

AUTOMATION TESTING WITH UFT AND HC FACETS

Project report submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Nikita Pandey (171053)

UNDER

COGNIZANT



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

May 2021

TABLE OF CONTENTS

CAPTION	PAGE NO.
DECLARATION	i
ACKNOWLEDGEMENT	ii
LIST OF ACRONYMS AND ABBREVIATIONS	iii
LIST OF FIGURES	iv
ABSTRACT	vi
CHAPTER-1: FUNCTIONAL TESTING	1
1.1 Software Development LifeCycle (SDLC)	1
1.2 Software Testing	5
1.2.1 Test Levels	5
1.2.2 Testing Types	6
1.3 Black-Box Testing Techniques	7
1.3.1 Equivalence Partitioning	7
1.3.2 Boundary Value Analysis	7
1.3.3 Decision Table Testing	7
1.3.4 State Transition Testing	8
1.3.5 Error Guessing	8
1.4 White-Box Testing Techniques	8
1.4.1 Statement Coverage	8
1.4.2 Branch Coverage	8
1.4.3 Path Coverage	9
1.5 Agile Methodology	9
1.5.1 Agile Testing Principles	10
CHAPTER-2: DATASOURCE	11
2.1 SQL Using MySQL Database	11
2.1.1 Relational Database Management System	11
2.1.2 Data Definition Language (DDL)	12
2.1.3 Data Manipulation Language (DML)	13
2.1.4 Some SQL Queries	13
2.1.5 Database Design	13
2.1.6 Normalization	14
2.2 Documenting APIs Via JSON	14
2.3 Documenting APIs Via XML	15
CHAPTER-3: VBSCRIPTING	16
3.1 Features of VBScript	16

3.2 Uses	16
3.3 Some of the Hands-on done on VBScript	17
CHAPTER-4: UFT AUTOMATION	20
4.1 Advantages of UFT	20
4.2 UFT/QTP IDE	20
4.2.1 Add-In Manager	20
4.2.2 Start Page	21
4.2.3 Actions	22
4.2.4 Transactions	23
4.2.5 Checkpoints	24
4.2.6 Parameterization	25
4.2.7 Recording	27
4.2.8 Object Repository	28
4.2.9 Environment Variables	30
4.2.10 Functions	33
4.2.11 Datatables	34
4.2.12 Descriptive Programming	34
CHAPTER-5: FUTURE WORK	36
CHAPTER-6: CONCLUSION	37
REFERENCES	
PLAGIARISM REPORT	

DECLARATION

I hereby declare that the work reported in the B.Tech Project Report entitled “**Automation Testing with UFT and HC Facets**” submitted at **Jaypee University of Information Technology, Wagnaghat, India** is an authentic record of my training carried out under **Cognizant** during my training period. I have not submitted this work elsewhere for any other degree or diploma.

.....
NIKITA PANDEY
171053

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

.....

Date:

Head of the Department/Project Coordinator

ACKNOWLEDGEMENT

I take this opportunity to express my gratitude to all the Subject Matter Experts, for their professional guidance, motivating suggestions, help and support in successful completion of this training and would also like to thank Cognizant for providing us with world class learning.

The resources provided by Cognizant throughout the training are equally acknowledgeable. It was great privilege and honor to work and get trained under their guidance. I am also very thankful to our coach who has always assisted us by supplying the requisite help.

Nikita Pandey

171053

LIST OF ACRONYMS AND ABBREVIATIONS

UFT: Unified Functional Testing

SQL: Sequential Query Language

RDMS: Relational Database Management System

XML: eXtensible Markup Language

JSON: JavaScript Object Notation

VBScript: Visual Basic Script

SDLC: Software Development Life Cycle

DDL: Data Definition Language

DML: Data Manipulation Language

IDE: Integrated Development Environment

LIST OF FIGURES

CHAPTER 1:-

Figure 1.1: Software Development Lifecycle	1
Figure 1.2: Sequential Development Model	3
Figure 1.3: Waterfall Model	3
Figure 1.4: V- Model	4
Figure 1.5: Incremental and Iterative Model	4
Figure 1.6: Features of Software Testing	5
Figure 1.7: Test Levels	6
Figure 1.8: Agile Model	9

CHAPTER 2:-

Figure 2.1: RDBMS	12
Figure 2.2: JSON File	15
Figure 2.3: XML File	16

CHAPTER 4:-

Figure 4.1: Add-in Manager	21
Figure 4.2: Start Page	21
Figure 4.3: Types of Actions	22
Figure 4.4: Multiple Actions	23
Figure 4.5: Selecting Checkpoints	25
Figure 4.6: Parameterization	26
Figure 4.7: Parameters in Global Sheet	26
Figure 4.8: Record Settings	27
Figure 4.9: Settings dialogue box	27
Figure 4.10: Run Button	28
Figure 4.11: Object Repository	29
Figure 4.12: Object Repository Manager	30
Figure 4.13: Environment Variables	31
Figure 4.14: Accessing Environment Variables	32

Figure 4.15: User Defined Variables	33
Figure 4.16: Function Library	33
Figure 4.17: Datatables	34
Figure 4.18: Descriptive Programming	35

ABSTRACT

This learning program draws in with a thorough learning pathway, offering us a chance to cooperate with Subject Matter Experts, comprehend the professional workplace, and grooms ourselves. The all out learning adventure is formalized using grown-up learning principles, where basic reasoning and applying the capacities obtained are given more importance than sensible learning.

After going through this Automation Testing with UFT and HC Facets Learning Path, we will be able to:-

- Understand how Agile Implementation is done in Software Projects.
- Illustrate the different phases of the Functional Testing Life Cycle for a given business requirement.
- Comprehend SQL and perform fundamental Database activity utilizing the MySQL database and learn Database Design with Normalization.
- XML/JSON file creation and parsing using UFT automation tool and VBscript.
- Learn the fundamentals of VBScript concepts, which are required for UFT automation.
- Understand UFT Automation tool on latest version from scratch.
- Segment the application under automation test script into logical business components.
- Illustrate Automated Test Cases using UFT.
- Illustrate fundamentals of Claims processing system and Basic Concepts of Health Insurance and networks.
- Illustrate FACETS Overview and Group, Subgroup and Class/Plan Configuration, Enrollment, Member & Eligibility, Medical & Hospital Claim processing.

The complete learning program is divided into five milestones:-

- Milestone 1: Functional Testing
- Milestone 2: Data Source –SQL, XML & JSON
- Milestone 3: VB Script
- Milestone 4: UFT Automation
- Milestone 5: HC Facets

CHAPTER 1

FUNCTIONAL TESTING

1.1 Software Development Life Cycle (SDLC)

A Software Development Lifecycle [1] Model portrays the sorts of exercises performed at each stage in a product advancement undertaking, and how the exercises identify with each other logically and sequentially.

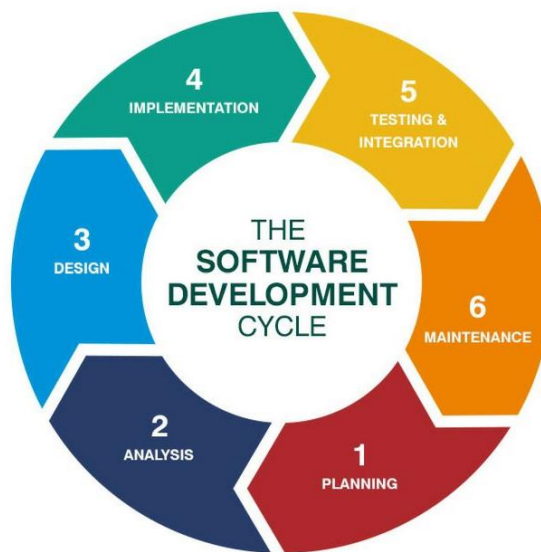


Figure 1.1: Software Development Lifecycle

The six total phases of Software development life cycle model are:-

- a) **Planning and analysis:** In this stage business prerequisites are analyzed and noted down. This stage is the fundamental focal point of the task chiefs and partners. Gatherings with chiefs, partners and clients are held to decide the necessities. From the client side, the pre requirements are collected and are checked for their correctness. Therefore, there requirements are integrated into the framework of the software successfully.
- b) **Design:** In this stage the framework and programming configuration is set up from the necessity particulars which were concentrated in the primary stage. How the framework of the software will look is determine by the design formulated and will also tell us about the

general functioning of the software. So for the next following stage this designed framework will help in the completion of the next stage of SDLC.

- c) **Implementation:** After getting the idea about how the general framework would look like, the work is divided into different sections. The coding start in this stage. So, it is the hub of designer. Looking from the programming point of view, this stage is the longest of all stages.
- d) **Testing:** The code that is created by the previous stage is tested. Requirements are tested whether they are getting fulfilled with the designed code or not. During this stage a wide range of utilitarian testing like unit testing, reconciliation testing, framework testing, acknowledgment testing are done just as non-useful testing are likewise done.
- e) **Deployment:** In this stage, the tried and tested code is sent to the client for their testing whether they are satisfied with the product or not. During this stage if the client find any error or is not satisfied with the code, the will contact the designing team for the required correction of the code. When those progressions are made or the bugs are fixed then the last arrangement will occur.
- f) **Maintenance:** In this stage, the client approves the product and if the during the usage of the software any issue pops up then it should be corrected at that very moment. This is what this stage does. This cycle where the consideration is taken for the created item is known as maintenance.

The two types of SDLC:-

- a) **Sequential SDLC:** In this SDLC [2] activities are completed one after the other. There is no overlap between the stages of SDLC. There is linear process of software creation. Well according to the rules there shouldn't be any overlap between the stages, but having a feedback will provide with better insights in making the software.

