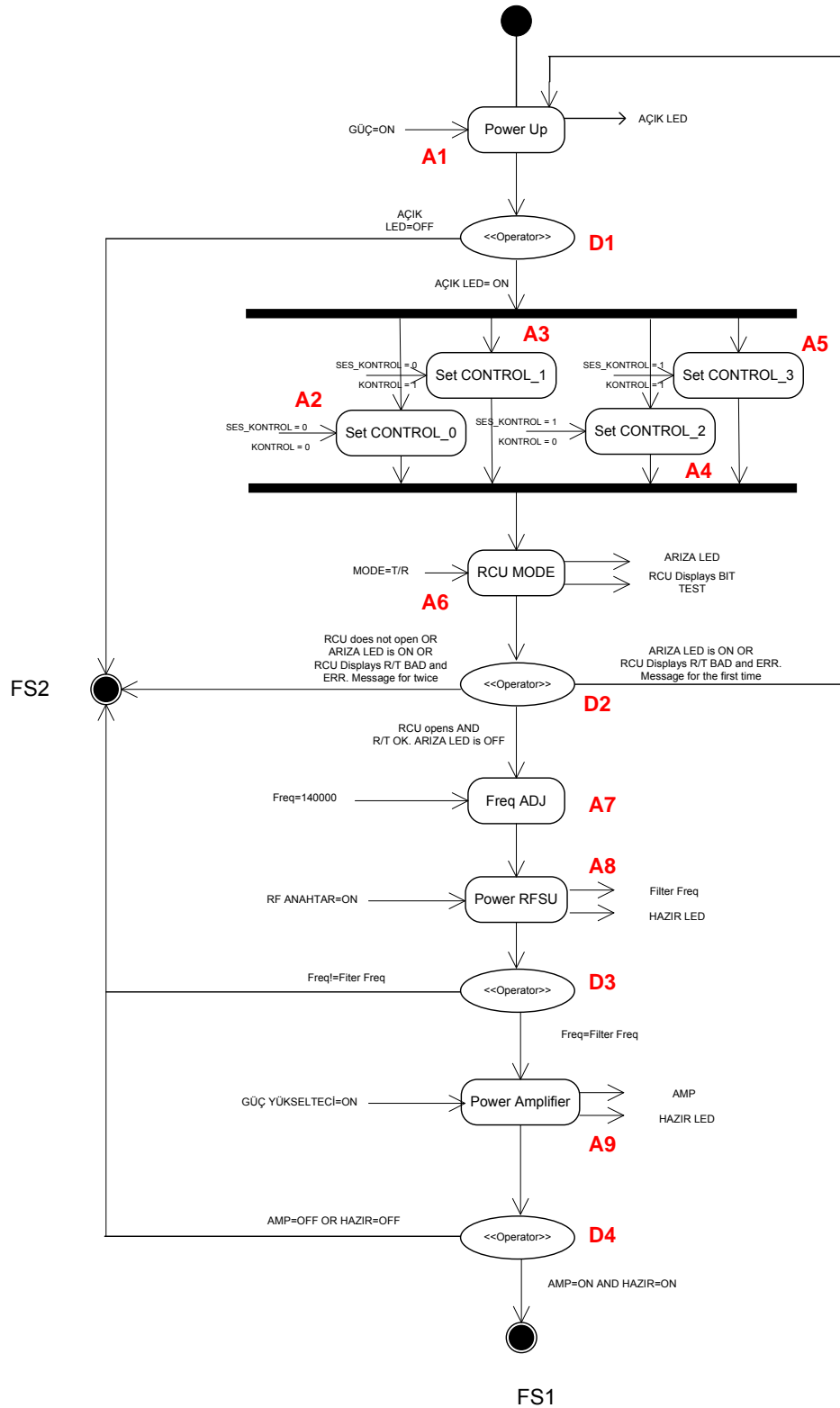


Figure 5-5: V/UHF T/Rx System Opening Scenario IFD



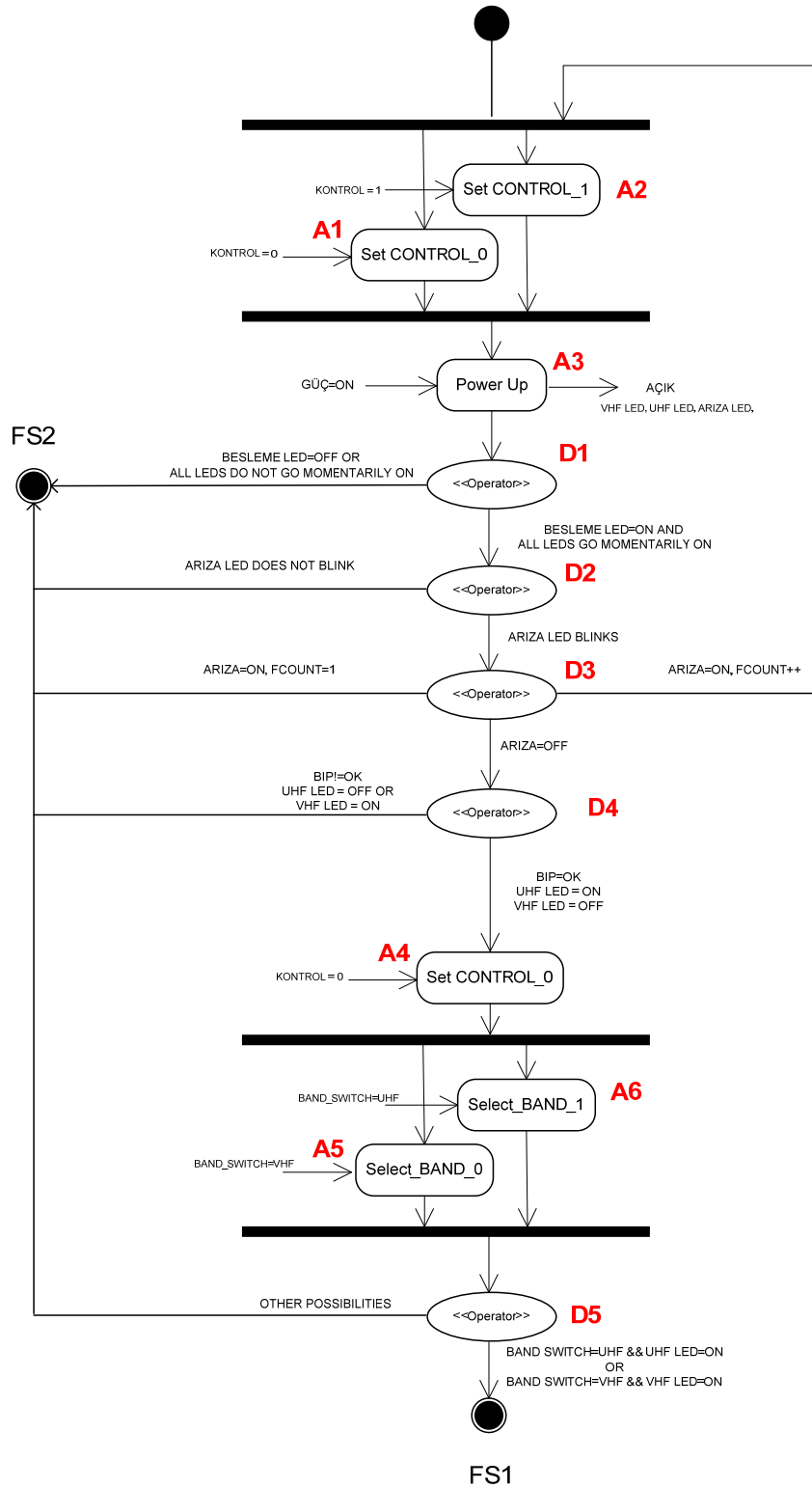


Figure 5-6: GUARD Receiver Opening Scenario IFD

As it is mentioned before, the IFD should include only user related actions. But absolutely, there are many actions handled by systems and their outputs are judged by user. In above and other IFDs, generated for this thesis work, these outputs are rearranged and connected with the user action which is responsible for the initiation of corresponding system action. This operation which is done while transforming from activity diagrams to IFDs makes IFDs easier to read and analyze for the generation of IFGs and test cases. FS1 and FS2 are the final states where FS1 represents the main success scenario final state and FS2 represents the failed test scenario final state. If scenario ends with FS2, test scenario is failed and necessary modifications should be handled on the system before executing the corresponding test scenario again.

### **5.4.3. Generating IFGs and Test Cases**

The last step of test case generation process in UBST is generating IFGs and using them to get the textual test scenarios which will be ready to be executed on the system. The IFGs of V/UHF T/Rx System Opening Scenario Use Case and GUARD Receiver Opening Scenario Use Case are given in Figure 5-7 and Figure 5-8.

