**API Automation Framework For Exhaustive**

**Functional Testing**

**BITS ZG629T: Dissertation**

By

GAURAV KHURANA

2011HZ13006

**Dissertation work carried out at**

**NetCracker Technology Solutions, Hyderabad**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

November, 2013

ii

**API Automation Framework For Exhaustive**

**Functional Testing**

**BITS ZG629T: Dissertation**

By

GAURAV KHURANA

2011HZ13006

**Dissertation work carried out at**

**NetCracker Technology Solutions, Hyderabad**

Submitted in partial fulfillment of M.S. Software Systems

Under the Supervision of

Mr. Rakesh Biswal,

NetCracker Technology Solutions, Hyderabad

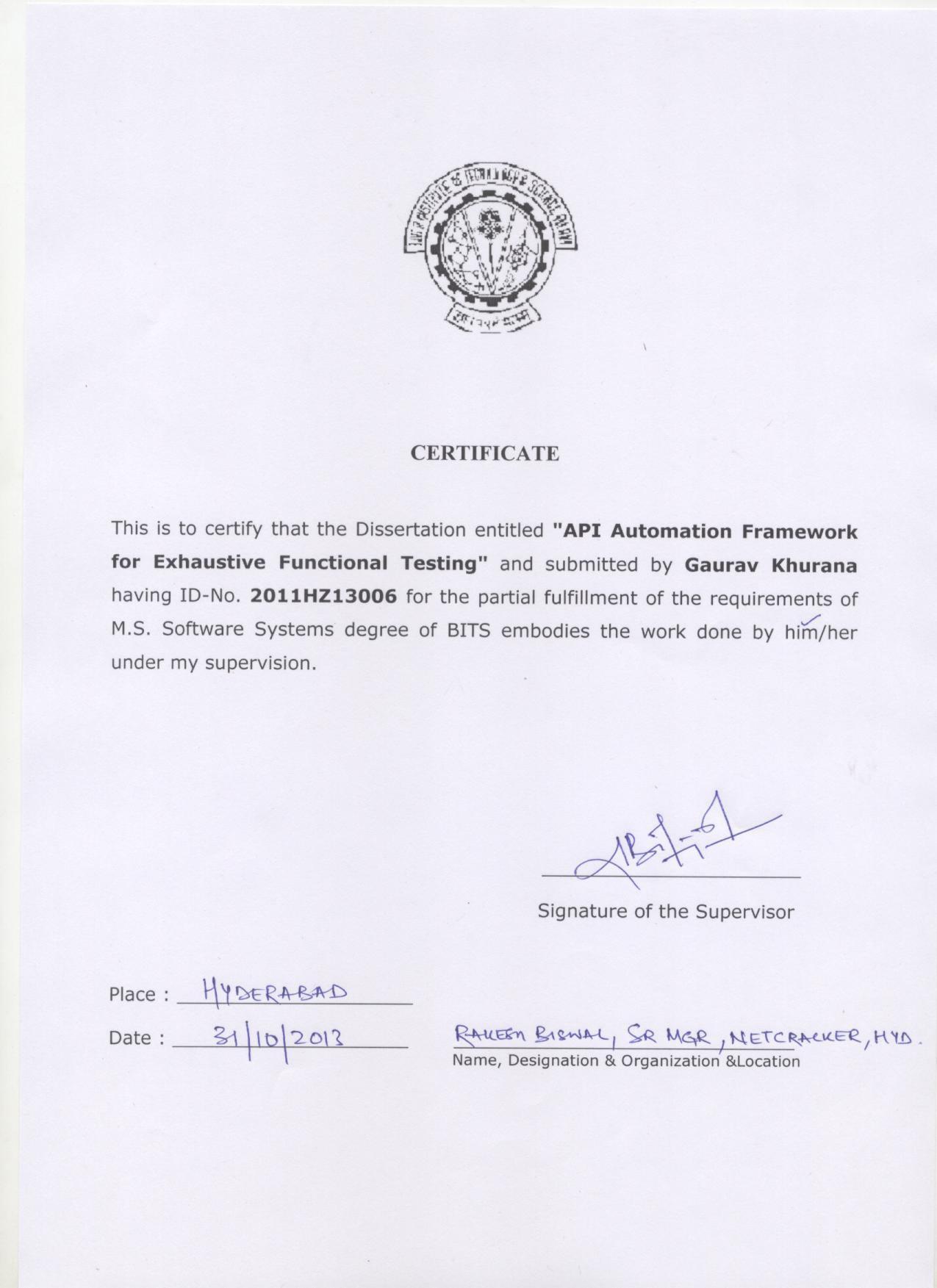


**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

November, 2013

iii



iv

**Birla Institute of Technology & Science, Pilani**

**Work-Integrated Learning Programmes Division**

**First Semester 2013-2014**

**BITS ZG629T: Dissertation**

**ABSTRACT**

**BITS ID No.** **: 2011HZ13006\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**NAME OF THE STUDENT** **: GAURAV KHURANA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EMAIL ADDRESS** **: gaurav.khuraana@gmail.com\_\_\_\_\_\_\_\_\_\_\_**

**STUDENT’S EMPLOYING** **: NetCracker Technology Solutions, Hyderabad\_**

**ORGANIZATION & LOCATION**

**SUPERVISOR’S NAME** **: Rakesh Biswal\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUPERVISOR’S EMPLOYING** **: NetCracker Technology Solutions, Hyderabad \_**