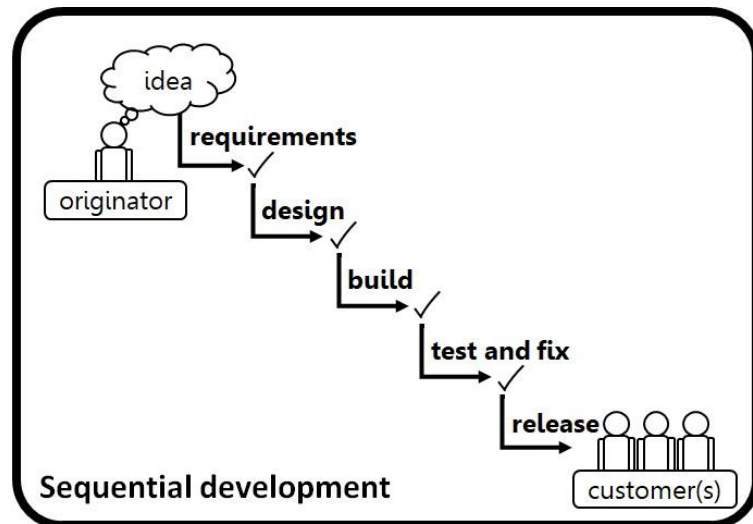


Figure 1.2: Sequential Development Model

- i. **Waterfall Model:** The stages or the activities of the cycle are completed one after the other without overlapping in waterfall model [3].

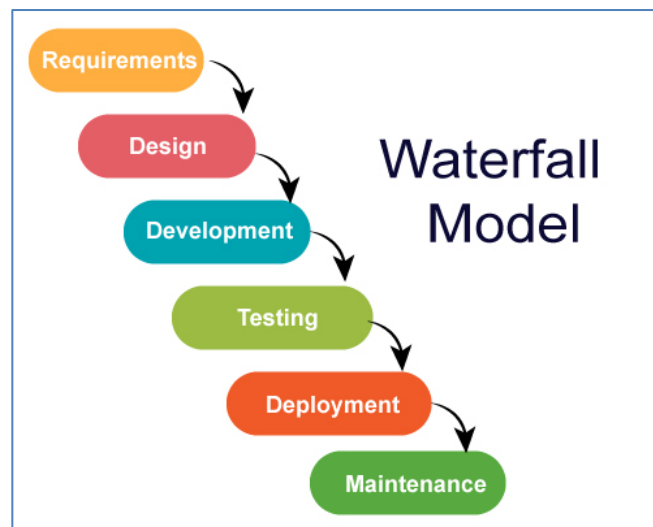


Figure 1.3: Waterfall Model

- ii. **V Model:** It is not like waterfall model [4]. The stages of this model called V model, stages are developed together side by side and feedback is collected at same time resulting in the early completion of the complete process.

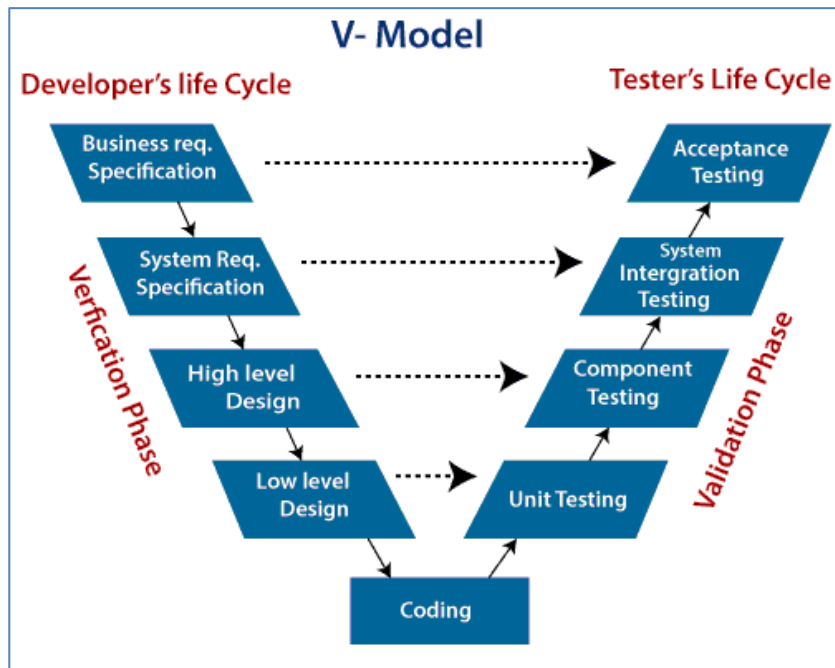


Figure 1.4: V- Model

b) **Iterative & Incremental SDLC:** In iterative and incremental type of SDLC, there are number short cycles with one complete development process [5].

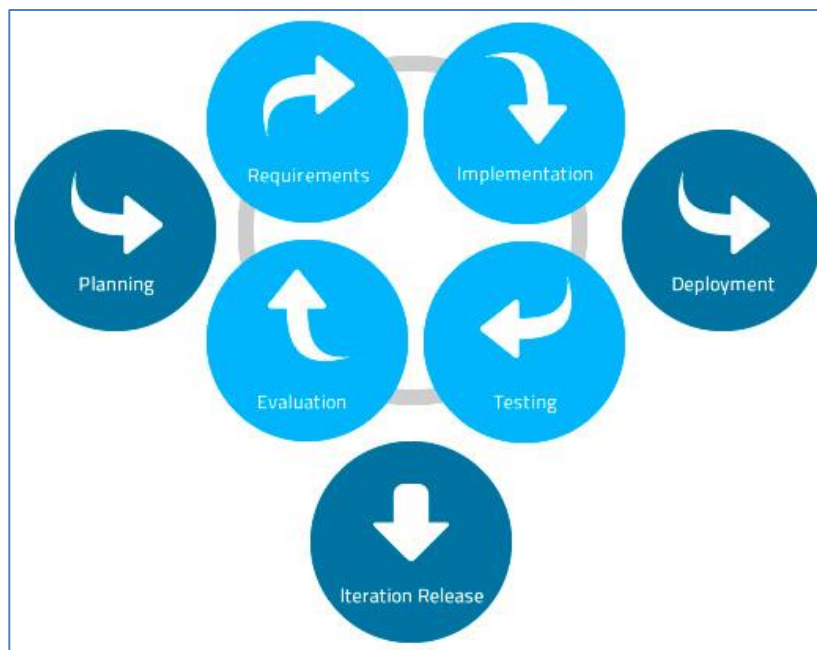


Figure 1.5: Incremental and Iterative Model

1.2 Software Testing

In software testing the software is tested for any error or failure [6]. The quality of the software is also checked and it reduces any risk present in the software. Software testing comprises many different activities and stages. Execution is one of them. Test execution is not the same as software testing.

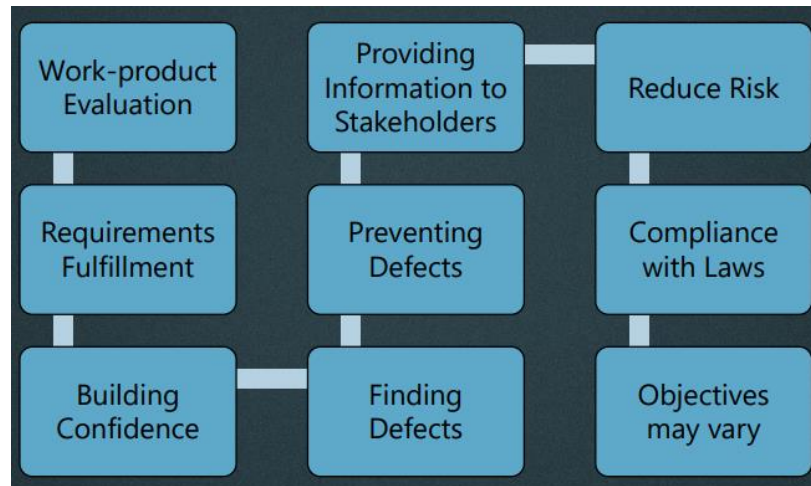


Figure 1.6: Features of Software Testing

1.2.1 Test Levels

The activities that are grouped together and are managed together are called test levels. Each and every level of test level is a complete test process [7]. Within the SDLC, test levels are related to two other activities also.

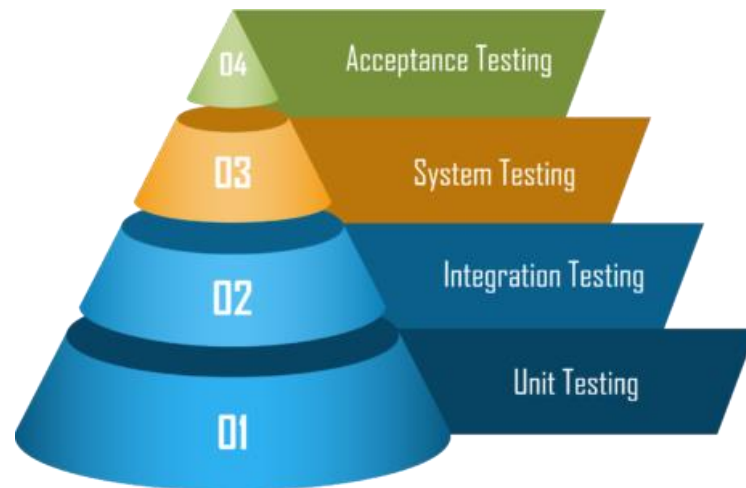


Figure 1.7: Test Levels

- a) **Unit Testing:** It is also called component testing or module testing. It mainly concentrates on the different units that are separately tested.
- b) **Integration Testing:** In this type of testing the main focus is on the relationship between the different units of the system.
- c) **System Testing:** This level of testing the complete system for the product is checked and verified whether it is meeting the requirements or not.
- d) **Acceptance Testing:** This type of testing is similar to a system. It also checks the functionality of the complete finished software.