In Figure 5-7, it is seen that there are four main branches that are coming out of A1. When these branches are compared, the only difference between them is the next action of A1, which are actually variations of the same action. There will be no need to verify again and again the same test steps. But, there should be a rule to define the verification of the actions which are variations of each other. So as a rule, the branches should be executed until the first action after the node of variation is executed successfully. That is, the branch of A2 should be executed fully, but execution of the branches A3, A4 and A5 should be stopped after having the successful outcome of A6. By this means, duplication of tests will be avoided for the corresponding use cases. For whole test procedure, duplication of tests will be considered and duplications will be handled after the creation of test cases.

Coming to the faulty final states, represented as FS2 in activity diagrams IFDs and IFGs, are the faulty ends of use cases. If there will be an error or fault during the test scenarios, the extension scenarios defined in textual use cases are executed and mostly, the scenario directly branches to FS2 which is actually the halting of test scenario with faults. In theory, for every final state node represented in IFG, there should be a test scenario ending with the corresponding node. But, in practice, it will be impossible to cover these faulty final states if the system is perfect and has no faults. To prevent crowded test sets, if a node has a direct transition connected to a faulty end state (FS2) because of a faulty system output, this extension will be defined in a column as “Faulty Output”. By doing this, the tester will be prepared for the occurrences of faulty outputs and the unnecessary and/or impossible test executions will be out of consideration.