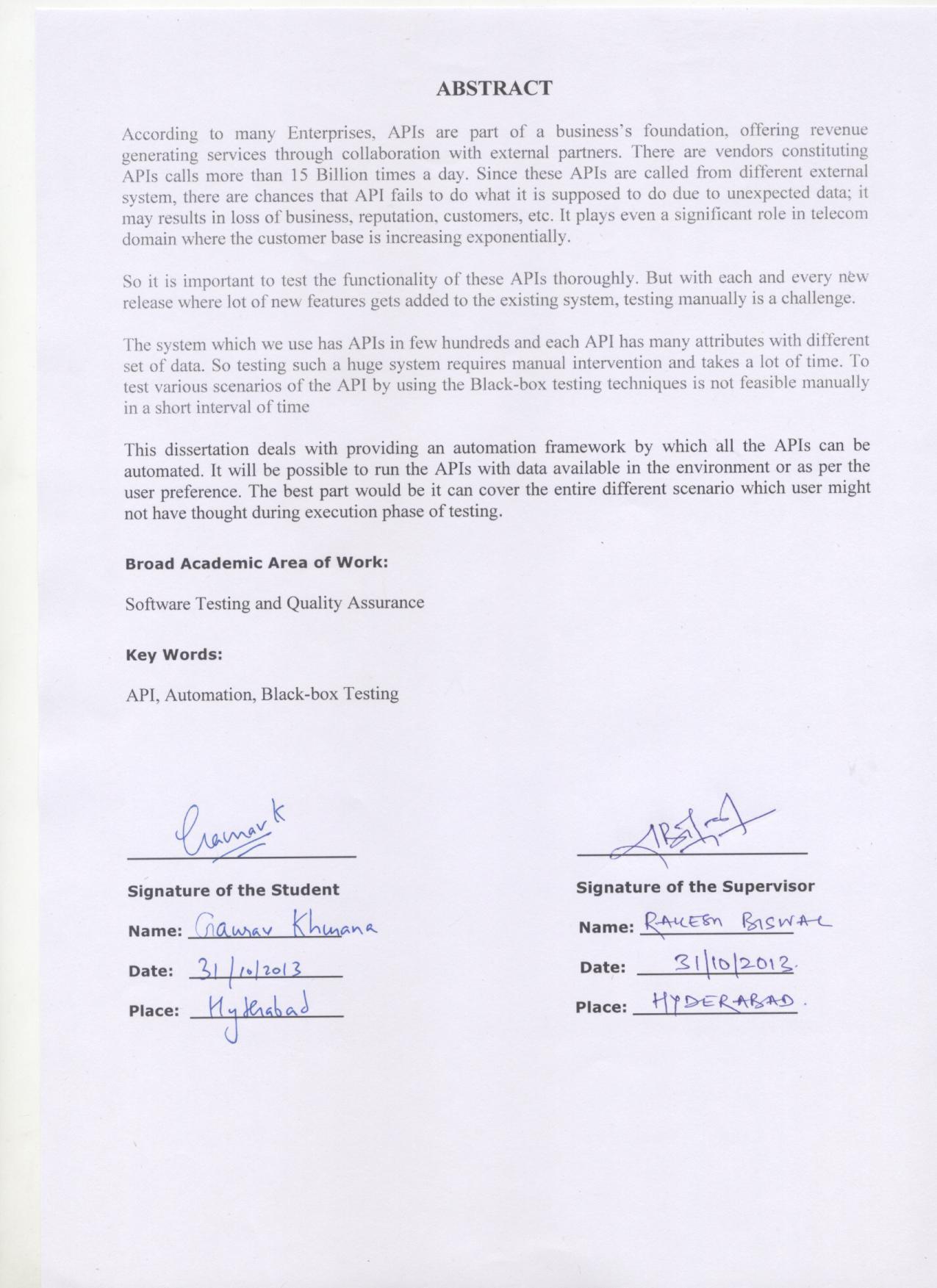
**ORGANIZATION & LOCATION**

**SUPERVISOR’S EMAIL ADDRESS: rakesh.biswal@netcracker.com\_\_\_**

**DISSERTATION TITLE** **:** **API** **Automation** **Framework** **for** **Exhaustive**

**Functional Testing**

v



vi

**ACKNOWLEDGEMENT**

This project would not have been possible without the support of many people. I would like to acknowledge and extend my heartfelt gratitude to the following persons who have made the completion of this project possible.

My Supervisor, Rakesh Biswal, for his full support and guidance throughout this project. My Manager, Mr. Devaraj Dey for encouraging and supporting me to pursue the project

Prof. Prasanna Chennupati from BITS Hyderabad, for giving the feedback at various stages of the project which acted as a motivation for working on the project

My testing team members who have helped me during various phases of the project.

Last but not the least, I would like to express my love and gratitude to my beloved family, for their understanding & motivation, through the duration of this project.

vii

**LIST OF ABBREVIATIONS USED**

API: Application Programming Interface

TC: Test Case

POT: Proof Of Testing

ENV: Environment

CACS: Customer Acquisitions and Customer Services

CSA: Communication Service Areas (CSAs)

HTML: Hyper Text Markup language.

IDL: Interactive Data Language

XML: Extensible Markup Language

CDR: Call Detail Record

SDLC: Software Development Life Cycle

EFT: Electronic Fund Transfer

BO: Business Object

viii

**LIST OF FIGURES**

[Figure 1 : Technical Architecture of Atlys](#page12) [2](#page12)

[Figure 2 : Current Mode of Operation](#page13) [3](#page13)

[Figure 3 : API Trends](#page18) [8](#page18)

[Figure 4 : API acting as Interface](#page19) [9](#page19)

[Figure 5 : XML APIs](file:///C:/BITS/Project/Final/2011HZ13006_final.docx%23_Toc371321313) [10](file:///C:/BITS/Project/Final/2011HZ13006_final.docx%23_Toc371321313)

[Figure 6 : Execution from Backend](#page21) [11](#page21)

[Figure 7 : Classification of Data](#page24) [14](#page24)

[Figure 8 : Overall flow of Tool](#page26) [16](#page26)

[Figure 10 : Input Path](#page31) [21](#page31)

[Figure 11 : Base Form](#page32) [22](#page32)

[Figure 12 : Detailed Level Attribute Values](#page32) [22](#page32)

[Figure 13 : Final form](#page33) [23](#page33)

[Figure 14 : Model for Testing](#page34) [24](#page34)

**LIST OF TABLES**

[Table 1 : Naming convention](#page31) [21](#page31)

[Table 2 : Statistics before tool implementation](#page35) [25](#page35)

[Table 3 : Statistics after tool implementation](#page35) [25](#page35)

ix

**Table of Contents**

Ch No. Title Page No

i) Certificate iii

ii) Abstract iv

iii) Acknowledgements vi

iv) List of Abbreviations vii

v) List of Figures viii

vi) List of Tables viii

[Chapter 1 Introduction](#page11) [1](#page11)

[1.1 Organizational Background](#page11) [1](#page11)

[1.2 About the Product](#page11) [1](#page11)

[1.3 Architecture](#page12) [2](#page12)

[Chapter 2 Project Overview](#page12) [2](#page12)

[2.1 Current Mode of Operation](#page12) [2](#page12)

[2.2 Problem with the existing solution](#page14) [4](#page14)

[2.3 Suggested solution](#page14) [4](#page14)

[2.4 Objectives](#page14) [4](#page14)

[2.5 Expected features](#page14) [4](#page14)

[2.5.1 User friendly design](#page14) [4](#page14)

[2.5.2 Cost Reduction](#page15) [5](#page15)

[2.5.3 Training](#page15) [5](#page15)

[2.6 Scope of Work](#page15) [5](#page15)

[Chapter 3 Motivation for Tool](#page16) [6](#page16)

[3.1 Automated Testing](#page16) [6](#page16)

[3.2 When we should do the Automation](#page16) [6](#page16)

[3.3 Automation is suitable for following types of Testing](#page16) [6](#page16)

[3.4 Benefits of Automation](#page17) [7](#page17)

[3.5 Areas where Automation can be attempted first](#page17) [7](#page17)

[3.6 Disadvantages of Automation](#page17) [7](#page17)

[3.7 Trends in industry](#page18) [8](#page18)

[Chapter 4 XML API](#page19) [9](#page19)

[4.1 What is an API ?](#page19) [9](#page19)

[4.2 What is XML?](#page19) [9](#page19)

[4.3 Working of XML API](#page20) [10](#page20)

[4.3.1 Extension Layer](#page20) [10](#page20)

[4.3.2 Integration layer](#page20) [10](#page20)

[4.3.3 Business Layer](#page21) [11](#page21)

[4.4 Why XML APIs](#page21) [11](#page21)

[4.5 Testing APIs](#page21) [11](#page21)

[Chapter 5 Design for Tool](#page22) [12](#page22)

x

[5.1 XML Parser](#page22) [12](#page22)

[5.2 Dynamic Control Generator](#page23) [13](#page23)

[5.3 Script Template](#page23) [13](#page23)

[5.4 Script Generator](#page24) [14](#page24)

[5.5 Classification of Data](#page24) [14](#page24)

[5.5.1 Integer](#page24) [14](#page24)

[5.5.2 String](#page24) [14](#page24)

[5.5.3 Float](#page25) [15](#page25)

[5.5.4 Boolean](#page25) [15](#page25)

[5.5.5 Date](#page25) [15](#page25)

[5.5.6 Query](#page25) [15](#page25)

[5.5.7 Custom Values](#page26) [16](#page26)

[Chapter 6 Implementation](#page27) [17](#page27)

[6.1 Base Class](#page27) [17](#page27)

[6.2 Dynamic Generation of controls](#page27) [17](#page27)

[6.3 Generic Functions](#page28) [18](#page28)

[Chapter 7: Project Details](#page29) [19](#page29)

[7.1 Introduction](#page29) [19](#page29)

[7.2 Hardware Requirements](#page29) [19](#page29)

[7.3 Software Requirements](#page29) [19](#page29)

[7.4 How to run](#page29) [19](#page29)

[7.5 Compatibility](#page30) [20](#page30)

[7.6 Features of Framework](#page30) [20](#page30)

[Chapter 8 Testing and Results](#page34) [24](#page34)

[8.1 Testing](#page34) [24](#page34)

[8.2 Results](#page34) [24](#page34)

[8.3 Before implementation](#page35) [25](#page35)

[8.4 After implementation](#page35) [25](#page35)

[Chapter 9 Conclusion](#page36) [26](#page36)

[Chapter 10 Recommendations and Future works](#page37) [27](#page37)

[Appendix I](#page38) [28](#page38)

[Category partitioning](#page38) [28](#page38)

[Bibliography](#page40) [30](#page40)

1

**Chapter 1 Introduction**

**1.1 Organizational Background**

NetCracker Technology is the proven partner for communications service providers and cable operators offering comprehensive, end-to-end solutions and delivery capabilities to optimize their enterprise. With its global reach, leading-edge technology, and unbroken track record of successful implementations, NetCracker helps eliminate operational and business silos and delivers real-time experience in an on-demand world.