1.2.2 Testing Types

- a) **Functional Testing:** This type of testing tests the functioning of the system and answers with yes or no.
- b) **Non-functional Testing:** Testing how the system performs. Hard to answer with Yes/No. Usually measured as a range.
- c) **Black-Box Testing:** Testing without knowing the internal structure of the system.
- d) **White-Box Testing:** Testing while monitoring the internal structure of the system.
- e) **Dynamic Testing:** Testing that includes executing the software.
- f) **Static Testing:** Testing that doesn't include executing the software.

- g) Retesting (Confirmation Testing):** Testing after debugging to ensure defects are fixed.
- h) Regression Testing:** Testing unchanged areas to ensure they are not affected by changes.
- i) Smoke Testing:** Testing main functionalities to ensure that the build is stable enough to continue testing.

1.3 Black-Box Testing Techniques

The internal paradigm of the complete system is unknown in Black box testing [8]. And input is given to the system and the output generated from the system is observed and noted down. This is used for predicting the output of the system how it behaves for different sets of input, the time, and related issues.

1.3.1 Equivalence Partitioning

All the possible inputs are divided into different classes or partitions. At the time of testing only one input from each partition is tested. Example partitioning users birth date into two groups under 18 and over 18. The tester can check for 1 value in the under 18 group and one value from the over 18 group.

1.3.2 Boundary Value Analysis

In this analysis the output of the system around the boundary values are tested. For example a field in software accepts values between 0 and 10. So in boundary value analysis focus will be on the boundary values which are -1 0, 10 and 11 check if the system is accepting the input values correctly.

1.3.3 Decision Table Testing

An output of a system is provided on the basis of a group of pre-conditions. In this testing the tester checks for these rules which are a set of requirements for conditions and observe the output of each rule and design a separate test case for each one of them.

1.3.4 State Transition Testing

When switching from one state to another some systems react in a significant way. For example when logging in into an account after a maximum number of attempts the user gets locked out of the account for some amount of time.

1.3.5 Error Guessing

In this testing the testers check for some of the common mistakes that the developers make during the development of the code. Example the testers check whether null values are getting rejected or not are, numeric fields accepting only numeric values or not and other mistakes.

1.4 White-Box Testing Techniques

In this type of testing technique the internal structure of the software is known [9]. Testing is done to check whether the complete internal structure behaves according to the requirements for a specific input and generates desirable output.

1.4.1 Statement coverage

According to the programming language a statement is a line of code that only computers understand and execute. And when the program is in the reading mode the statement becomes an executable line of code and is converted into the object which only the computer understands.

1.4.2 Branch Coverage

In programming language the “if statement” is known as branch. The if statement has two branches: if the condition is true the code executes and if it is false the code does not get executed. It is also called decision coverage. In this each and every branch is checked whether it is being executed at least once.

1.4.3 Path Coverage

In this coverage every different part of the program is tested. In this testing it is checked whether each and every part of the program is executed at least once. This technique is more useful than the branch coverage because it is used for testing the complexity of the program.

1.5 Agile Methodology

Agile methodology is a different method of software development cycle where all the prerequisites and requirements of the client are met through the complete collaboration and efforts of the entire software as well as testing teams [10]. Teams formed in this methodology are called agile teams, which each and every member is equally held responsible for success or failure of the system. It is not only the responsibility of the software team for the testing team to ensure the success of the product, it is the responsibility of the entire team. The organization inside the team is done in a way that each in every step is followed by feedback and through trial and error method.

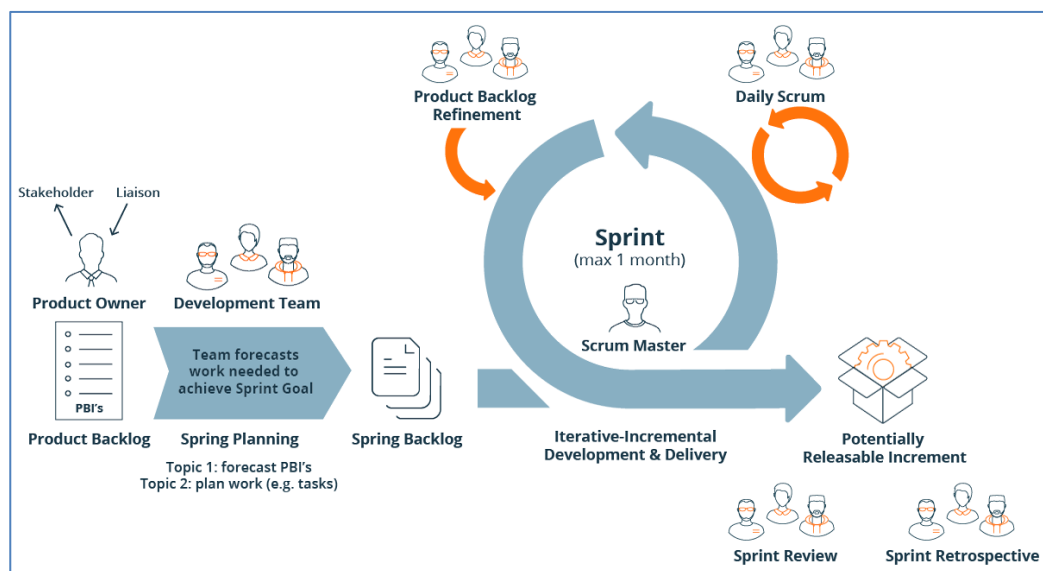


Figure 1.8: Agile Model

1.5.1 Agile Testing Principles

- a) **Continuous Testing:** The agile team tests the product continuously to ensure whether the product is meeting all its prerequisites and to ensure the success of the product.
- b) **Continuous feedback:** Ensuring that the product makes all its pre requirements each and every step is followed by a feedback on a regular basis.
- c) **Tests performed by the entire team:** The entire agile team is held responsible for the success or the failure of the product. Not only one member or one team like the developing team or the testing team is held responsible but the entire agile team.
- d) **Decrease time of feedback:** The client team is completely involved in the entire process and it provides feedback after each and every stage.
- e) **Simplified and concise code:** All the errors reported by the agile team are debugged or corrected in the same cycle so that the resultant code is concise and meets the requirement.
- f) **Reduced documentation:** There is a checklist that the agile team uses on a regular basis to check the status of the work completed.
- g) **Driven test:** In normal software development cycle the testing is done after the implementation of the code. But in agile methodology testing is done side by side the creation of the code.

CHAPTER 2

DATASOURCE

2.1 SQL using MySQL Database

Structured query language is a programming language which is solely designed for a specific domain and is created for handling data present in the relational database management system [11]. This programming language is designed for handling structured data. Structure data is the one in which there is a relationship between entities and variables. There are many different types of statements that SQL uses which can be classified into different sub languages. Data definition language, data control language, data manipulation language. SQL is used to insert, delete or update the data present in the database and gain access to control the data.

2.1.1 Relational Database Management System (RDMS)

Organizing the data into two rows and columns within a table is called relational database design. The tables order relation has different rows and different columns in which each row represents a record and each column represents an attribute. The programming language SQL is used to manage the data for accessing the data in these RDMS [12]. There are total four stages in which the RDMS is designed which are:-

- Define relations/attributes
- Define primary keys
- Define relationships
- Normalization

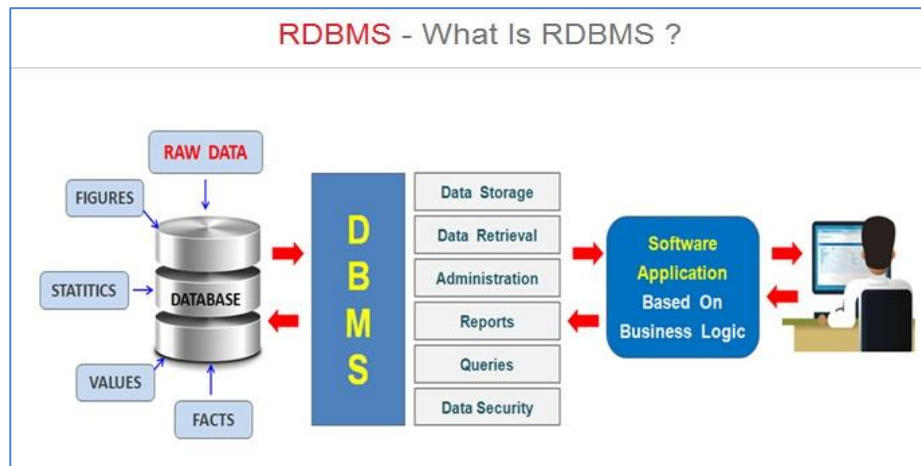


Figure 2.1: RDBMS

RDBMS is different from other database management systems with their way of organizing data and accessing data are very different. In this management system the data is organized in tables which are related to one another. There should be a database server in applications so that relational database design is made possible for data management.

Search relational database management system is MySQL. It is an open source system. It is also used for developing web based applications in the field of RDBMS.

2.1.2 Data Definition Language (DDL)

As mentioned above data definition language is a sublanguage which mainly deals with the creation and finishing of the database [13]. The structure of the database is defined and modified by this data definition language and describes the overall look of the database. Some of the data definition language commands are:-

CREATE – this command is used to create a table or a database.

DROP – this command is used to delete tables or delete databases.

ALTER – this command is used to make any changes to the structure of the database.

TRUNCATE– this command is used to remove all the values from the table.

2.1.3 Data Manipulation Language (DML)

Data manipulation language is a sublanguage of SQL which is used to manipulate the data which is already present inside the database [14]. It is used to update the data inside the database.