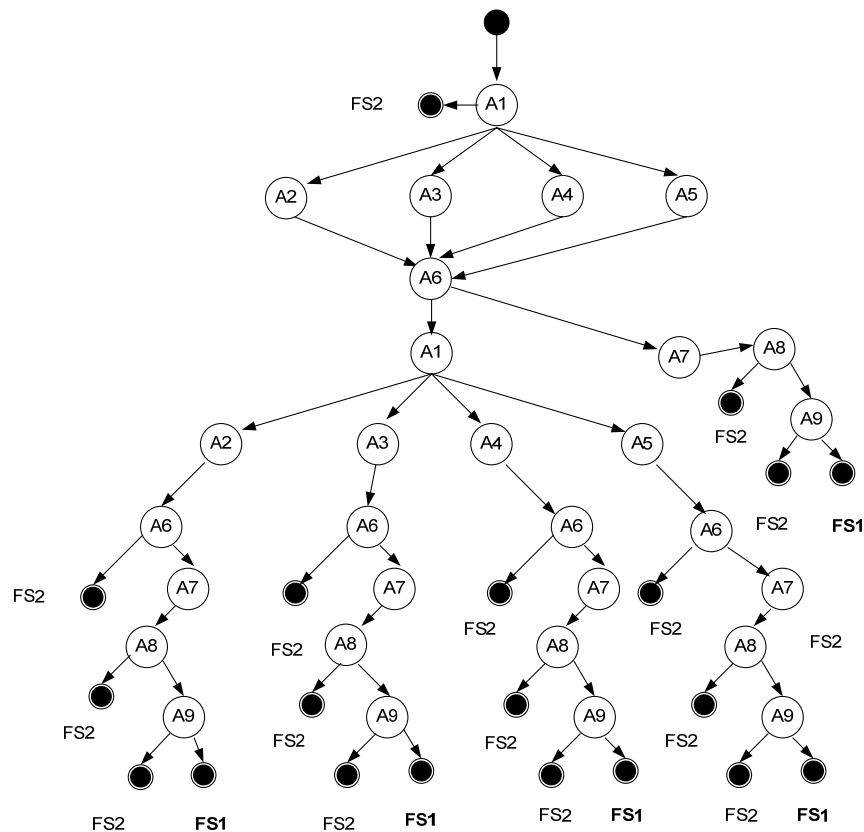


Figure 5-7: IFG of V/UHF T/Rx System Opening Scenario Use Case

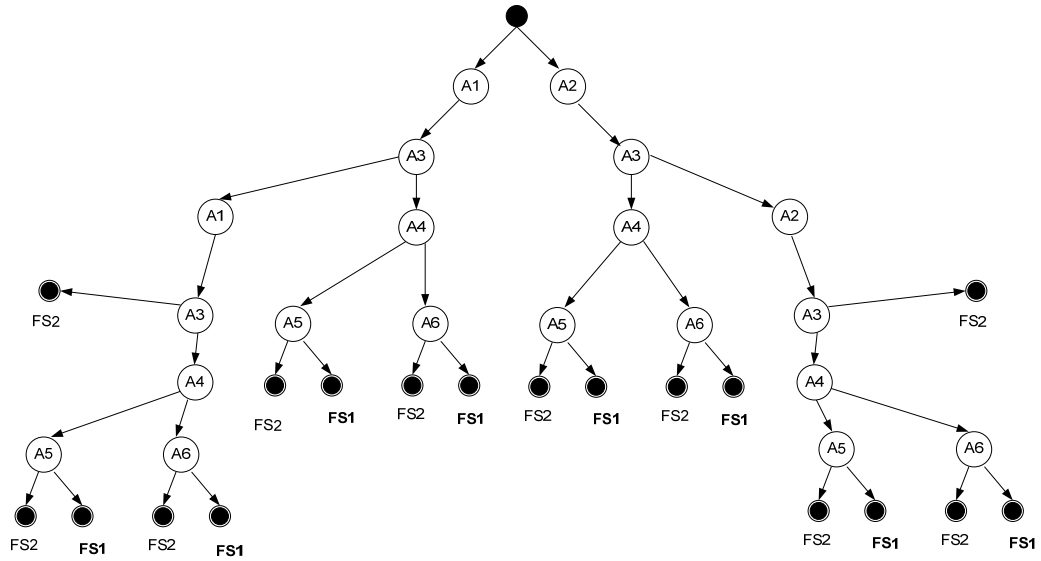


Figure 5-8: IFG of GUARD Receiver Opening Scenario Use Case

One generated test scenario for the V/UHF T/Rx Opening Scenario Use Case and one for the GUARD Receiver Opening Scenario Use Case by considering the above discussed situations are given in Table 5-8 and Table 5-9. If the IFG of V/UHF T/Rx System Opening Scenario is analyzed, it will be found that there should be 77 distinct test scenarios for full coverage of the IFG. But, for this use case, by accounting the previous discussions, only 5 test scenarios is generated. When the test scenarios are examined, it is seen that all possible states are covered at least once.

Examining the test scenario given in Table 5-8, the scenario corresponds to the path A1-A2-A6-A7-A8-A9-FS1 of IFG given in Figure 5-7. As mentioned before, scenarios ending with FS2 are not generated as test scenarios. But, it is seen that in the column of “Faulty Output End Scenario”, in TC1, TC2 and TC3, the conditions for transition to faulty final state FS2 are given. As it is discussed before, a scenario which ends with FS2 cannot be executed every time, except when the unique faults that enable the transition are occurred during the execution. The addition of corresponding column will be sufficient to cover faulty

outputs and direct transition to FS2 final states. This addition also dramatically decreases the test scenario number.

If the IFG given in Figure 5-8 and one of its scenario given in Table 5-9 are compared, it can be determined that the scenario corresponding to the path A1-A3-A1-A3-A4-A5-FS1. Again, in test scenario, the faulty outputs that may occur are also stated. But, it should be noted that, in the expected output of TC2 row, the faulty output that leads to the repetition of A1 and A3 instead of the output that should lead scenario directly to successful final state FS1. This is true for the path given above, but the question is what if the expected “faulty” output will not occur while the execution of this test scenario. In theory, the scenario will be unsuccessful and system will fail the corresponding scenario. But, for the work done in this thesis work, the paths other than main success scenarios mainly cover the faulty states that the system may not recover from by the actions of the tester. Actually these states are the final states that the test scenario should be halted and the system should be fixed for retesting. Accounting the above discussed situations for the scenarios generated for a use case, it is reasonable to execute first the main success scenarios, than to execute the scenarios that repeat some actions because of some faulty outputs in at least one of their actions.

Noting another decision made during test case generation process, some variations and extensions that are occurred in a use case are not covered in test cases if the corresponding variations are going to be tested in the previously generated test scenarios. Giving an example, VHF and UHF operation scenarios of V/UHF T/Rx system is very similar to each other, nearly replication of each other except working frequencies, and the extensions that can be occurred in both operation modes are not covered in the test scenarios generated for UHF operation. Making it more specific, if enabling and disabling is working well for VHF mode, than it will work well for UHF and if it is not working correctly, it should be fixed and retested before going on with testing scenarios generated for UHF operation mode.

Table 5-8: A Test Scenario for V/UHF T/Rx System Opening Scenario Use Case

V/UHF T/Rx System Opening Scenario (TS1)					
S1					
Use Case Name					
Scenario No					
Test Case #	Pre-Conditions	Input	Summary	Expected Output	Faulty Output End Test Scenario
TC1	The system must be shut down. The system must be connected to power supply appropriately. (220 VAC ± 10 % 50 Hz ± 10 % Input Voltage.) The MODE switch of RCU in the CU should be in OFF position. The GÜÇ, RF ANAHTAR and GÜÇ YÜKSELTECİ switches of CU must be in OFF position.	GÜÇ Switch=ON	Operator powers the CU by switching on the "GÜÇ" switch.	"AÇIK" LED is ON. "AÇIK" LED is not ON.	"AÇIK" LED is not ON.
TC2	"AÇIK" LED is ON.	SES KONTROL Switch = DAHİLİ KONTROL Switch = DAHİLİ	Operator switches both "SES KONTROL" and "KONTROL" switches to "DAHİLİ" positions.	-	-
TC3	-	MODE=T/R	Operator switches RCU to T/R and RCU opens. BITTEST is started. ARIZA LED becomes ON. BITTEST is ended. ARIZA LED is OFF and "R/T OK" message appears on RCU display.	RCU opens. BITTEST starts. ARIZA LED becomes ON. After BITTEST, ARIZA LED is OFF "R/T OK" message appears on RCU display.	RCU does not open OR BITTEST does not start OR ARIZA LED stays OFF OR After BITTEST, ARIZA does not become OFF OR After BITTEST "R/T BAD ###" message appears on RCU screen.
TC4	ARIZA LED is OFF.	Frequency=140 MHz	Operator sets the frequency to 140.000	RCU displays 140.000	-
TC5	-	RF ANAHTAR=ON	Operator switches RF ANAHTAR switch to ON position. RFSU and Filter is powered. Filter Frequency is adjusted.	Filter Frequency = 140 MHz HAZIR LED=ON	Filter Frequency does not equal to 140 MHz.
TC6	-	GÜÇ YÜKSELTECİ=ON	Operator switches GÜÇ YÜKSELTECİ switch to ON position. UHF amplifier is powered.	Amplifier is powered. HAZIR LED=ON	Amplifier is not powered or HAZIR LED is not ON.