In 2008, following 15 years of independent and profitable growth, NetCracker Technology was acquired by NEC Corporation. The acquisition was followed by NetCracker’s large-scale business expansion whereby NEC consolidated its services business under NetCracker. The assets encompassed innovative applications and service platforms, including customer and service management, network management. In May 2012, NetCracker completed the acquisition of the Convergys Information Management (IM) business. The Convergys IM business, with its long history of innovative BSS solutions, successful implementations, and managed services, strengthened the expansion into the BSS space.

NetCracker’s BSS/OSS solutions and services have been implemented and used successfully by hundreds of customers worldwide, including Fortune 1000 companies and the U.S. government

**1.2 About the Product**

**Remove the details about “company product” due to policies**

**You may add about the product here**

2

**1.3 Architecture**

Figure 1 : Technical Architecture of our Product

**Remove the details about “company product” due to policies**

**You may add about the product here**

**Chapter 2 Project Overview**

1. **Current Mode of Operation**
   * User has to provide the values manually for the APIs for all attributes.
   * The current API execution is being done via browser which is a lengthy process.
   * User has to paste the API in the given textbox and then press the submit button to submit the API to the particular server.

This is shown in the figure 2 how the browser is being used to fire the APIs.

3

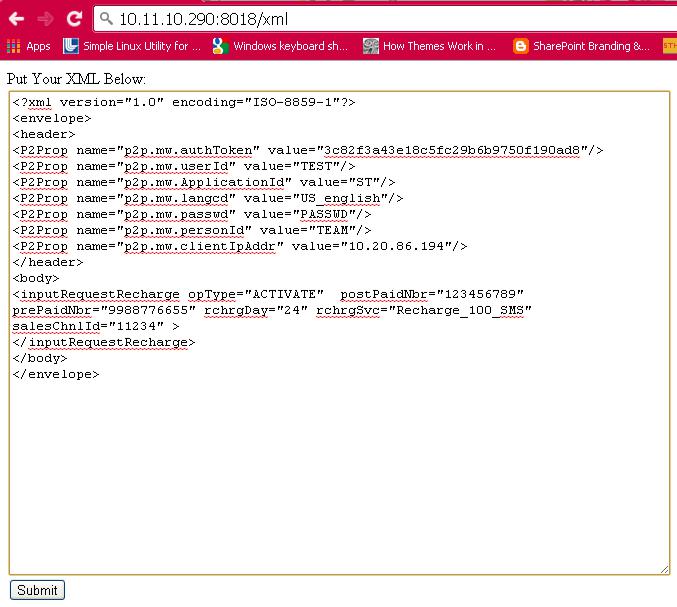


Figure 2 : Current Mode of Operation

4

1. **Problem with the existing solution**
   * User has to query the database and get the values manually which is time taking
   * Each and every time user has to copy the API, change the contents in case of errors and re-paste into the designated area and send it for processing.
   * This execution involves three software to be opened simultaneously. A notepad for the backup of file, a browser to run the APIs, a word document to take the POT for the scenarios.
   * It not only consumes space in system but also makes the system slow. It also distracts the user from the main task of testing the API.
   * With all the above mentioned problems, user is not able to test all the possible scenarios.
   * Some attributes get the same value each & every time for many releases.
   * Only few set of values get tested.
   * All the error codes are not tested every time.
   * The most importantly it’s a time taking process.
2. **Suggested solution**

Considering the different problems faced in the existing way of manual execution of API, the project includes the following different aspect to facilitate the user with necessary automation and overcome the problem providing additional advantages.

1. **Objectives**
   * Reduce the number of Person-hours required to complete testing thus saving a lot of effort for the project
   * Cover Scenarios which are not possible to be done manually covering Black Box testing techniques.
2. **Expected features**

The solution should be made to bring automation into operation part of execution of API including the following basic feature

**2.5.1 User friendly design**

This project should provide the user with the ease of avoiding the hassle of querying database every time using three software’s simultaneously. This will save a lot of effort as the user interface that is being designed is very easy considering the factor that anyone in the team can use it.

5

**2.5.2 Cost Reduction**

Reduce the number of Person-hours required to complete testing thus saving a lot of effort for the project APIs that are run manually as of now. Preparing an API which is having attributes as many as forty in numbers takes a lot of time. It is done in every release. Normally it takes more than half an hour in preparation of the API itself searching all the data from database. For new resources in the team it takes even more time. A person should even be knowledgeable enough to remember all the tables from which data needs to be taken for those APIs. As sometimes even that is not there in the test case. So it takes time to go through the table’s link and check the various different attributes before finalizing the values for the attributes. There are chances that a tester needs to refer to more than one table for getting the values.

So making the above process automated will save a lot of time as it would be a one-time effort creating an automated script for each API. That script can be used forever for the API with minimal interaction with user until the API gets modified.

**2.5.3 Training**

The solution should be designed in such a way that end user can create automation scripts without the knowledge of any scripting language.

User should be able to use the software with minimal training.

1. **Scope of Work**
   * Analysis and categorization of different kind of APIs available in the system.
   * Creating a user friendly UI for the end user for using the system.
   * Covering Positive as well as negative testing of the API.

6

**Chapter 3 Motivation for Tool**

**3.1 Automated Testing**

"Automated Testing" involves automating the manual testing process currently in use. This section will be limited to API automated testing

Automation refers to the use of such tools which can reduce the need of manual or human involvement in redundant, time taking tasks. API Automation process includes – Creation of test scripts, which have been derived from raw APIs having some default values

A test environment in \*nix operating system with a database is needed such that test scripts are able to be repeated each time there is need to run the APIs

Automation is the basic motivation for creation of the tool and due to listed discussions.[4]

**3.2 When we should do the Automation**

Lot of effort and skills are required for automating any manual execution. It involves a lot of time. The maintenance part plays a major role as with time and new functionalities as per the request from the client gets added into the existing system which makes the automation script obsolete.

Hence some sort of cost-benefit analysis is quite helpful before investing money and putting efforts into it.

1. **Automation is suitable for following types of Testing**
   * **Functional Testing** –concentrates on operations part which performs as per theexpectations.
   * **Regression Testing** –To make sure no side effects on the unchanged part of the systemchanged.
   * **Negative Testing** –Testing with error prone date in the system.
   * **Performance Testing** –to provide assurance that the performance of the system will notbe impacted when run in production environment business projections and requirements.







7

* **Load Testing** –to determine the points at which the capacity and performance of thesystem degrades to the situation that hardware or software upgrades would be required.

In this project, our focus is on first three points, however we will cover last two points also indirectly with some manual intervention

**3.4 Benefits of Automation**

* **Reliable**: Tests perform precisely the same operations with different data each time they are run,thereby eliminating human error
* **Repeatable**: We can test how the software reacts under repeated execution of the sameoperations.
* **Efficient**: Tests run in a reasonable amount of time.
* **Sufficient**: Tests verify all the requirements of the software being tested.
* **Programmable**: We can program complex tests that bring out hidden information

from the application.