Some of the examples are:-

INSERT – this command is used to insert values inside the table.

UPDATE – this command is used to update the data inside the table.

DELETE – this command is used to delete the records from the table.

2.1.4 Some SQL Queries

```
1 select distinct h.hotel_id, h.hotel_name, h.rating from hotel_details h
2 join orders o on h.hotel_id = o.hotel_id
3 where o.order_date between '2019-07-01' and '2019-07-31'
4 order by h.hotel_id;
```

```
1 select distinct o.owner_id, o.owner_name, o.address, o.phone_no from owners o
2 join cars c on o.owner_id = c.owner_id
3 where c.car_name like 'Maruthi%'
4 order by o.owner_id;
```

```
1 select c.car_id, c.car_name, c.car_type from cars c
2 left join rentals r on c.car_id = r.car_id
3 where c.car_id not in (select r.car_id from rentals r)
4 order by c.car_id;
```

2.1.5 Database Design

The entire collection of different processes and activities which are responsible for the designing and development of the database management system is called database design [15]. The databases which are properly designed and are regularly maintained are easy to

read and require less storage than the database which is not properly designed. The designers who design the database keep in mind to correlate the data within the database effectively so that maximum amount of information can be conveyed through data utilizing minimum storage space. It is used to design database systems:-

- a) That fulfills the requirements of the client or the user.
- b) That has high performance.

2.1.6 Normalization

Normalization is a technique in designing databases which is used to remove all the redundant data from the database and eliminate the undesired anomalies introduced by data manipulation language [16]. The main idea of normalization is to divide large sets of tables into smaller different sets of tables and link them together using a common relationship. Main purpose of normalization is to reduce the repetitive data and in short the effectiveness of the database.

The normalization is still being developed further but for the most cases it is used in three forms.

a) First Normal Form (1NF) Rules

- There should be a single value inside a cell.
- Each row should be unique.

b) Second Normal Form (2NF) Rules

- Should be in first normal form
- There should be a column of primary key.

c) 3NF (Third Normal Form) Rules

- Should be in second normal form
- Should not have any functional dependencies which are transitive.

2.2 Documenting APIs Via JSON

JSON full form stands for JavaScript object notation. JSON was originally created and is based on JavaScript language it is used to contain and handle the structured data within JavaScript

[17]. It is very easy to understand it became popular among users and most of the web applications were based on JSON.

The four data types of JSON are:-

1. Number- Can be an integer or decimal and can be positive or negative.
2. String- Used to store characters and is enclosed within single or double quotation marks.
3. Boolean- It contains only two values: true or false.
4. Null- empty variable

There is no one such fixed standard for documentation JSON files. The figure below shows how the JSON file is documented.

```
1  {
2  "string": "Hi",
3  "number": 2.5,
4  "boolean": true,
5  "null": null,
6  "object": { "name": "Kyle", "age": 24 },
7  "array": ["Hello", 5, false, null, { "key": "value", "number": 6 }],
8  "arrayOfObjects": [
9      { "name": "Jerry", "age": 28 },
10     { "name": "Sally", "age": 26 }
11 ]
12 }
13
```

Figure 2.2: JSON File

2.3 Documenting APIs Via XML

XML is a short form for extensible markup language. Language is very similar to HTML but its main purpose is to describe the data present [18]. Tags inside the XML are not predefined and one can define its own tags. So because of this convenient nature of XML it is used for defining any kind of structure data. XML has only one data type which is string. Similar to JSON there is no predefined one way of defining XML files.

Documentation of API often uses either JSON or XML mostly. Design structure of both JSON and XML files are pretty much the same. The difference lies in the availability of attributes. In this case XML uses attributes and JSON does not.


```
1 <?xml version= "1.0"?>|
2 <Department>
3   <Employee>
4     <empid>1001</empid>
5     <name>Tom</name>
6     <salary>20000</salary>
7     <email>tom@gmail.com</email>
8     <phoneno>9874563210</phoneno>
9   </Employee>
10  <Employee>
11    <empid>1002</empid>
12    <name>Sam</name>
13    <salary>25000</salary>
14    <email>sam@gmail.com</email>
15    <phoneno>7876545676</phoneno>
16  </Employee>
17  <Employee>
18    <empid>1003</empid>
19    <name>Shiny</name>
20    <salary>20000</salary>
21    <email>shiny@gmail.com</email>
22    <phoneno>9876543210</phoneno>
23  </Employee>
24 </Department>
```

Figure 2.3: XML File

CHAPTER 3

VBSCRIPTING

VBScript short form for Visual Basic Scripting is a scripting language that is a part of visual basic for application [19]. VBScript is used for creating applications that can be run on Microsoft Windows. The file extension used for VBScript is .vbs. This language makes the use of the environment called Windows script host environment.

It is a very simple and easy to understand language and can be coded on a text editor like Notepad. And any VB script file created can be run by double clicking on the VBS file.

3.1 Features of VBScript

- i. This language is simple and easy to understand and has a very fast interpreter.
- ii. The syntax of this language is pretty easy to understand and simple and for the most part, the language is case insensitive.
- iii. It is an object-based scripting language, not object oriented programming language like C++ or java.
- iv. VB script can only be executed in some host environment such as internet explorer or Windows scripting host and internet information services.

Script interacts with the end user through functionalities like “Msgbox” and “Inputbox” that is used to provide a simple box for displaying messages for asking the user for input. For increased interaction between the user and the environment, VBScript can be used with HTML.

3.2 Uses

- i. First and the most important use of VB script is it is used for automation testing. UFT utilizes this language for descriptive programming.

- ii. For automation of the windows desktop, Windows system administrator make use of Windows scripting host.
- iii. This language is also used for development of web pages in a server scripting environment.
- iv. Internet explorer by Microsoft also uses VBScript for client side scripting.

3.3 Some of the Hands-on done on VBScript

1. Factorial

```
1  option explicit
2
3  Dim num, f, m
4
5  num = InputBox("Enter the number")
6  m = num
7  f = 1
8
9  Do while num>0
10     f=f*num
11     num=num-1
12 Loop
13
14 MsgBox "Factorial of " & m & " is " & f
15
16
```

2. Directory Creation

```
1  Dim ofso , startfolder
2  startfolder = "C:\Users\Acer\OneDrive\Desktop\cogni_training\Week 3 VBscripting"
3  set ofso = CreateObject("Scripting.FileSystemObject")
4
5  if ofso.FolderExists(startfolder) then
6     MsgBox startfolder & " already exists" , 0, "Result:"
7  else
8     ofso.CreateFolder(startfolder)
9     MsgBox startfolder & " created." , 1, "Status:"
10 end if
```

3. FSO objects

```
1 option explicit
2
3 Dim oFSO, strFolder
4
5 strFolder = "C:\Users\Acer\OneDrive\Desktop\cogni_training\Week 3 VBscripting\Folder2"
6 Set oFSO = CreateObject("Scripting.FileSystemObject")
7
8
9 if oFSO.FolderExists(strFolder) then
10     MsgBox " Already exists"
11 else
12     oFSO.CreateFolder(strFolder)
13 end if
14
15 Set oFSO = Nothing
16
17
```

4. Error Handling

```
1 Dim num1,num2,div
2
3 num1=10
4 num2=0
5
6 On Error Resume Next
7 div = num1/num2
8
9 Err.Raise 101
10 MsgBox "Error: " & CStr(Err.Number) & " " & Err.Description
11 MsgBox CStr(div)
12
13 Err.Clear 'Reset
14 MsgBox "Error: " & CStr(Err.Number) & " " & Err.Description & "EOM"
15
```

5. Regular Expression

```
1 dim n
2 n = InputBox("Enter Positive Number : ")
3
4 Set name = New RegExp
5 name.Pattern = "Inbox"
6 name.IgnoreCase = False
7 name.Global = True
8
9 if n = 0 then
10     msgbox name.Pattern
11 elseif n<0 then
12     msgbox "Enter Positive Number"
13 else
14     msgbox name.Pattern & "(" & n & ")"
15 end if
16
17 set name = Nothing
18
```

CHAPTER 4

UFT Automation

Micro Focus Unified Functional Testing was previously known as QuickTest Professional. This software is used to test applications and different environments for their functionality [20].

This software utilizes the keyword view as well as scripting view for better graphical user interface. Also uses VBScript language for descriptive programming and to create objects and control the application which is under the test. This software allows the developing team to test all the layers of an application: the user interface, the service layer and the database.

4.1 Advantages of UFT

- i. It can be used to record as well as playback.
- ii. It can record script side by side running and application so that the user can refer to the recorded object properties.
- iii. It can easily and effectively identify all the objects present in any application.
- iv. One can add a number of different add-ins like .net web Java etc.
- v. It has an inbuilt IDE.
- vi. It also supports XML files.
- vii. It provides a detailed result report for analysis.
- viii. It supports different kinds of testing approaches like data driven testing, keyword driven testing etc.

4.2 UFT/QTP IDE

4.2.1 Add-In Manager

The add-in manager is the first box that pops up on opening UFT. In this we can select a different number of add-ins that we require. Selecting all the items that we need, ok button should be clicked.