Table 5-9: A Test Scenario for GUARD Receiver Opening Scenario Use Case

GUARD Receiver Opening Scenario Use Case						
Use Case Name	S1					
Scenario No	S1					
Test Case #	Pre-Conditions	Input	Summary	Expected Output	Faulty Output End Test Scenario	
TC1	The system must be shut down. The system must be connected to power supply appropriately. (220 V AC $\pm$ 10 % 50 Hz $\pm$ 10 % Input Voltage.) The GÜÇ switch of GUARD Receiver must be in OFF position.	KONTROL Switch = DAHİLİ	Operator switches "KONTROL" switch to "DAHİLİ" positions.	-	-	-
TC2	"AÇIK" LED is OFF.	GÜÇ Switch=ON	Operator powers the CU by switching on the "GÜÇ" switch.	"AÇIK" LED is ON. *All LEDs on GUARD RECEIVER become momentarily ON. *BITTEST starts and ARIZA LED blinks. *BITTEST ends and ARIZA LED becomes ON.	*"AÇIK" LED is not ON OR *All LEDs on GUARD Receiver do not become momentarily ON OR *BITTEST does not START OR *ARIZA LED does not blink.	
TC3	The system must be shut down. The system must be connected to power supply appropriately. (220 V AC $\pm$ 10 % 50 Hz $\pm$ 10 % Input Voltage.) The GÜÇ switch of GUARD Receiver must be in OFF position.	KONTROL Switch = DAHİLİ	Operator switches "KONTROL" switch to "DAHİLİ" positions.	-	-	-
TC4	ARIZA LED is OFF.	GÜÇ Switch=ON	Operator powers the CU by switching on the "GÜÇ" switch.	"AÇIK" LED is ON. *All LEDs on GUARD RECEIVER become momentarily ON. *BITTEST starts and ARIZA LED blinks. *BITTEST ends and ARIZA LED becomes OFF.	*"AÇIK" LED is not ON OR *All LEDs on GUARD Receiver do not become momentarily ON OR *BITTEST does not START OR *ARIZA LED does not blink.	
TC5	-	KONTROL Switch = DAHİLİ	Operator switches "KONTROL" switch to "DAHİLİ" position.	-	-	-
TC6	ARIZA LED must be OFF.	BAND Switch=VHF	Operator switches BAND switch to VHF position.	A BIP sound is heard. VHF LED is ON and UHF LED is OFF	No BIP sound. VHF LED is OFF OR UHF LED is ON OR Both UHF and VHF LEDs are ON OR Both UHF and VHF LEDs are OFF	

## **5.5. TESTING OF SYSTEMS**

Until now, the systems are modeled using UBST and generation of test cases and scenarios driven from these models are introduced. But, having test cases and scenarios is not adequate to create an effective testing procedure. Test planning phase is as important as test case generation considering whole system testing process.

Test planning process applied in this thesis work only covers the preparation of RTM, identification of verification methods, preparation of testing environment, planning the execution sequence and appointing special fail, success or “pass with faults” cases that the tester should obey while executing the test scenarios.

The verification of a system is carried out by applying a procedure which is created for the execution of verification methods like inspection, analysis, testing or combination of these methods. In the beginning of this thesis work, considering the previous testing efforts of the corresponding systems, the verification methods of the system requirements are determined as it is discussed before this chapter. Some metrics are gathered by using the previous test efforts’ records and these metrics represent the whole system verification effectiveness of corresponding systems. But, the UBST defined in Chapter 3 and applied in this chapter is developed for generating functional test cases. So, any inspection or analysis methods mapped to the requirements are not going to be verified with this process. Because of this situation, the results of verification steps of requirements which are mapped to analysis or inspection methods are gathered from previous testing efforts. Because of unchanged electrical and mechanical design of corresponding systems, these results can be used with no concern; they will all give the same result on the systems under test.

Test planning steps that are carried out before executing the test scenarios are presented below.



### **5.5.1. RTM**

RTM is actually created in order to be sure that each requirement that was marked as “Test” for its verification method will be verified with at least one test case.

Furthermore, this matrix will be used later on this thesis work in order to make some measurements that will be needed to discuss the effectiveness of the system testing processes of systems under test.

### **5.5.2. Defined Rules for Test Case Outputs**

Before executing test scenarios, some rules for test cases which define their “success”, “fail” or “pass w/fault” criteria. If the examples given in Table 5-8 and Table 5-9 are examined, the expected and faulty output conditions are given. But, the question how the tester should behave if he/she comes across with a faulty output should be answered. A set of rules defined for test scenarios are given below:

- R1: If tester ends up with faulty output in a test case and faulty condition does not affect the next test case’s user input and makes it possible to enter the input, whether if preconditions are not supplied, tester will continue to execute the test case and mark the faulty test case as “pass w/ faults”.
- R2: If the faulty condition affects the next test case’s user input and makes it impossible for user to enter the input, the test case and test scenario will be marked as “fail”.
- R3: If the test scenario ends with no faults, the test scenario will be marked as “success”.

These rules makes it easier for tester to give the necessary decisions while testing the systems and give the ability to try more input combinations whether there exist faults in system. By this way, it becomes possible to find another faults which have no relations to the faults found before.

### 5.5.3. Execution Sequence

The planning of execution sequence of testing scenarios is becoming very important in this thesis work. Because, the test scenarios include fail scenarios and scenarios of variations. The planning of execution sequence of test scenarios is done by considering the following rules:

- R4: The main success scenario of a use case (S1 of each Test Scenario (TS) in this thesis work) should be executed firstly.
- R5: If main success scenario of a use case is successful, no major faults is occurred which are unable to transmit and receive for V/UHF Transceiver and unable to receive for GUARD Receiver, the unsuccessful test scenarios should not be executed, the variation scenarios should be executed.
- R6: If main success scenario (S1) fails, the unsuccessful test scenarios should be executed in sequence considering their scenario number. If these scenarios also fail, variation scenarios should not be executed.
- R7: If main success scenario (S1) fails, the unsuccessful test scenarios should be executed in sequence considering their scenario number. If these scenarios successfully finalize, variations should be executed.

These rules enable faster testing and prevent tester from executing unnecessary scenarios which may end up with meaningless results. By applying these rules, the tester can verify the system without wasting time.

The testing scenarios of the use cases should be executed in an order considering the test set number which is given in the format TSx where x is the number of sequence number of test set. Giving an example, for V/UHF opening scenario test scenarios which are given in APPENDIX C should be executed in the orders given below considering the rules given above:

- For R4 and R5: TS1-S1 → TS1-S3 → TS1-S4 → TS1-S5
- For R4 and R6: TS1-S1 → TS1-S2
- For R4 and R7: TS1-S1 → TS1-S2 → TS1-S3 → TS1-S4 → TS1-S5

Here TS1-S1 is the main success scenario, TS1-S2 is the fail scenario and TS1-S3/S4/S5 are variation scenarios.

Until now, the way of testing work which was handled for this thesis work is discussed. The defined rules which were taking into account during testing of the systems are introduced and limitations of them are given. The results gathered from the testing process of V/UHF T/Rx System and GUARD Receiver and the metrics calculated from these results are going to be presented in the following parts of this document.

## 5.6. TESTING RESULTS

In this part of the thesis, the results and numerical measures gathered during and after the system testing process will be presented. For the V/UHF T/Rx System, 33 test scenarios and for the GUARD Receiver, 8 test scenarios are generated by the UBST process. 28 of the V/UHF T/Rx System and the entire GUARD Receiver test scenarios are executed by considering the rules mentioned in the previous part. Inspection on the RTM leads to the results in Table 5-10.