 **Comprehensive**: We can build a suite of tests that covers every feature in our application.

 **Reusable**: We can reuse tests on different versions of an application, even if the user interfaceschanges.

 **Better Quality Software**: Because we can run more tests in less time with fewerresources

* **Fast**: Automated Tools run tests significantly faster than human users.
* **Economical**: As the number of resources for regression test are reduced.

**3.5 Areas where Automation can be attempted first**

Highly redundant tasks or scenarios

* Repetitive tasks that are boring or tend to cause human error.
* Well-developed and well-understood use cases or scenarios first

Relatively stable areas of the application over volatile ones must be automated.

**3.6 Disadvantages of Automation**

Though this automation has many advantages, it has its own disadvantages too. Some of the disadvantages are:

8

* Test maintenance is needed which is cost consuming if API is getting changed frequently.
* People with the knowledge of scripting languages are required to debug in case of errors.
* Problems if got while creation of scripts will remain forever if not checked.

**3.7 Trends in industry**

Figure 3 clearly shows how the importance of API calls has been increased in industry . Similar trend has been observed for telecommunication service providers. Since at the back end there are many systems that works together so API correct functioning is important not only for better functioning of the system but for collaborations with external partners.



**Figure 3 :** API Trends

9

**Chapter 4 XML API**

**4.1 What is an API ?**

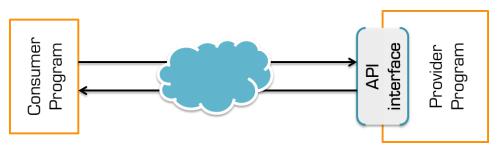
An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers[3]

An API [specifies](http://en.wikipedia.org/wiki/Specification) how some [software components](http://en.wikipedia.org/wiki/Software_component) should interact with each other. It’s an interface as being depicted in Figure 4.

In addition to accessing [databases](http://en.wikipedia.org/wiki/Database) or [computer hardware,](http://en.wikipedia.org/wiki/Computer_hardware) such as [hard disk drives](http://en.wikipedia.org/wiki/Hard_disk_drive) or [video cards,](http://en.wikipedia.org/wiki/Video_card) an API can be used to ease the work of programming [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) components. In practice, many times an API comes in the form of a [library](http://en.wikipedia.org/wiki/Library_(computing)) that includes specifications for [routines,](http://en.wikipedia.org/wiki/Subroutine) [data](http://en.wikipedia.org/wiki/Data_structure) [structures,](http://en.wikipedia.org/wiki/Data_structure) [object classes,](http://en.wikipedia.org/wiki/Class_(computer_programming)) and variables. In some other cases, notably for [SOAP](http://en.wikipedia.org/wiki/SOAP) and [REST](http://en.wikipedia.org/wiki/REST) [services,](http://en.wikipedia.org/wiki/Web_service) an API comes as just a specification of [remote calls](http://en.wikipedia.org/wiki/Remote_procedure_call) exposed to the API consumers.

An API specification can take many forms, including an International Standard such as [POSIX,](http://en.wikipedia.org/wiki/POSIX) vendor documentation such as the [Microsoft](http://en.wikipedia.org/wiki/Microsoft) [Windows API,](http://en.wikipedia.org/wiki/Windows_API) the [libraries](http://en.wikipedia.org/wiki/Library_(computing)) of a [programming](http://en.wikipedia.org/wiki/Programming_language) [language,](http://en.wikipedia.org/wiki/Programming_language) e.g. Standard in [C++](http://en.wikipedia.org/wiki/C%2B%2B) or [Java API.](http://en.wikipedia.org/wiki/Java_API) Web APIs are also a vital component of today's web fabric. An API differs from an [application binary interface](http://en.wikipedia.org/wiki/Application_binary_interface) (ABI) in that an API is source code based while an ABI is a binary interface. [1]

They are good for [users](http://www.webopedia.com/TERM/U/user.html) because they guarantee that all programs using a common API will have similar interfaces.



**Figure 4 : API acting as Interface**

**4.2 What is XML?**



Stands for E**x**tensible **M**arkup **L**anguage

Is a **markup language** much like HTML

Was designed to **describe data***.*

Tags are not predefined in XML. We must define our own tags

Is **self-describing**.

10

**4.3 Working of XML API**

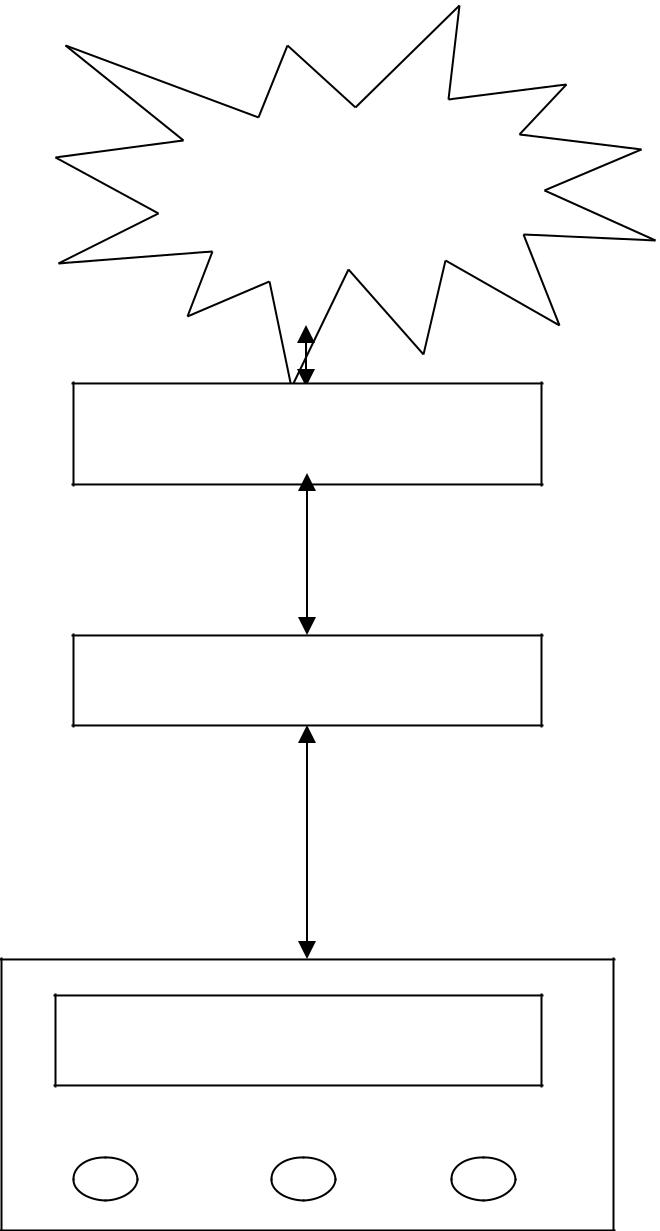
WE have three layers in XML API which are mentioned below in Figure 5

**4.3.1 Extension Layer**

– Interface with Middleware

– Interacts with integration layer

– Always return success



Middleware interface

Extension Layer

Integration Layer

Business Layer

**Figure 5 : XML APIs**

1. **Integration layer**

– Extract data from Node class

– Perform minimum validation

– Interact with BO (can be more than one)

– Populate the output

– Return Success to extension layer

11

**4.3.3 Business Layer**

– Perform data related validations

– Perform business logic validations

– Interact with Database

– Throw exception if any validation or DB operation fails

– Return Output back to integration layer

1. **Why XML APIs**
   * + A single XML server can execute both as a CORBA server and Tuxedo server
     + No IDL files required
     + Factory registration and interface advertising is accomplished dynamically.
     + XML offers data portability and reusability across different platforms and devices.
     + It is also flexible and extensible, allowing new tags to be added without breaking an existing document structure.
     + Based on Unicode, XML provides global language support.
   1. **Testing APIs**
2. **Using testMWBusClient**

– Booting the servers

– Executing the test driver testMWBusClient < apiNameData.xml

Below figure 6 shows the execution of API at Unix prompt



**Figure 6 : Execution from Backend**

**4.5.2 Executing the server directly**

<serverName> -stdin < apiNameData.xml

Below figure 7 shows the execution of API at Unix prompt in standalone mode



**Figure 7 Execution in standalone mode**

**4.5.3 Executing through Browser**

This is shown in Figure 2 where we paste the API is browser session and submits it for processing.