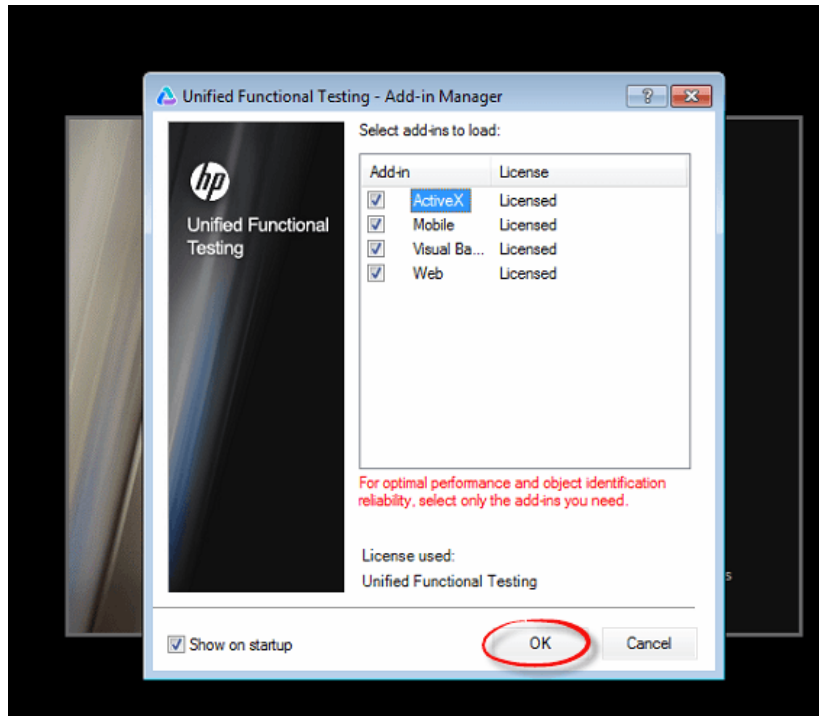


Figure 4.1: Add-in Manager

4.2.2 Start Page

The first page shows all the current features available in the UFT software. It also has a menu bar with a lot of different options. There are different kinds of tabs available namely toolbox tab output tab and data tab. This is also an option of opening recently saved files.



Figure 4.2: Start Page

4.2.3 Actions

An action is basically a logical unit of activity. Action is used to create a script which is Efficient and modular. When a new script is created it consists of only one action but there is also an option of adding more actions to a single script. On clicking the script editor window and selecting properties, we can access the properties of the selected action.

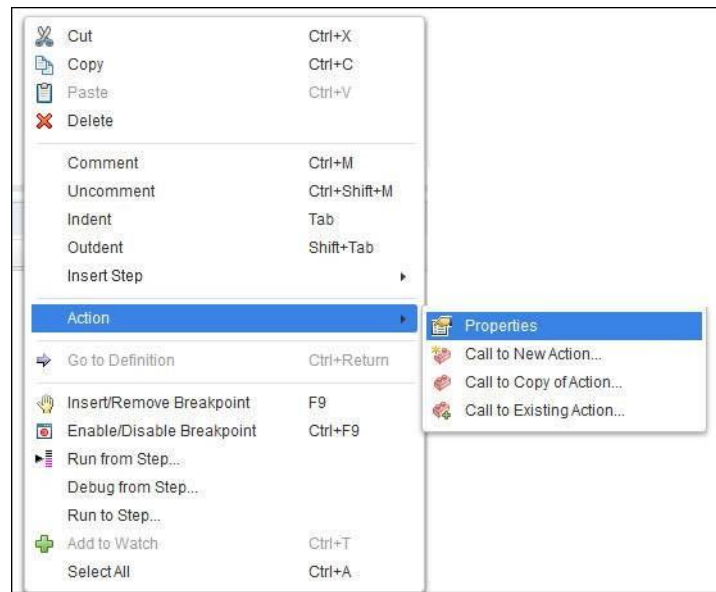


Figure 4.3: Types of Actions

Actions are of two types:-

- i. Reusable Actions: these kinds of actions can be reused in other scripts and can also be used in the same script multiple times.
- ii. Non-Reusable Actions: kind of actions cannot be used in other tests and can only be used once in the same test script.

Actions are also allowed to access data stored in the data sheets. The main purpose of using action in a script is that an action can be used in other test scripts also. There are two methods by which we can import actions into different scripts:-

- i. Call to COPY of an Action: When this method is invoked a copy of action is made available. In addition, all related properties like objects, parameters, and data tables

related to Action also get copied into the script. In this method the user can make any number of changes in the copied action.

- ii. Call to an EXISTING Action: When will method is invoked the actions which are being copied are in read only format. Cannot make any changes to that action. And in this method only reusable actions can be called.

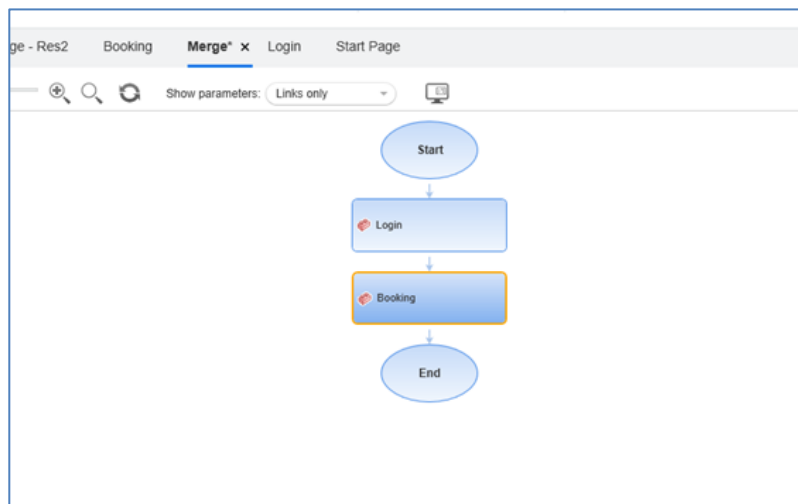


Figure 4.4: Multiple Actions

4.2.4 Transactions

Basically a transaction is used to check how much time and activity is taking for its completion. Transactions are used on activities that have start and an end.

Transactions are useful for:-

- i. It is basically used to measure the performance length of the script
- ii. The output which is generated to the transactions can be assessed for analysis and optimizing the script at certain areas.

Defining a Transaction:-

After choosing the section of the script that we want to measure, “StartTransaction” statement should be included before the section and “EndTransaction” statement should be included after the end of the Section. Transaction can be also defined within another transaction there is and there is no limit to which how many transactions can be used within a script. The end test report also shows the measurement of the transaction.

Below is an example of the syntax for defining a transaction.

```
Action.StartTransaction "Booking"  
...  
...  
Action.EndTransaction "Booking"
```

4.2.5 Checkpoints

This method is used to check whether the output value is the same as the expected value or it is different. If the expected value is the same as the output value then the checkpoint passes otherwise it fails.

Types of checkpoints in UFT:-

- i. Standard checkpoint- It checks whether the properties of the object are the same during both recording sessions as well as run sessions.
- ii. Page checkpoint- It is a type of standard checkpoint which is specifically focused on webpage. The number of links present and the number of images present on a webpage can be checked through this checkpoint.
- iii. Bitmap checkpoint- Every image has the bitmap associated with it. This checkpoint is used to check the map of every image or a webpage also.
- iv. Image checkpoint- This point the properties of the image of the webpage is checked rather than the pixels of the image.
- v. Text checkpoint- This checkpoint is used for checking any text written in a web page or Windows based application.
- vi. Database checkpoints- During the recording this checkpoint can create a table of expected values. And during runtime the actual output values and the expected values in the data table are compared.

- vii. Table checkpoint- In this checkpoint user can check the values present in the table in any environment. The properties of the table can also be checked.

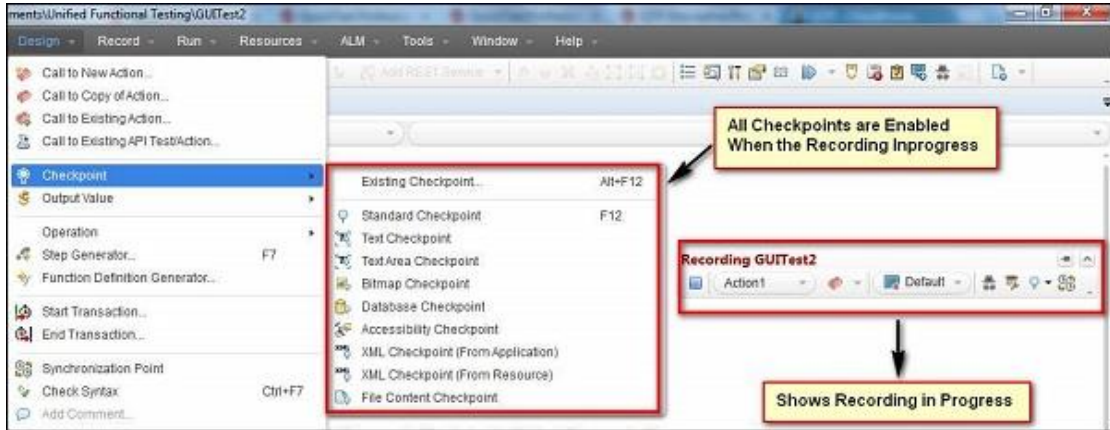


Figure 4.5: Selecting Checkpoints

4.2.6 Parameterization

During the run time, user can choose different test values for a single input which is termed as parameterization. This is done through giving external parameters to the input value.

Different types of parameterization are:-

- i. Datable parameters
- ii. Action parameters
- iii. Environment variable parameters
- iv. Random number parameters

We can parameterize using the keyword view also which is very easy to understand and convenient.

- i. Click on the Keyword View
- ii. Click the Parameterization button.
- iii. In Value Configuration Dialog Box, Parameter Radio Button should be selected and give an appropriate name to the parameter.