Table 5-10: Requirements by Verification Methods

	V/UHF T/Rx System	GUARD Receiver
<b>Requirements Verified by Testing</b>	31	64
<b>Requirements Verified by Analysis</b>	44	25
<b>Requirement Verified by Inspection</b>	32	17
<b>Requirements that are not covered by testing</b>	4	2
<b>Requirements that are not verifiable</b>	1	1
<b>Total Requirements</b>	99	89

Investigating Table 5-10, for about 35 % of the V/UHF T/Rx System requirements are verified by testing where this percentage increases to 67 % for GUARD Receiver system requirements. The ambiguous requirements mentioned in Chapter 4 have been removed, because they are rewritten and organized to be verifiable. There remain only two requirements that are not verifiable. These two requirements actually give information and present definitions of some concepts. Another note about the table is that the total number of requirements does not match the sum of the first five rows. The existence of requirements which are verified by more than one verification methods causes this discrepancy.

The testing process is executed for the verification of the requirements which have “testing” as their verification method.

During the testing process, 12 unique faults are found for the V/UHF T/Rx System. Most of these faults have occurred more than once during the testing process. These faults can be classified as “not critical” meaning they do not affect the critical functions, namely transmission and receive operation of the V/UHF T/Rx system. These faults affect the verification process of 20 requirements of the V/UHF T/Rx System and these requirements are marked as “FAILED”.

Investigating testing process of the GUARD Receiver, 15 faults are found during testing process. These 15 faults do not affect the critical operation of GUARD Receiver which is receiving from GUARD Frequencies. But, these faults lead to failure of 37 requirements of the GUARD Receiver.

As it is mentioned before, the V/UHF T/Rx System and the GUARD Receiver are operationally in use in Turkish naval vessels. By investigating and using the previous fault reports which are also used in CHAPTER 4Chapter 4 and comparing them with the faults that are observed during the testing process of UBST, it can be estimated whether there will be any faults and what they will be. In this way, the fault finding performance of UBST can be determined as presented in Table 5-11:

Table 5-11: Number of Defects Found by UBST and Prediction of Future Faults

	<b># Defects/Faults Found During Testing (UBST)</b>	<b>Prediction of # Defects/Faults Found After Testing</b>
<b>V/UHF T/Rx System</b>	12	2
<b>Guard Receiver</b>	15	1

The prediction in Table 5-11 is based on the comparison of the faults found by executing the test scenarios of UBST and previous testing efforts and examining the faults that are not found during testing process based on the UBST. Table 5-9 and Table 5-11 present the necessary data to calculate the testing metrics given in part 2.3.1 for the testing process of UBST. Calculation of the corresponding metrics based on the data gathered in equations (2-1) to (2-4) is presented below:

For V/UHF-2 Transceiver System,

$$Test\ Coverage = \frac{35}{99} \times 100 = 35\% \quad (5-1)$$

$$Test\ Effectiveness = \frac{12}{14} \times 100 = 86\% \quad (5-2)$$

$$Test\ Effort = \frac{12}{14} \times 100 = 86\% \quad (5-3)$$

For Guard Receiver,

$$Test\ Coverage = \frac{64}{89} \times 100 = 72\% \quad (5-4)$$

$$Test\ Effectiveness = \frac{15}{16} \times 100 = 94\% \quad (5-5)$$

$$Test\ Effort = \frac{15}{15} \times 100 = 100\% \quad (5-6)$$

In addition, by using the number of requirements that have “test” as their verification method, the number of test cases to test these requirements and number of links from these test cases to requirements, we can obtain the needed information for the analysis of test complexity of the generated testing process. In Table 5-12 some metrics that are needed for the application of the analysis mentioned in 2.3.1.4 and [28] are given.

Table 5-12: Number of Test Cases, Links and Requirements and Their Ratios

	V/UHF T/Rx System	GUARD Receiver
<b># of Requirement (R) Verification Method=Test</b>	35	65
<b># of Test Cases (T)</b>	214	32
<b># of Links (From Test Cases to Requirements) (L)</b>	109	107
<b>L/R</b>	3.11	1.64
<b>L/T</b>	0.51	3.34
<b>R/T</b>	0.16	2.03

## 5.7. COMPARISON OF PREVIOUS TESTING PROCESS AND UBST PROCESS

As it is discussed before, previous testing effort of the V/UHF T/Rx System and the GUARD Receiver System does not have a methodological basis. The previous testing process was conducted by engineers who have experience in V/UHF transceivers and knowledge about their operational uses and possible user needs. UBST Process was developed, test cases were generated, testing rules defined and test scenarios were conducted by the author of the present thesis. The basis of UBST and all other UML-based approaches are the clear and detailed use cases which are created considering the operational use of the systems. After creation of

use cases, every intermediate step until the generation of test cases has been handled methodologically by applying the UBST process. After the application of this methodological approach, only planning issues of generated test scenarios were handled. On the contrary, in the previous testing effort, only the requirements on SRS document of V/UHF T/Rx System and GUARD Receiver had been tried to be confirmed one by one.

Metrics gathered from both the previous testing efforts and the testing process that is developed based on UBST are presented in the Table 5-13 for ease of comparison:

Table 5-13: Previous Test Metrics vs. UBST Process Test Metrics

	<b>Previous V/UHF T/Rx System (%)</b>	<b>UBST V/UHF T/Rx System (%)</b>	<b>Previous GUARD Receiver (%)</b>	<b>UBST GUARD Receiver (%)</b>
<b>Test Coverage</b>	37	35	64	72
<b>Test Effectiveness</b>	61	86	50	94
<b>Test Effort</b>	80	86	100	100

Table 5-13 shows that while some properties of testing process have improved after application of the UBST process, some have worsened and some have not changed. Below, these properties are discussed one by one.

Testing coverage of the previous testing effort is found higher when it is compared to the corresponding metric value of UBST process for V/UHF T/Rx System. This small difference is caused by the previous testing effort's characteristics. As it was mentioned before, previous testing effort was based on a "Test Everything Possible" approach and the verification methods of requirements were decided during testing processes. This *ad hoc* approach attempted to test every requirement which was decided as testable during the testing process. Because of this approach, the previous testing process covers more requirements when compared to UBST process for V/UHF T/Rx System. It should also be

noted that UBST process for V/UHF T/Rx System covered fewer requirements but ran a higher number of tests than the previous testing effort.

Unlike the V/UHF T/Rx System, test coverage is higher for UBST process for the GUARD Receiver. This is because of the unplanned previous testing effort. Remembering again, the verification methods of requirements were defined just before the verification of requirements and these decisions were not revised. The UBST process requires the definition of verification methods of requirements to be one of its early steps and after generation of test scenarios and test cases, links between test cases and requirements are generated. So, during the application of UBST process, the requirements were investigated deeply and the ones that could be verified by testing were defined. Moreover, these definitions are controlled more than once and revised. Early and controlled definition of verification methods of requirements leads to have more requirements that have “test” as their verification methods. The increase in test coverage metric for GUARD Receiver arises from the situation described in this paragraph.