12

**Chapter 5 Design for Tool**

Designing involves problem solving and planning a [software](http://en.wikipedia.org/wiki/Software) solution. This includes both low-level component and [algorithm design](http://en.wikipedia.org/wiki/Algorithm_design) and high-level [architecture](http://en.wikipedia.org/wiki/Software_architecture) design. This is one of the crucial stages in SDLC.

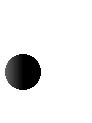
We have divided the tool into four different components. The first one is the parsing component i.e. the XML parser, then we have the Dynamic Controls Generator, after this come the Automation Template in Perl, and the final one is the Script Generator. Apart from discussion on the components design talks about the classification of input which is the core thing of the tool on the basis of which input data for the tool is getting decided. Figure 8 shows the overall flow of the tool.

**5.1 XML Parser**

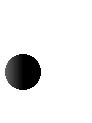
This is the first component of the tool which should get initiated as soon as the user starts the tool. It gets the input XML file and it should extract necessary information out of it after parsing it.

There are five fundamentally different ways to parse XML, the "best" method depends both on the particular development situation we are in and on the style of programming we prefer.

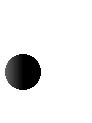
Each technique is based on a different .NET Framework class and its associated methods:



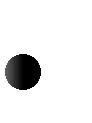
XmlTextReader:- Is appropriate for lower level of abstraction



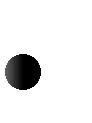
XmlDocument: - Is appropriate if we need to extract data in a non-sequential manner.



XPathDocument:- Is appropriate when the XML file to parse is deeply nested or has a complex structure.

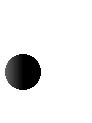


XmlSerializer:- It gives less control over the parsing of XML.

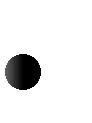


Dataset :It has a very relational database feel, provides medium level of abstraction

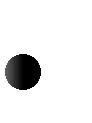
After analyzing all the above different ways for parsing the API, It was concluded that XmlTextReader should be used because of the following reasons.



It is straightforward and effective



Operates at a lower level of abstraction



Programmer can keep track of where we are in the XML file

We have used the read function of XmlTextReader to first read the elements and then used the HasAttributes function to check if the element has attributes as we are interested in only those elements which have some values, others are just ignored. Values for all three namely elements, attributes and corresponding values are being stored as soon as they are retrieved. MoveToNextAttribute function is being used for moving to the next element.

Using the above methods, we should be able to get the required things from the API, from where we can proceed to our next step after parsing.

13

**5.2 Dynamic Control Generator**

This tool is generic tool which is having the capability of automating any API. So there are different APIs which will act as the input for the tool. So tool will have a dynamic control generation capability so that it can generate interface for the user for any size of API. So Dynamic Control Generator has been designed considering this factor into mind to generate controls.

We have used the following controls in our forms which are designed to have interaction with the user[7]

1. **Labels:** - A Label control is used as a display medium for text on Forms. Label control doesnot participate in user input or capture mouse or keyboard events.

We have used it extensively in the tool to provide necessary information to the users mostly near the buttons and textboxes.

1. **Combo Boxes**: - Combo Box is a combination Textbox with a drop-down. Its drop-down listpresents preset choices. Only one list item is displayed at one time in a Combo Box and other available items are loaded in a drop down list. The user can type anything into the Combo Box, or select something from the list.

We have tried to use combo boxes at most of the places so that user has minimum opportunity to type and try to cover maximum choices in the pre-defined list. We have restricted the user to type user defined choices to avoid unpredictable behavior.

1. **Textbox:** -Itlets user’s typeletters and enters data. Many events and properties are availableon the Textbox control. We have used the properties to adjust the textboxes covering all the different requirements of sizes as per the data types of the attributes.

We have given preference to combo boxes at most of the places and where we are not able to accommodate combo boxes, we have used text boxes to get customized data from the user.

1. **Buttons: -** It accepts click events and performs other actions in the user interface. This controlprovides a way to accept input and invoke logic based on that input. Button controls provide a way for users to commit a command or perform an action, like submitting or resetting a form.

We have used it to allow users to proceed to next forms, generate the final automation script, and add new objects into existing form

**5.3 Script Template**

In any automation, we try to collect some set of steps which were performed manually earlier and put them in one place in such a way that there is chain of events between all the events and things start working in an automatic way. So here comes to rescue scripting languages which help us to achieve this task of automation. We have chosen Perl as a scripting language as it’s very easy to modify and syntax has minimal restrictions. Also there are more than one way of doing things in Perl.

This template should be used as a base for all the automated scripts. It should have all the basic things like database connectivity, execution of APIs, putting the input API and its corresponding output results for later verification. It should be written in a generic way to accommodate the different APIs. Environment details should have generic variable so that it can be used in any environment

It should create a script which should have edit permission so that if the user wants any modification in the input file

14

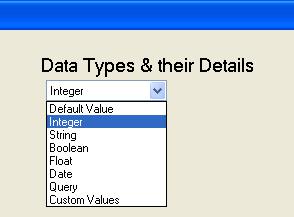
**5.4 Script Generator**

This is the last component in the flow which should use the script template and the values received from the user and generate the final script. It first uses the script template to an extent and then uses the user provided values and transforms the data into the form which is correct in logically and syntax form for Perl.

It will show the user the final output path of the script generated

**5.5 Classification of Data**

We have classified the data into following to cover the possible scenarios.[5] Basic data types are being considered for classification to help user to achieve operations covering most of the things. Information given in Appendix 1 is used to come to below classification. Some of the possible things are intentionally removed as there is very less probability of those values in any of the available APIs. This will save user time too by avoiding unnecessary information. The data types are shown in Figure 7.



**Figure 7 : Classification of Data**

**5.5.1 Integer**

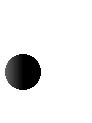


Fixed Length

Maximum Integer Length

* 1. Variable Length
  2. Minimum Integer Value
  3. Maximum Integer Value

1. **String**



Fixed Length



Variable Length

Alphanumeric-Fix

Alphanumeric-Variable

Numbers-Fix

15



Numbers-Var

SpecialCharacters-Fix

SpecialCharacters-Var

All the above have one sub-category that is “Maximum String Length” which will be suffice for covering most of the scenario.

**5.5.3 Float**



Upto 1 Decimal Point

Upto 2 Decimal Point



Upto 3 Decimal Point

Upto 4 Decimal Point

All the above have two further below sub-categories that are



Minimum Float Value

Maximum Float Value

**5.5.4 Boolean**

Since Boolean can always have two values. But those values are not fixed. Those values vary as per the implementer of the API and sometimes as per the client’s requirements. So it has been given as a choice to the end user to put any two values here.