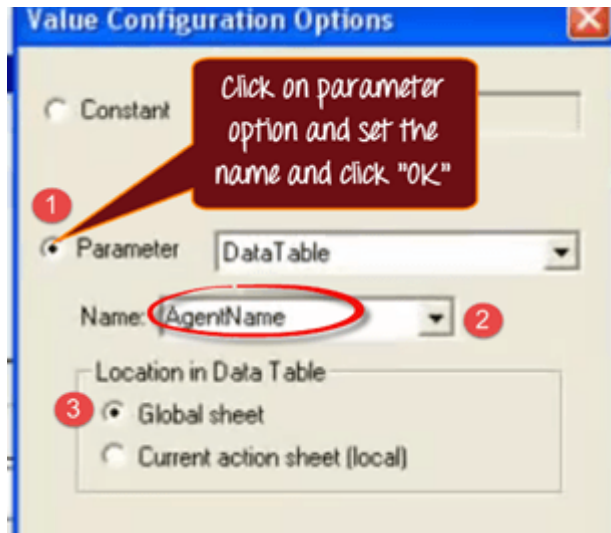


Figure 4.6: Parameterization

A column with the name of the parameter is created in the Global Sheet and we can further add more values in the datasheet for this parameter.

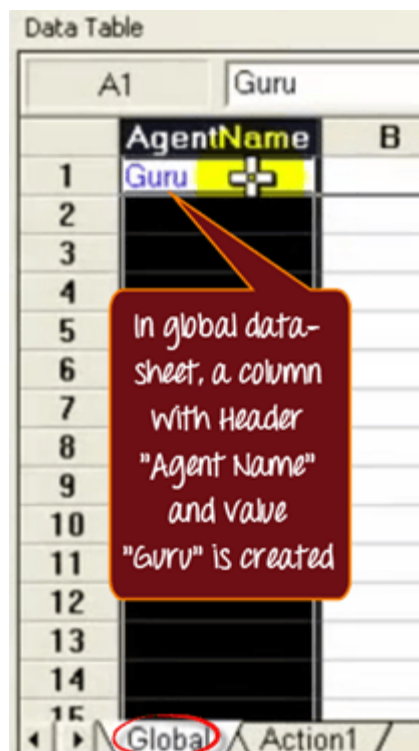


Figure 4.7: Parameters in GlobalSheet

Some of the Advantages of parameterization are:-

- i. During the runtime session user can add more values to one single input.
- ii. It therefore reduces time and efficiency is increased.

4.2.7 Recording

During recording, the user can record all the action that the user performs on the webpage or any window based application. After recording during runtime the UFT automatically generates all the actions that the user performed step by step. And record and run settings on the menu bar can help us to modify the settings of recording whether the recording is to be done on a web based application or a window based application. The steps of recording are shown below:-

Click on the “Record” dropdown and select “Record and Run Settings”

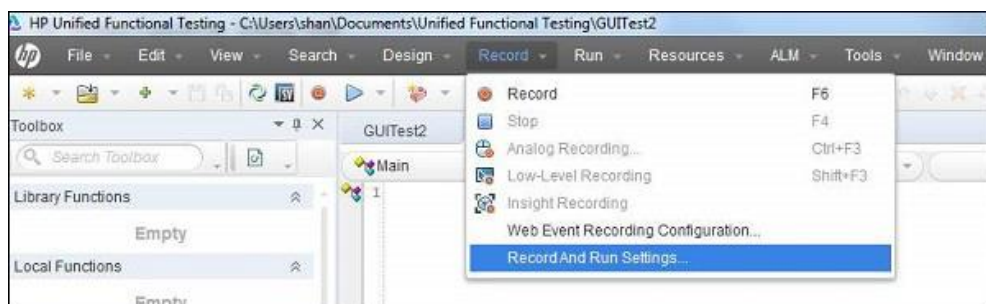


Figure 4.8: Record Settings

Select whether the recording is to be done on web based application or window based application.

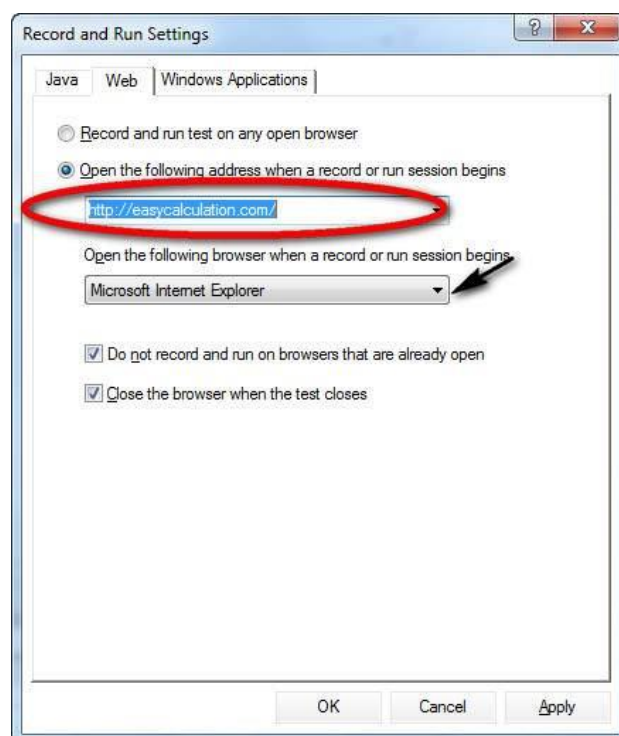


Figure 4.9: Settings dialogue box

Click on the record and the application will automatically be opened.

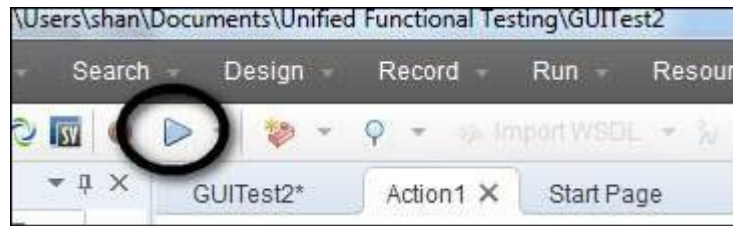


Figure 4.10: Run Button

The importance of record and playback:-

- i. It is the first method to check whether the UFT supports a specific application or not.
- ii. Basic functionality of an application can be tested and developed using this.
- iii. Recording can be done for both mouse as well as keyboard inputs.

There are four different Modes of recording:-

- i. Normal recording- it records the properties of the objects and operations being performed on the application. This is the default mode.
- ii. Analog recording- in this recording both mouse as well as keyboard imports are recorded.
- iii. Low level recording- type of recording only the coordinates of the different objects on the application is recorded. It is not necessary whether the UFT recognizes the object or not.
- iv. Insight recording- in this recording operations are recorded based on their appearance but not on the basis of their native properties.

4.2.8 Object Repository

Object repository contains all the objects and the properties are recognized by the UFT. During the recording session all the objects and their associated properties are captured by default in the object repository. If the UFT is not able to recognize the objects and its properties it will not be able to play back the script.

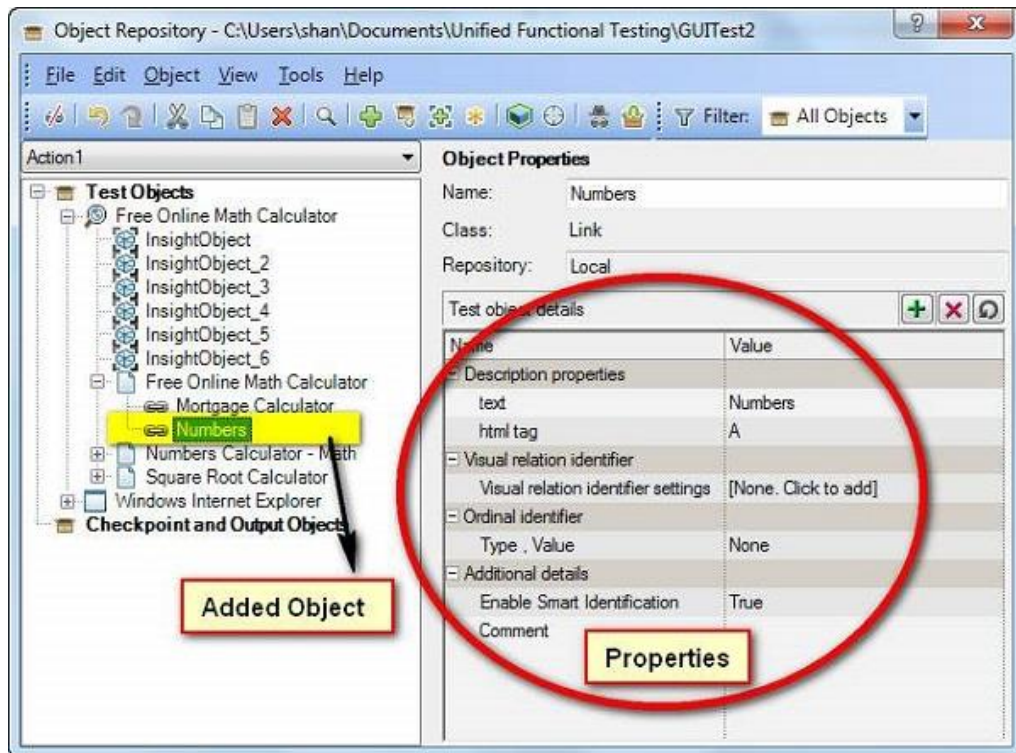


Figure 4.11: Object Repository

The object identification is done in UFT through:-

- i. UFT recognizes the object using a "human" like technology.
- ii. UFT recognizes the objects and properties during recording of the application on which the operations are being performed.
- iii. During the runtime session the stored object properties are compared with the actual properties of the object in the application visible on the screen.
- iv. The object which is stored in the repository along with its properties is termed as a test object.
- v. During run time the application under test has its own object which is called runtime object.
- vi. Object repository contains all the information regarding all the test objects.

There are two types of object repository

- i. Local Object Repository
- ii. Shared Object Repository

Local object repository

- i. It is the default repository in UFT.

- ii. Local object repositories are different for every action and every action has its own individual object repository.
- iii. Operations can be performed on the local object repository like copy and pasting modifying and deleting objects.