Covering more requirements within a testing process does not mean that it is an effective testing process [27]. It is seen in Table 5-13 that the test effectiveness metrics of UBST process are much higher than the previous testing effort’s for both V/UHF T/Rx System and GUARD Receiver. This great difference arose from more than one characteristics of UBST process which are presented below:

- UBST process includes the step of revising of requirement documents in which ambiguous requirements, unsuitable requirements and requirement which will cause possible misunderstandings between customer and developer are eliminated. This step was not covered for the previous testing process. By this step of UBST process, the developer has a clear requirements specification that will be used for both creating use cases and deciding acceptance criteria.



- The methodological approach of UBST process brings the advantages of scenario-driven approach of system testing. By generating and executing operational test scenarios, the possibility of finding operational faults that were missed by previous testing effort is increased. By this way, UBST process discovered the faults that the previous testing process could not detect.
- UBST process found many faults that the previous testing process missed. Moreover, the operational boundaries of both V/UHF T/Rx System and GUARD Receiver were also tested in UBST process and one new fault had been discovered in an operational boundary which is never encountered before.

Test effort metric of UBST process is higher when compared to previous testing effort's. This result is interesting as the system under test is in its final hardware configuration. Just the embedded software version that was used during previous testing effort is reinstalled on system and the tests are executed. So, it is expected to have the same test effort value for V/UHF T/Rx System considering all software originated faults are fixed as it is for GUARD Receiver. But, the difference between UBST and previous testing process is arisen from the increased effectiveness of testing process which increased the fault finding capability of testing process. The number of unresolved deficiencies is the same for UBST and previous testing process, but the amount of faults found makes the difference. The test effort metric of both UBST process and previous testing process is 100 %, which means that all the faults which were found during testing processes are fixed.

The ratios of R/T, L/R and L/T cannot be called as metrics when they are compared to the ones discussed above. But according to [28], these ratios give information about the complexity of the testing process to the developer.

The R/T value is equal to 0.16 for V/UHF T/Rx System and to 2.03 for GUARD Receiver. This value seems normal for GUARD Receiver; one test case verifies two requirements on average. But for V/UHF T/Rx System; there are possible risks of over testing and duplication of test cases. If the test scenarios of V/UHF T/Rx System are investigated again, the duplication of test scenario steps can be seen. Nevertheless, most of them have different previous steps which can give different outputs to the repeated test step. On the contrary, some of scenarios can be merged, duplicated actions can be executed once and the steps after variations can be executed one after another. Unfortunately, UBST process does not control the duplications of tests and one of the causes of duplications is variations. But, variations helps to increase the test effectiveness, so duplications are tolerable for making the whole testing process more effective.

Both L/R values for V/UHF T/Rx System and GUARD Receiver are greater than 1. If this value was lesser than 1, the existence of requirements which are not verified with a test case would be certain. The UBST process does not seem problematic in case of L/R value. But, this value does not guarantee that all of the requirements are linked to a test case, it only gives a general idea about testing process. As Table 5-10 shows, both UBST processes for V/UHF T/Rx System and GUARD Receiver has uncovered requirements which should be verified by testing.

The L/T value measures the complexity of the testing process. In UBST processes, L/T value is not considered as high to call these processes as complex. But, an interesting value for L/T which equals to 0.59 is acquired. This value is occurred because of variations. For example, the illumination of “GONDERMEDE” LED after pressing Push-to-Talk (PTT) can be verified by various test cases. The links of these various test cases are counted as 1. Because, when one of them verifies the requirement, it is unnecessary and costly to verify it again. So, the L/T value for V/UHF T/Rx System is unexpected, but this value of L/T is understandable when the above action is considered.

In conclusion, the UBST process brings a methodological approach to system testing process. The most important benefit of this methodological approach is increasing test effectiveness. By increasing test effectiveness, the risk of late realization of problems decreases and possible cost problems will be eliminated by finding more system faults. But making things methodological always brings documentation load which increases manpower costs. So, this process should be applied to complex and large systems which have many inner-system and inter-system interfaces. But, if the UBST process is improved and system designs begin to be standardized based on UML and SysML, the UBST process will become easy to apply and it will be automated to some point. This will result in increased developer confidence about providing high testing effectiveness and ease of generating test cases. Moreover, by standardizing this process and generating straightforward, understandable and clear test cases, the testing process is made easier to conduct and even technicians who have limited knowledge about system functionality can execute the test cases. Unlike UBST, for previous testing processes, testing could be conducted only by an engineer who has experience about system or by the supervision of an experienced engineer.

## **CHAPTER 6**

### **CONCLUSION**

This chapter provides the summary and the concluding remarks of this study. Furthermore, the possible future studies are also suggested.

#### **6.1. CONCLUSIONS**

System testing and system acceptance testing have an important role in system verification and validation in which system and user requirements are verified. Moreover, these phases are not standalone phases and have relations with requirements management, project planning and system design phases.

In this thesis work, system testing and acceptance testing approaches and methodologies for software and hardware-based systems are investigated.

There are lots of work about software system testing approaches and methodologies in literature. Many approaches attribute high importance to requirements reviewing for system testing phase. Moreover, determining verification methods for requirements and acceptance criteria are also stated as important in literature.

In addition, test case generation methods for software systems are investigated. It should be noted that for hardware-based systems, there is no methodology found about test case generation during literature review. Generally, the test case

generation methods begin with user requirements and try to design the systems in a standardized model. Some of these standardized models are based on building FSM, and UML models which are sequence diagrams, state diagrams and activity diagrams.

UML approaches to system testing process and SysML extension of UML are also investigated. Application of UML models to system design and test case generation methods from these models and automation of these methods are popular research areas in literature. Applicability of these various UML-based test case generation processes is evaluated. Some of the methods focus on structural behavior or both structural and functional behavior and these properties of these methods make them unnecessarily complicated for the scope of this thesis work. The method using activity diagrams as basis diagram is chosen because of its scope “behavioral system testing”. This method differs from the other methods in suitability of its scope to the scope of this thesis work and ease of application.

The chosen method creates activity diagrams using textual use cases. Then intermediate diagrams of IFD and IFG are created before the generation of test cases. These diagrams, especially IFG is created for making automation of test case generation possible. But, in this thesis work, automation process is not considered.

Furthermore, metrics that can be used for measuring testing effectiveness of functional system testing processes are presented. Measures which are specifying the characteristics of testing process are mentioned. Moreover, some methods for increasing testing effectiveness are also stated in this thesis work.

After the literature review, a system testing process called UBST process is developed by using the chosen UML-based test case generation approach. In developed UBST process, preliminary work before modeling the system, system modeling and test case generation process are presented in detail.

In addition, system testing and acceptance testing process of SEDNC is investigated and problems of these processes are presented by comparing the processes to the publications in literature and interviewing colleagues and experts. The most important problems are stated as requirements that are not detailed, lack of operational scenarios in system testing procedures, lack of test case generation methodologies and planning issues.