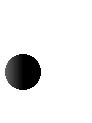


First Boolean Value

Second Boolean Value

**5.5.5 Date**

Date is a non-trivial type of attribute. It has several representations and the way they are being used in API. So we have decided to take the below one’s which are specific to our project. Since there is no specific standard for dates anywhere. We have to list down few only. However with some modification some new ones can be added



YYYY-MM-DD



DD-MM-YYYY

YYYYMM

MMYYYY

YYYYMMDD

DDMMYYYY

DD-MMM-YYYY

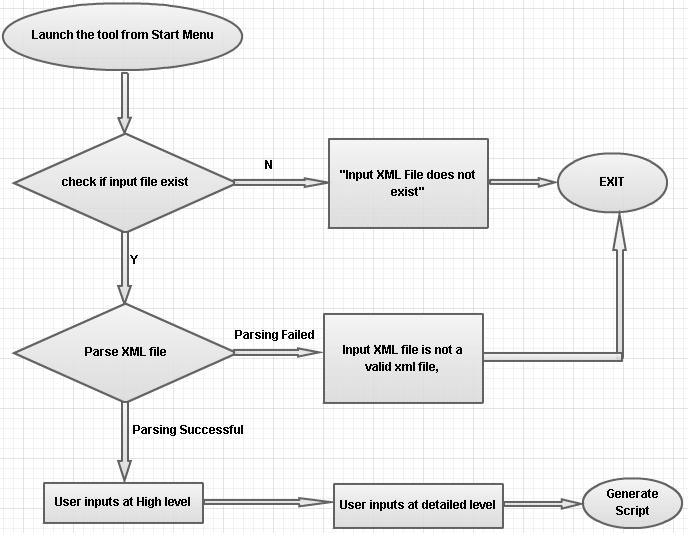
**5.5.6 Query**

This is the most important attribute among all, as this is the place where user can take out data from the database. This will be random data. This part of the tool will cover the different valid values retrieved from database. User is advised to give generic query so maximum data could be checked. User may take advantage of this by providing those query which are not specific to retrieve data from some customer database but to form patterns using the dual table.

16

**5.5.7 Custom Values**

There are chances that we might have missed something in the classification above. So this data type is defined providing user the flexibility to define customized values. User is given the choice to fill up custom values. User is allowed to give maximum 7 different values.



**Figure 8 : Overall flow of Tool**

17

**Chapter 6 Implementation**

**6.1 Base Class**

We have created one namespace named as “API\_Automation\_Tool”. This namespace contain the class that has all the storage elements which are responsible for parsing the XML API which is being passed by the end user. That XML file will be parsed here and is stored in the system which will be utilized by the tool.

Raw data that is being received is classified into elements, attributes and values of those attributes. We have used three different arrays to store the three types parallel at same indices all the three arrays

This class contains the load function which contains the logic of XML parsing, and is being called at the start of execution of the Tool.

**6.2 Dynamic Generation of controls**

**6.2.1 Base Form** : This is the first form which will be shown to the user. It will be having all theelements and attributes from the API provided as input and will be having corresponding combo boxes and textboxes to allow user to take actions

This form has the following functions in its implementation

**createLabelsTextboxes** :- This generic function is designed to create Textboxes on the formdynamically on the basis of input data. It can alone handle creation of labels Textboxes and also combo box for the first basic form which gets open as user start the tool. It will show the available choice to the user. User may select any values from combo boxes or may prefer to keep the default values.

**Next\_Click** :-Clicking on the “Next”button on“Base Form”will call this function which will nowparse the controls on the form and stores the values selected by the user. It will show the further selection combo boxes which has detailed information regarding the values already selected after the initialization of the form.

**Proceed\_Click** :- After the user has selected the detailed choices too, he/she may press the

“Proceed” button which will take the user to a new form.

**6.2.2 Final Form**

This form is generated on the detailed level selection by the user on the first form. It will allow user to provide values in various form. This form is different from the first as it is having information at the lowest level and may have textboxes enabled , disabled based on the last inputs. Some of the textboxes will be disabled here as user might have selected.

18

**Create\_TextBoxes\_2\_form** :- This function is to create textboxes on the second form.

**Generate\_Script\_Click** :- This is where the tools functionality comes to end. As this button willgenerate final script for the input API. This script would be in Perl and can be run in Linux environment.

**6.3 Generic Functions**

We have tried to implement reusability in the code. There are some generic functions which are common for all the forms which any function can use to achieve the purpose.

**ValueSelelected** :- This function is written to check which are the values selected by the user. As weneed this functionality required by multiple forms. This one is written in a generic way.

19

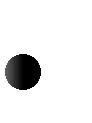
**Chapter 7: Project Details**

**7.1 Introduction**

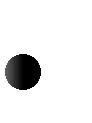
Below Steps define the standards and practices which should be adopted for Automation Tool.

**7.2 Hardware Requirements**

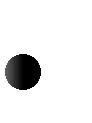
Any Windows machine with matches the following criteria:



**Processor**: Pentium 1 GHz or higher

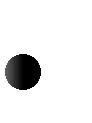


**RAM:** 850 MB for x86 / 2GB for x64

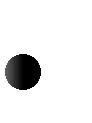


**System** Type: 32 bit Operating System/64 Bit Operating System

**7.3 Software Requirements**



Dot net framework 4.



Windows XP SP3 or higher.

1. **How to run**
   1. Place the input xml into C:\API\_AUTOMATION\input.xml
   2. Start the tool.
   3. Verify the Values on the Base form and click Next
   4. Further select the values categorization & click proceed
   5. Now fill the final details at the detail level & click Generate Script
   6. Now you can verify the output script in C:\API\_AUTOMATION\output
   7. You have the script now, ftp it to the environment
   8. Just type the name and press enter.
   9. Output file will be generated in the same path. Make sure your user id has write permission in the directory

20

**7.5 Compatibility**

The Current Automation framework supports windows platform only.

1. **Features of Framework**
   * + Work for various kinds of API.
     + Provides easy to use interface.
     + Script Generation in Perl language which is simple to understand.
     + Provision of providing custom values as well as system retrieved Values.



* + - Provision of showing API result was success of failure based on certain parameter.
    - Flexibility to user to decide how many times the script should be run
    - Provision of saving all the iteration of the API to be stored as POT for later verification
  1. **Assumptions and limitation**

User of the tool has basic knowledge of the API

API should have the ‘ISO-8859-1’ encoding tag into it.

In case of errors, tool has to be run again.

**7.8 Naming Conventions**

Attributes values are named combining the names of elements and attributes so that end user can differentiate in case there are duplicate attributes for different elements

**For example**

The below API is for swapping price plan (services) for a subscriber.

<envelope>

<body> <inputSwapPlan >

<sourceSvc svcName =”Service 300 Minutes”> <targetSvc svcName = “Service 600 Minutes”>

</inputSwapPlan>

21

</body>

</envelope>

So if we get such API, we have same Attribute name (svcName) which is common for two elements. So in this case, to facilitate the user we are providing the name like below, so it’s easy for end user to understand which attribute belong to which element

1. sourceSvc.svcName
2. targetSvc.svcName

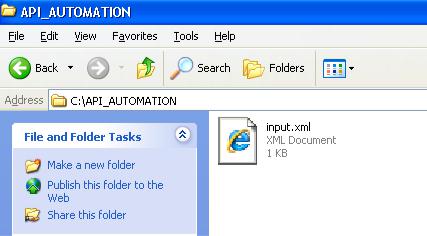
The following Table 1 lists recommended conventions for various controls which are

being used.

|  |  |  |  |
| --- | --- | --- | --- |
| **Control Type** | **Prefix** | **Example** |  |
|  |
|  |
|  |  |  |  |
| Textbox | txt | txtElementAttribute |  |
|  |  |  |  |
| Combo box | cmb | cmbElementAttribute |  |
|  |  |  |  |
| Label | lbl | lblElementAttribute |  |
|  |  |  |  |