Shared Object repository

- i. Also called global object repository, it is used for storing objects which change its value and description frequently.
- ii. In most projects that are focused on automation, shared object repositories are used more than local.
- iii. Shared object repositories are saved as .tsr files.
- iv. To create shared object repository, it is created through object repository manager as shown below

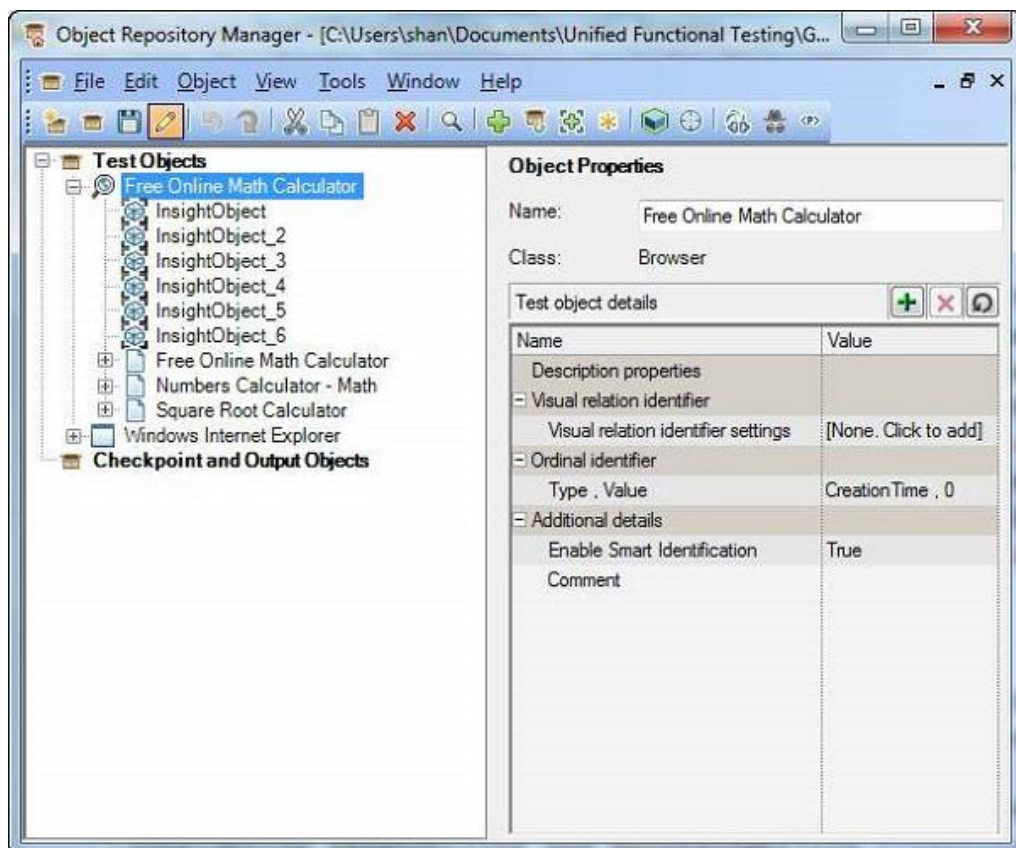


Figure 4.12: Object Repository Manager

4.2.9 Environment Variables

These are the dynamic objects which are stored on a computer and are assigned a value. These variables can be accessed by any software programs in the Windows operating system. These variables are dynamic in nature and they are not constant and can be modified. Are many environment variables that are internally stored inside the system which can be accessed by any program in the operating system to find out the information regarding the computing environment.

Mainly there are two different types of environment variable present:-

- i. Built-in variables
- ii. User defined variables

Built-in variables

These variables are predefined inside the system and have a constant value. Variables are used to retrieve the information regarding the operating system on which the test is being executed. Many different built-in variables like operating system, test directory, operating system version etc.

Environment tab in Settings is used to view all the environments variables and their values as shown below:-

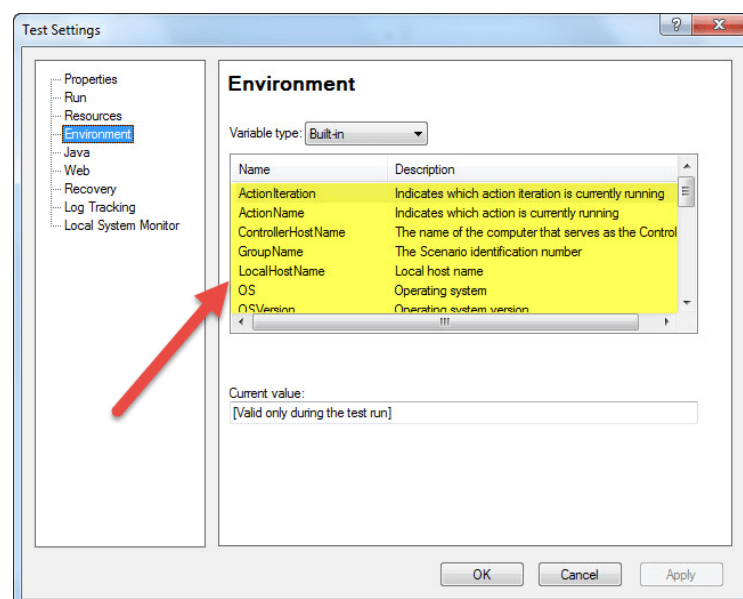


Figure 4.13: Environment Variables

Environment variables can be accessed anytime and anywhere during the runtime session of the UFT as shown below:-

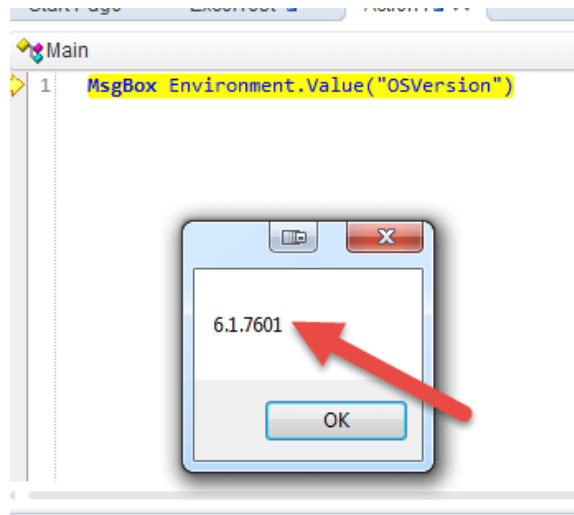


Figure 4.14: Accessing Environment Variables

User defined variables

Some of the variables can be defined by the user that is called user defined variables. These variables are made available to each and every test or can be restricted to only one test.

User defined variables can be further divided into two types:-

- i. Internal variables- these variables are created by the user within a test and it is made available only in one single test.
- ii. External variables- these variables are also defined by the user within a test but it can be used by other tests also.

There are two ways in which external variables can be defined. It can be created through the environment tab in test settings or it can be loaded from an external XML file.

User defined variable can be defined as shown below:-

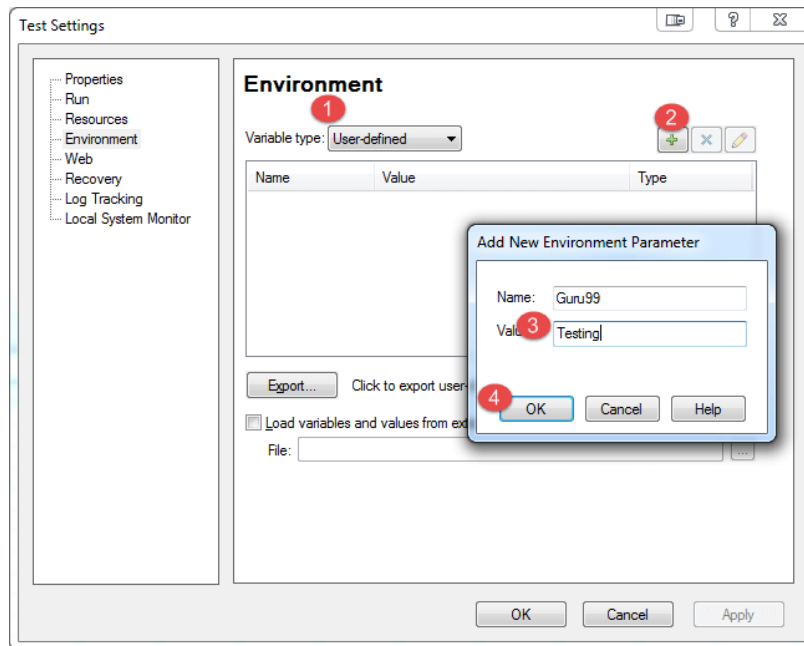


Figure 4.15: User Defined Variables

4.2.10 Functions

Functions are the segments of code that is defined within the parentheses and can be used multiple times within a program. The functions which the users create by themselves are called user defined functions. By using these functions code redundancy can be reduced and efficiency can be increased. Define functions in unified functional testing usually contain programs written in VB script and can comprise of different modules and subroutines.