The previous system testing processes of V/UHF T/Rx System and GUARD Receiver are investigated and details of these processes are given. Records that are gathered during the testing processes are presented. These records are used to calculate metrics of test coverage, test effectiveness and test effort.

After gathering the metric calculations of previous testing process, UBST process is applied to the V/UHF T/Rx System and GUARD Receiver. SRS document of these systems is reviewed and revised in order to handle ambiguous requirements and requirements that are not verifiable. Having understandable and clear requirements is necessary for determining acceptance criteria for requirements that are going to be verified. Moreover, making these requirements understandable and measurable helps the developer to design the product which really meets the user requirements. In this thesis work, it helped the developer to model the systems and create correct use cases which is the basis of test case generation process.

Use case diagrams and textual use cases are created by using the requirements and considering the operational scenarios. Textual use cases which are written in a scenario-driven approach are modeled as flow of these scenario activities by using activity diagrams. These activity diagrams are converted to IFDs which are a simplified version of activity diagrams and interested only in actions and responses between system and user, ignoring inter-system actions. At the end of the process, loop-free IFGs are generated in a tree format and executed using depth-first algorithm which is resulting in generation of test scenarios in a scenario-driven approach.

The generated test scenarios include success and fail scenarios. In this step, the need of test scenario prioritization and planning issues steps forward. A decision making system is developed to be used during executing tests for deciding both what to do next during execution of test cases and the scenario which will be executed next. Some rules and instructions included in the decision system are taken into account for a reasonable and effective sequence of test scenarios.

After executing the test scenarios, the needed data to compare the UBST process and previous test process is gathered. Considering the main scope of testing, which is finding faults, the UBST process is reasonably improves the fault finding capability of system testing process. The difference between test effectiveness metric values of compared processes proves this argument.

Test coverage metric which measures the number of requirements tested by testing process shows variation between GUARD Receiver and V/UHF T/Rx System. The test coverage metric of GUARD Receiver is improved while the value of V/UHF T/Rx System is dropped. A good testing process should absolutely cover requirements as much as it can, but test coverage is not very effective on fault finding capability considering test effectiveness metric results. This result shows that test coverage metric is not related directly with test effectiveness.

Test effort metric shows how many of the faults which are found during testing process are fixed. Test effort of V/UHF T/Rx System is improved because of the increase in the number of faults found during the UBST process. These faults are the faults that were not found during testing process and appeared after the tests were conducted. So these faults were not taken into consideration while calculating the test effort metric of previous testing process of V/UHF T/Rx system. Test effort is increased by UBST process with the increase in test effectiveness.

Some measurements are made instead of metrics and it is seen that improvements should be made for a more effective testing process. But, considering previous testing effort, UBST process improved the effectiveness of behavioral system testing applied to V/UHF T/Rx System and GUARD Receiver.

In short, the number of defects found during system testing process (test effectiveness) is increased and the number of faults fixed which are found during system testing process (test effort) is increased naturally by the increase in the number of detected defects whereas the number of requirements verified by system testing process (test coverage) is not changed meaningfully. It should be noted again that while test effort is directly related to test effectiveness metric, test coverage metric is not related to either test effectiveness or test effort metrics.

This thesis study has shown that by a methodological approach, system testing becomes an effective step for finding faults of systems before system delivery that have several varying interfaces with other systems and equipments. Moreover, by applying the UBST process, it is realized that system testing should be more than just verification of the system requirements one by one. This realization should be taken seriously and at least a methodological approach for generating testing scenarios should be integrated to system and acceptance testing processes of SEDNC which are conducted for the systems that are designed by subcontractors and company that author is employed. This integration may decrease the cost of late realization of problems by decreasing the cost of manpower spent for the solutions of problems which is actually not measured by the company for author's department.

The work on this thesis work was begun for improving the system and acceptance testing of a whole communications system which is actually a system of systems. The department in which the author is employed is the developer and the Dz.K.K.1iđı is the user for the communications system of a naval platform. But, because of the difficulty of bringing whole communications system together and handling system testing process of a very large and varying system, and the lack



of data from the previous testing efforts of the whole system, the author has focused on subsystems of V/UHF T/Rx System and GUARD Receiver as the systems under test which can be easily tested and have data about the previous testing efforts. The equipments subject to test in this thesis work are manufactured in author's employer company. The design departments are in the role of subcontractor for the author's department if the big picture is investigated. In this thesis work, this viewpoint was the basis of the idea of investigating and improving the system testing process of V/UHF T/Rx System and GUARD Receiver. During the development of UBST process, the author has put himself in the place of the developer and has modeled the systems from a functional viewpoint and in a scenario-driven approach. Narrowing down the application scope of the UBST process will not affect its applicability to larger systems and system of systems such as communications system of a naval ship, from the viewpoint of UBST process. There will be no difference in modeling of the systems, generating test scenarios and testing of a system between a sub-system and system of systems except the complexity of the jobs that should be handled.

UBST process includes some requirements management steps which should not be in testing phases. However, as it is frequently mentioned in literature review, for effective testing of systems, user and system requirements should be clear, verifiable and should not be understood differently by developers and users. For an effective and non-problematic product lifecycle, the requirements management steps should be handled in project beginnings not at the beginning of system testing phase. Moreover, acceptance criteria, test methods and general RTM should be prepared in the beginning of the projects and these information should be supplements of TEMP document. These are also covered in the UBST process in this thesis work for creating a more effective testing process.

## **6.2. FUTURE WORK**

In this study, the main aim has been to demonstrate the improvement of the effectiveness of system testing process by using UML-based system test case generation methods. System development process and software development process have many similar phases. This thesis work can be extended by trying to adapt the UML-based software development phases from requirements management and modeling up to maintenance period to system development phases.

Furthermore, the present study only covers the testing part of whole system verification process. The verification steps with other methods (analysis, inspection, etc.) can also be handled in a methodological way.

Besides, as a future study, the automation of the UBST process can be achieved by specifying the inputs, outputs and acceptance criteria in a more formal format and by directing developer to make decisions while creating the flow models.

## REFERENCES

- [1] Freeman, H., "Software Testing", IEEE Instrumentation & Measurement Magazine, September 2002.
- [2] Yu, Y., Wu, F., "A Software Acceptance Testing Technique Based on Knowledge Accumulation", Ninth Great Lakes Symposium on VLSI Proceedings, 1999.
- [3] Scully, J.K., "The Hidden Crises in Test Effectiveness", AUTOTESTCON '98, IEEE System Readiness Technology Conference Proceedings, 1998.
- [4] Causevic, A., Sundmark, D., Punnekkat, S., "An Industrial Survey on Contemporary Aspects of Software Testing", 3<sup>rd</sup> International Conference on Software Testing, Verification and Validation, 2010.
- [5] Naik, K., Tripathy, P., "Software Testing and Quality Assurance", Wiley, 2008.
- [6] Smith, D., Russell, S., "How to measure effects of systems engineering on the outcomes of a project", Systems engineering test and evaluation conference pp. 1-13 1, 2008.
- [7] Faulconbridge, R.I., Ryan, M.J., "Managing Complex Technical Projects: A Systems Engineering Approach", Artech House, 2003.
- [8] Kossiakoff, A., Sweet, W.N., "Systems Engineering Principles and Practice", Wiley, 2003.