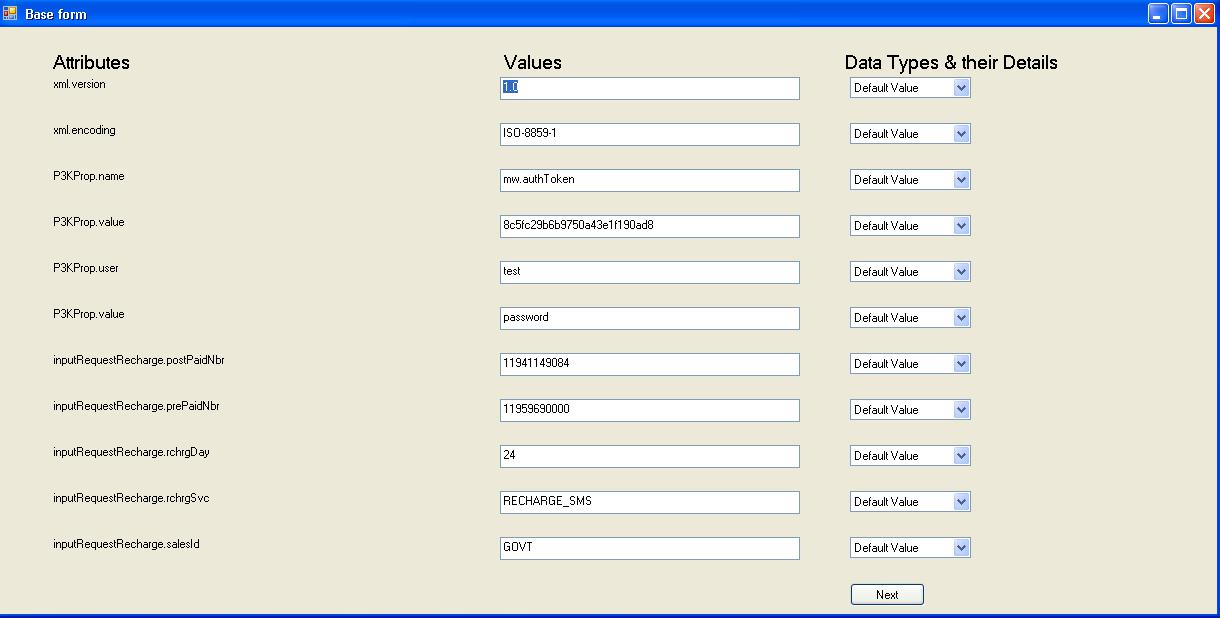
**Table 1 : Naming convention**

The images shown below Figure 10-13 depicts the execution of tool

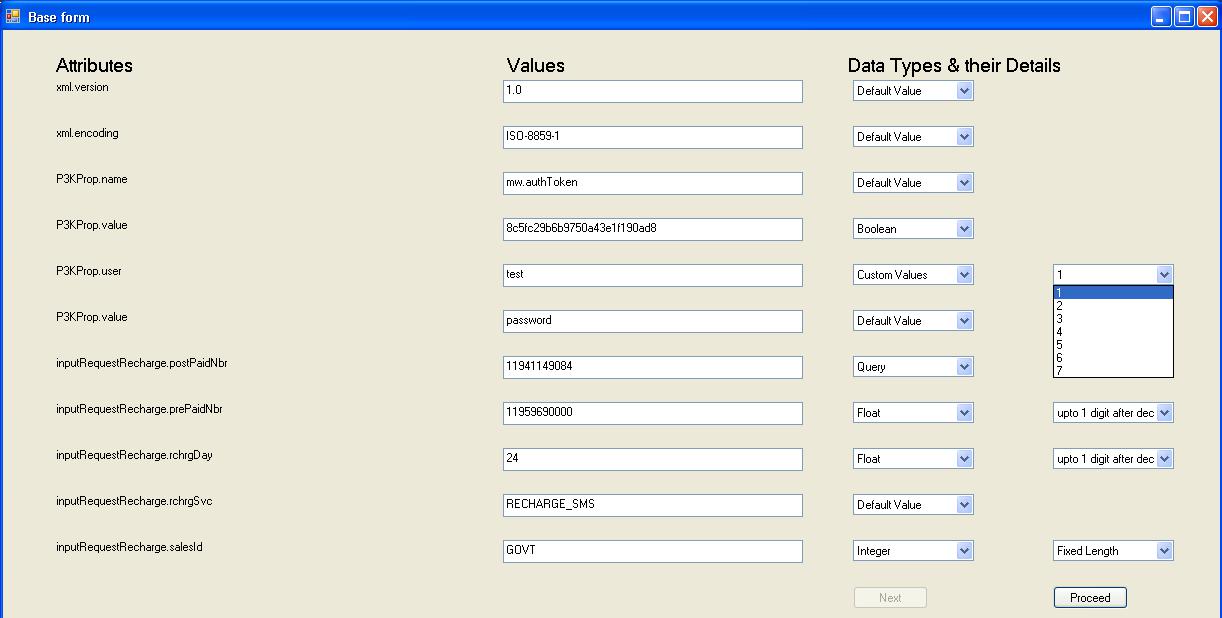


**Figure 9** : Input Path

22

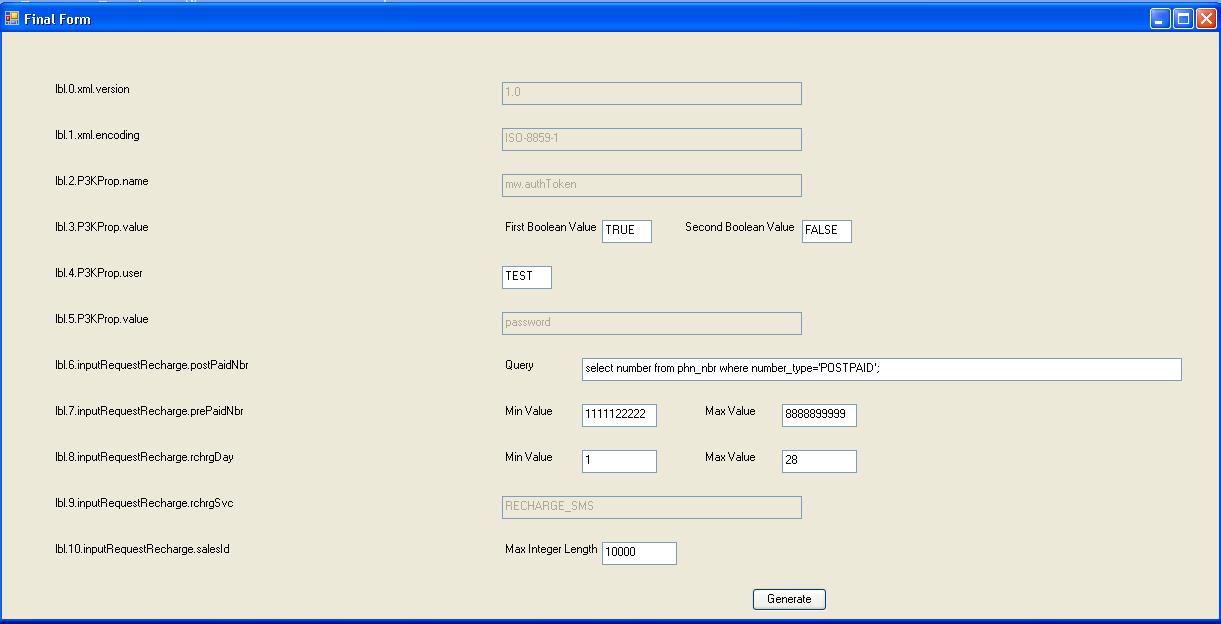


**Figure 10** : Base Form



**Figure 11** : Detailed Level Attribute Values

23



**Figure 12** : Final form

24

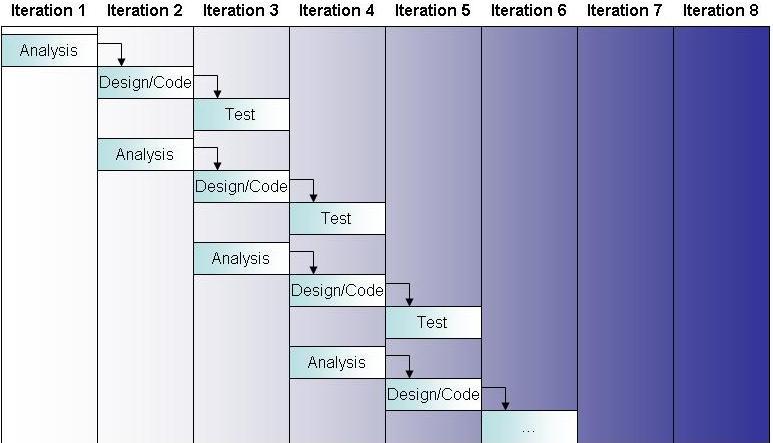
**Chapter 8 Testing and Results**

**8.1 Testing**

As the title of the Dissertation says this tool is for Functional testing, we have made in such a way that to do that in an exhaustive way.