To create a function library inside a test script, the function library option is to be selected under File dropdown menu. This will open a new tab which will be a function library as shown below:-

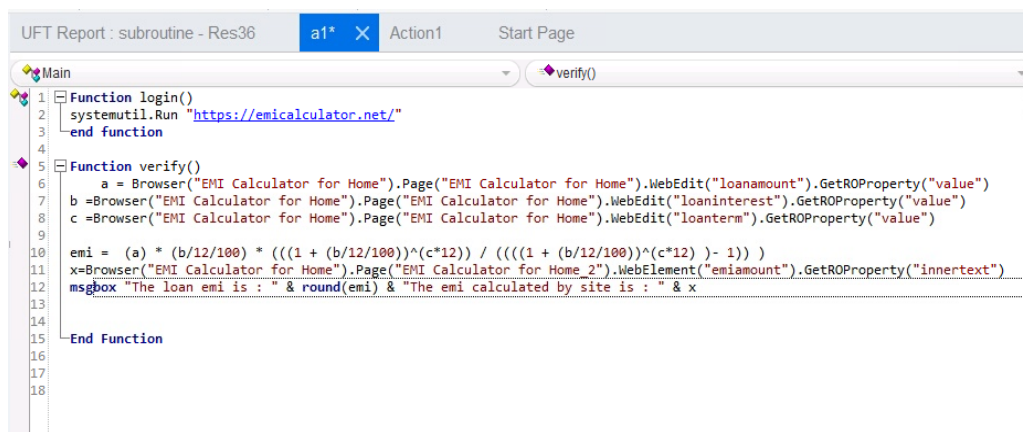


Figure 4.16: Function Library

Multiple functions can be defined inside a single file and these function libraries are saved with the extension .qfl.

4.2.11 Datatables

These Datatables contain datasheets just like Microsoft Excel for storing the data. These datasheets can be used for or giving multiple inputs to a specific test case which can be further used to run an action many times.

The two different types of datatables are:-

- i. Local DataTable– Every action is associated with its own unique datatable which is known as local data table. This local data table can and also be accessed by other actions.
- ii. Global DataTable – Every test has one global data table which is also accessed by other actions.

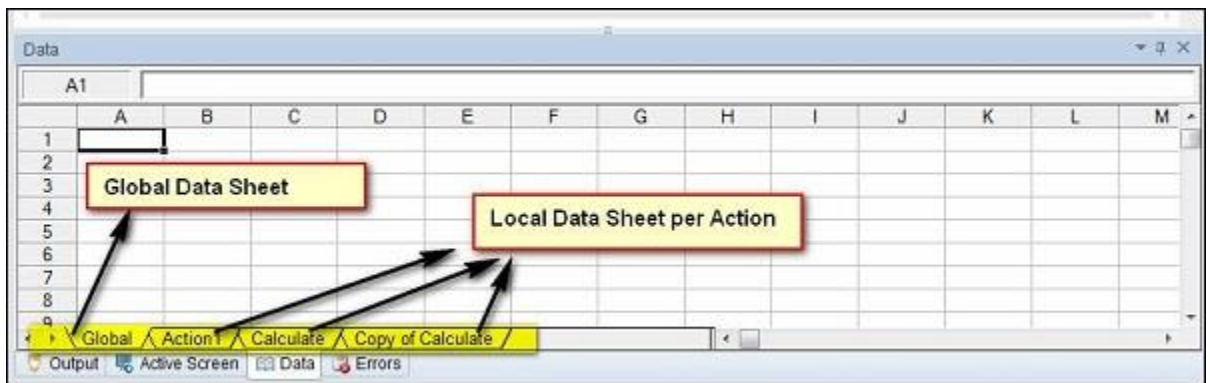
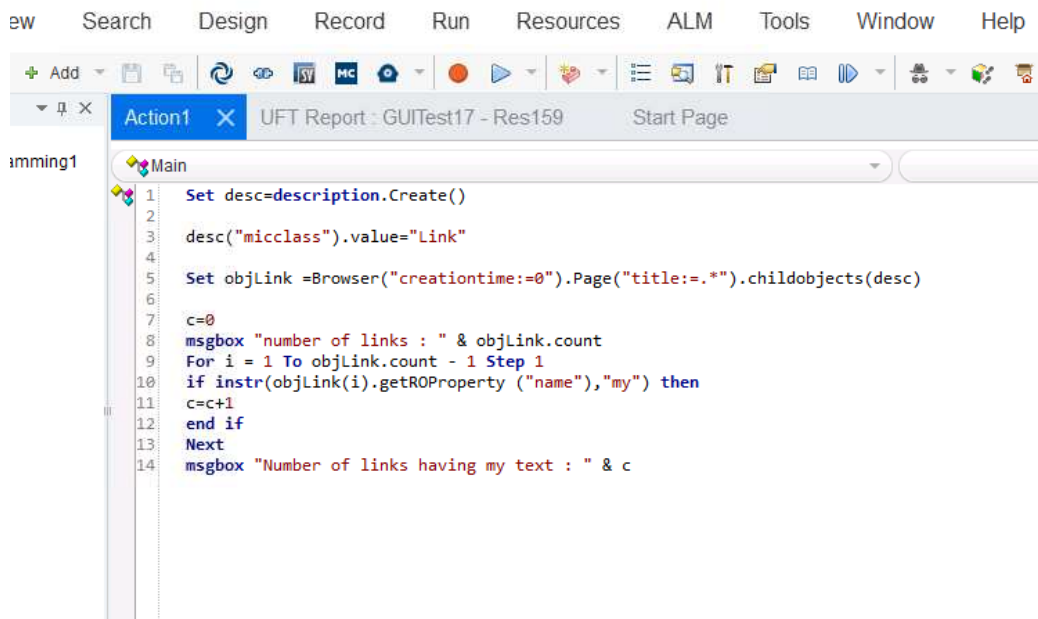


Figure 4.17: Datatables

4.2.12 Descriptive Programming

Each and every object present in the object repository has some description related to it. The descriptions that are associated with the object are created using a special programming called descriptive programming. This programming is used when the user wants to perform an operation on the object that is not available in the object repository. It is also required when the test script is very dynamic and the object repository is too crowded IT results in slow performance of the UFT.

Below is an example of how to use descriptive programming for creating objects and accessing their unique properties.



The screenshot shows a software interface with a menu bar (ew, Search, Design, Record, Run, Resources, ALM, Tools, Window, Help) and a toolbar. The main window is titled 'Action1' and contains a code editor with the following code:

```
1 Set desc=description.Create()
2
3 desc("micclass").value="Link"
4
5 Set objLink =Browser("creationtime:=0").Page("title:=.*").childobjects(desc)
6
7 c=0
8 msgbox "number of links : " & objLink.count
9 For i = 1 To objLink.count - 1 Step 1
10 if instr(objLink(i).getROProperty ("name"),"my") then
11 c=c+1
12 end if
13 Next
14 msgbox "Number of links having my text : " & c
```

Figure 4.18: Descriptive Programming

Each property of individual objects has their corresponding values associated to it. The child object method of descriptive programming enables users to create objects.

CHAPTER 5

FUTURE WORK

- i. In the coming weeks we will start with HC Facets course and will work with Healthcare Facets Product technology.
- ii. Along with the completion of the course, we will be working on multiple hands-on related to the topic for better understanding.
- iii. Quizzes and assessments will be taken for evaluating our knowledge in the related subject.
- iv. We will also work on our Hackathons, Mini as well as Main projects exhibiting all the skills that we learned throughout the training.

CHAPTER 6

CONCLUSION

- i. The comprehensive learning path provided by Cognizant gave us an opportunity to understand the corporate environment, and groom ourselves.
- ii. With the availability of latest tools and resources, practical knowledge was given more importance than theoretical knowledge through constant practice.
- iii. Regular quizzes, hands-on and assessments allowed us to continuously evaluate if we are able to apply those self-learnt skills to solve a practical business problem.
- iv. The complete program is designed in such a unique way allowing us to showcase our abilities, gain and measure programming skills and showcase our understanding.

REFERENCES

- [1] “Software Development LifeCycle” [Online]. Available: <https://medium.com/@jilvanpinheiro/software-development-life-cycle-sdlc-phases-40d46afbe384>
- [2] “Sequential Models” [Online]. Available: <https://richrtesting.com/a-simple-comparison-of-sequential-and-iterative-software-development-methods/>
- [3] “Waterfall Model” [Online]. Available: <https://www.javatpoint.com/jira-waterfall-model>
- [4] “V Model” [Online]. Available: <https://www.javatpoint.com/software-engineering-v-model>
- [5] “Iterative & Incremental Development” [Online]. Available: <https://istqbfoundation.wordpress.com/2017/09/18/iterative-incremental-development-models/>
- [6] “Software testing” [Online]. Available: <https://www.softwaretestingmaterial.com/software-testing/>
- [7] “Test levels” [Online]. Available: <https://www.edureka.co/blog/software-testing-levels/>
- [8] “Black box testing” [Online]. Available: <https://www.imperva.com/learn/application-security/black-box-testing/>
- [9] “White box testing” [Online]. Available: <https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/>
- [10] “Agile Testing Principles” [Online]. Available: <https://reqtest.com/testing-blog/agile-testing-principles-methods-advantages/>
- [11] “SQL” [Online]. Available: <https://en.wikipedia.org/wiki/SQL>
- [12] “RDMS” [Online]. Available: <https://www.tutorialspoint.com/Relational-Database-Management-System-RDMS>
- [13] “DDL” [Online]. Available: <https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/>
- [14] “DML” [Online]. Available: <https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/>
- [15] “Database design” [Online]. Available: <https://www.guru99.com/database-design.html#5>
- [16] “Normalization” [Online]. Available: <https://www.guru99.com/database-normalization.html>
- [17] “JSON” [Online]. Available: http://jawa9000.com/tutorials/Writing/Documenting_API-JSON.html#DocumentingJSON
- [18] “XML” [Online]. Available: http://jawa9000.com/tutorials/Writing/Documenting_API-XML.html#:~:text=Like%20JSON%2C%20XML%20is%20another,XML%20files%20for%20API%20documentation.
- [19] “VB scripting” [Online]. Available: https://www.tutorialspoint.com/vbscript/vbscript_overview.htm
- [20] “UFT” [Online]. Available: https://en.wikipedia.org/wiki/Micro_Focus_Unified_Functional_Testing/