- [9] Technical Board of International Council on Systems Engineering (INCOSE), “Systems Engineering Handbook A “What to” Guide for All SE Practitioners”, INCOSE, INCOSE-TP-2003-016-02, Version 2a, 1 June 2004.
- [10] Department of Defense, “Systems Engineering Fundamentals”, Defense Acquisition University Press, January 2001.
- [11] Hsueh, M.C., “Large Complex System Test: Objectives & Approaches”, Proceedings of First IEEE Conference of Engineering of Complex Computer Systems, IEEE, 1995.
- [12] Beizer, B., “Black-Box Testing: Techniques for Functional Testing of Software and Systems”, Wiley, 1995.
- [13] Hsia, P., Gao, J., Samuel, J., Kung, D., Toyoshima, Y., Chen, Y., “Behaviour-Based Acceptance Testing of Software Systems: A Formal Scenario Approach”, Eighteenth Annual International Computer Software and Applications Conference Proceedings, 1994.
- [14] Fröhlich, P., Link, J., “Automated Test Cases Generation from Dynamic Models”, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, pp. 472-491 2000.
- [15] Heinecke, A., Brückmann, T., Griebe, T., Gruhn, V., “Generating Test Plans for Acceptance Tests from UML Activity Diagrams”, 17<sup>th</sup> IEEE International Conference and Workshops on Engineering of Computer-Based Systems, 2010.
- [16] Perry, W.E., “Effective Methods for Software Testing”, 3<sup>rd</sup> Edition, Wiley, 2006.
- [17] Riebisch, M., Philippow, I., Götze, M., “UML-Based Statistical Test Case Generation”, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, pp. 394-411, 2003.

- [18] Sarma, M., Kundu, D., Mall, R., “Automatic Test Case Generation from UML Sequence Diagrams” 15<sup>th</sup> International Conference on Advanced Computing and Communications, IEEE, 2007.
- [19] Sarma, M., Mall, R., “System Testing using UML Models”, 16<sup>th</sup> Asian Test Symposium, IEEE, 2007.
- [20] Kumari, M., Sharma, A., Kamboj, V., “Replacement of S/W Inspection with S/W Testing”, International Journal of Information Technology and Knowledge Management, July-December 2009, Volume 2, No.2, pp. 257-261, 2009.
- [21] Baresi, L., Pezze, M., “An Introduction to Software Testing”, Electronic Notes in Theoretical Computer Science 148 (2006), pp. 89-111, Elsevier, 2006.
- [22] Frederick, J., Jaggard, C., Paglione, M., Baldwin, C., “Verification and Validation Standards to Test and Evaluate New Complex Systems for the National Airspace System”, ITEA Journal 2009; 30: 277-287, June 2009.
- [23] Atkins, R., “Software contracts and the acceptance testing procedure”, Computer Law & Security Report, pp. 51-55, 2005.
- [24] Director of OT&E, “Policy for Application of Modeling and Simulation in support of OT&E”, Washington DC, January 24 1989.
- [25] Ramos III, J., “Modeling and Simulation (M&S) Issues in Operational Test and Evaluation (OT&E)”, Proceedings of the 1994 Winter Simulation Conference, 1994.
- [26] Lamancha, B.P., Usaola, M.P., Velthius, M.P., “Software Product Line Testing A Systematic Review”, ICSOFT 2009 - 4<sup>th</sup> International Conference on Software and Data Technologies, 2009.
- [27] Hutcheson, M.L., “Software Testing Fundamentals: Methods and Metrics”, John Wiley & Sons Inc., 2003.

- [28] Rosenberg, L.H., Hammer, T.F., Huffman, L.L., “Requirements, Testing, and Metrics”, 15th Annual Pacific Northwest Software Quality Conference, 1998.
- [29] Lang, P., Card, M., Saalwaechter, S., Godkin, T., “Application of Test Effectiveness in Spacecraft Testing”, Reliability and Maintainability Symposium, Proceedings, 1995.
- [30] Neogy, R., Dharan, H., “Measures of Test Effectiveness in a Communications Satellite Program”, IEEE Journal on Selected Areas In Communications, October 1986.
- [31] Williams, L., Snipes, W., Meneely, A., “On Increasing System Test Effectiveness through a Test Case Prioritization Model Using Static Metrics and System Failure Data”, Reliability Analysis of System Failure Data Workshop, Cambridge, UK, 2007.
- [32] Sneed, H.M., “Measuring the Effectiveness of Software Testing”, First International Workshop on Software Quality (SOQUA 2004) Proceedings, 2004.
- [33] Dustin, E., “Effective Software Testing”, Pearson, 2003.
- [34] Khan, S.u.R., Rehman, I.u., Malik, S.u.R., “The Impact of Test Case Reduction and Prioritization on Software Testing Effectiveness”, International Conference on Emerging Technologies, 2009.
- [35] Ronne, J.V., “Test suite minimization: An Empirical Investigation”, June 1999, Retrieved from URL: <http://www.cs.utsa.edu/~vonronne/pubs/jvronne-uhc-thesis.pdf>, 12/03/2010.
- [36] Harrold, M.J., Gupta, R., Soffa, M.L., “A Methodology for Controlling the Size of Test Suite”, ACM Transactions on Software Engineering and Methodology, ACM, 1993.

- [37] Holden, I., Dalton, D., “Improving Testing Efficiency using Cumulative Test Analysis”, Proceedings of the Testing: Academic & Industrial Conference – Practice and Research Techniques, 2006.
- [38] Willard, B., “UML for systems engineering”, Computer Standards & Interfaces, Volume 29, pp. 69-81, Elsevier, January 2007.
- [39] “OMG System Modeling Language (OMG SysML) Version 1.2”, <http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf>, 16/06/2010.
- [40] Heide, C., Hoover, R., “Optimizing Test Systems for Operational Test Benefits Using Parallel Test Capable Instruments”, IEEE Autotest Conference, 2008.
- [41] “Test and Evaluation Management Guide Fourth Edition”, The Defense Acquisition University Press, November 2001.
- [42] Lee, Y.H., Lee, B.G., Lee, J.C., Kim, Y.K., “Preparing Test and Evaluation Master Plan (TEMP) for the Korean CBTC System Development Project”, Proceedings of 2<sup>nd</sup> IEEE International Systems Conference, 2008.

## **APPENDIX**

### **USE CASES, BEHAVIORAL DIAGRAMS AND TEST SCENARIOS**

Supplied CD includes the Use Cases, Behavioral Diagrams and Test Scenarios which are generated during the application of UBST process on V/UHF T/Rx System and GUARD Receiver.