Functional testing is a [quality assurance](http://en.wikipedia.org/wiki/Quality_assurance) (QA) process and a type of [black box testing](http://en.wikipedia.org/wiki/Black_box_testing) that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered. Functional Testing usually describes what the system does.

For this tool, Testing was conducted in phases, as we have implement iterative model where development and testing went on side by side for each module as correctly shown in figure 14.



**Figure 13** : Model for Testing

**8.2 Results**

There are many regression issues that were found that were residing in the system since long. The following are some of the issues which are shown in a generic way

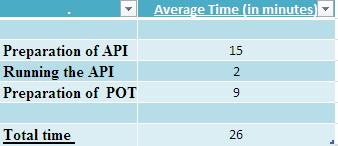
1. Negotiation API failing for a floating amount near to the maximum limit.
2. Some API were throwing wrong error messages
3. Code was shown instead of error messages

25

1. Reference data was not there for many tables.
2. There were time out errors for some APIs for specific set of values
3. Some performance issues were also found during testing.

Apart from these, there are other issues which we are expecting and still it has to be adopted by the whole team and then we will be able to find more issues. We are using it to make system better.

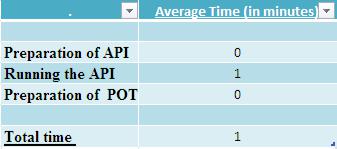
**8.3 Before implementation**



**Table 2 : Statistics before tool implementation**

This time is just for one iteration of the API and will just test one set of values. If a user want to test for more than (say 5), for 5 attributes add 14 more minutes, which will turn around to approx. 40 minutes for running an API.

**8.4 After implementation**



**Table 3 : Statistics after tool implementation**

Now you just have to run the API, system will itself prepare the API, and as well as the POT for the user. User just has to type the name of the script and run.

However there is one time investment of time which for creating the script itself. How much time that will take depends on the skills of the person and how large the API is.

26

**Chapter 9 Conclusion**

In this Dissertation an effort is made to automate the functional testing for the API and to minimize the human effort involved in testing the API. It has been observed that API testing accounts for around 30% of overall testing in the project and rest components of the software also calls API for getting the work done.

So automation has brought into picture to cover this major part and it is successful, there are issues which were there in the system for long were not caught for past ten release are caught and fixed even before coming to the knowledge of client and thus have added significant value to the system.

Regression testing which is done in every release and is time consuming has been taken first and API are automated and thus has reduced significant amount of time for team and is more effective .

27

**Chapter 10 Recommendations and Future works**

This automation tool can be enhanced to do automation directly from the basis of XSD. User need not to provide anything. System will be intelligent enough to read the XSD and prepare the automation script.User need not to provide API’s attributes details at all.

Even provision for providing the Pre-written Query could be given for some general basic attributes which are common in most of the API like phone number, account number, subscriber id.

Feature can be added to join two scripts so that input of one can act as the output of other. Auto verification of output can also be accommodated in future version of tool.

28

**Appendix I**

An API (Application Programming Interface) is a collection of software functions and procedures, called *API calls* that can be executed by other software applications. Application developers code that links to existing APIs to make use of their functionality. This link is seamless and end-users of the application are generally unaware of using a separately developed API.

During testing, a test harness—an application that links the API and methodically exercises its functionality—is constructed to simulate the use of the API by end-user applications. The interesting problems for testers are:

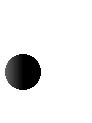
1. Ensuring that the test harness varies parameters of the API calls in ways that verify functionality and expose failures. This includes assigning common parameter values as well as exploring boundary conditions.
2. Generating interesting parameter value combinations for calls with two or more parameters.
3. Determining the content under which an API call is made. This might include setting external environment conditions (files, peripheral devices, and so forth) and also internal stored data that affect the API.
4. Sequencing API calls to vary the order in which the functionality is exercised and to make the API produce useful results from successive calls.

**Category partitioning**

The category-partition technique is a general-purpose functional testing method that goes through a series of decompositions based on characteristics of the input domain. For our purposes, there are three basic steps to category-partitioning: 1) create a set of *categories* that describe properties of inputs; 2) partition the categories into *choices* that enumerate specific values or value-ranges that inputs can assume; and 3) determine constraints among the choices that describe how the choices interact

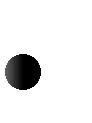
**Step 1** of the category-partition method requires thinking through attributes of this input thathave an effect on the behavior of the routine. For example, pertinent categories might be the *length* of the string and the *content* of the string because both of these attributes will have to bevaried in order to properly test the routine. In general, categories are not detailed, instead they represent high-level attributes of the inputs.

**Step 2** adds more detail to the categories by further partitioning them into specific values orvalue-ranges that can be assigned to the input. For example, the *length* category could be assigned a value of 32 or a range of values, say, and 1-16.

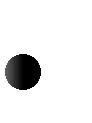


Create a choice for each value or value-range that forces a default setting to occur. For example, if the default is to ignore spaces in the string, then we’d want choices that specify that spaces be present in the input so that this behavior is tested.

29



Create a choice for each value that causes the output to change in some substantive way. Output could be a return value for a function call or visible displayed results. For example, if the length of the string is set to 0, it is illegal and an error message will be displayed. In this case, zero should be a choice for the length category so that this behavior gets tested.



Create a choice for each value on or around distinguishable “boundaries.” Obviously, 0 is again a choice for the length category and so is 1 since it is adjacent to the actual boundary.

**Step 3** consists of writing constraints that describe how choices for one category affect choicesfor another category. For example, a constraint that specifies that an input choice cannot have embedded spaces *and* cannot be of length 0 is required to keep us from considering impossible choice combinations. Conversely, if we want to force certain choice-combinations to occur, then a constraint can also be written to achieve this.

When category partitioning is complete, a single input, like the string example above, might be partitioned into any number of actual values that are necessary for testing. When performing category-partitioning on an API call with multiple parameters, this number can grow quite large

30

**Bibliography**

1. Understanding about API <http://www.3scale.net/wp-content/uploads/2012/06/What-is-an-API-1.0.pdf>
2. This link provided me with examples to avoid small mistakes in coding [http://www.youtube.com](http://www.youtube.com/)
3. Overview of various testing concepts [http://en.wikipedia.org](http://en.wikipedia.org/)
4. Things to consider during automation

[http://www.testpoint.com.au/attachments/093\_Seven%20Steps%20to%20Test%20Automatio](http://www.testpoint.com.au/attachments/093_Seven%20Steps%20to%20Test%20Automation%20Success.pdf) [n%20Success.pdf](http://www.testpoint.com.au/attachments/093_Seven%20Steps%20to%20Test%20Automation%20Success.pdf)

1. Category portioning techniques for attributes of API <http://cm.techwell.com/sites/default/files/articles/XDD2052filelistfilename1_0.doc>
2. Coding related issues <http://www.stackoverflow.com/>
3. Various Concepts in dot net <http://www.dotnetperls.com/>

31

1. 