Webgen Installation and Automation

Submitted By

Shital Tank

14MCEI24

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

December 2015

Webgen Installation and Automation

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Shital Tank

(14MCEI24)

Guided By

Prof. Rupal Kapdi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

December 2015

Certificate

This is to certify that the major project entitled **"Webgen Installation and Automation"** submitted by **Shital Tank (Roll No: 14MCEI24)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering(Information and Network Security) of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-I, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Ms. Ritu Singh
Sr. Application & Software Eng.
T&DP Department,
STMicroelectronics,
GreaterNoida, UP

Mr. Ashu Talwar
Manager,
T&DP Department
STMicroelectronics,
Greater Noida,UP.

Prof. Rupal Kapdi
Guide & Assistant Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Sharada Valiveti
Associate Professor,
Coordinator M.Tech - CSE(INS)
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. P. N. Tekwani
Director,
Institute of Technology,
Nirma University, Ahmedabad

Certificate

This is to certify that the major project entitled **"Optimization of WebGen Installation Automation"** submitted by **Shital Tank (Roll No: 14MCEI24)**, towards the fulfillment of the requirements for the award of degree of Master of Technology in Information and Network Security (CSE) of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Date:
Project Manager
Mr. Ashu Talwar
TRnD Department,
STMicroelectronics, India

Statement of Originality

I, **Shital Tank**, Roll. No. **14MCEI24**, give undertaking that the Major Project entitled "**Webgen Installation and Automation**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering**(Information and Network Security) of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student
Date:
Place:

Endorsed by
Prof. Rupal Kapdi
(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Rupal Kapdi**, Assistant Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. P. N. Tekwani**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Shital Tank
14MCEI24

Abstract

Webgen is memory installation system for making or generating libraries for particular memory. This installation process is very lengthy and complicated task. Installation requirement is generated by customer and the constraint is that there is need to finish installation within 48 hours. For completing this process with less time and burden Automation process has been implemented. This automation is working for small set of inputs. It is required to have modification of this automation process because even though we have this process automated due to changes in input the automation is failing to give desired result. We still need to perform this task manually. Changing automation process some new functionality is required to be incorporated and testing of this modification has become necessary. In this development and enhancement process we are giving effort to make system to give better performance and error rate reduction.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Webgen is project developed for customer who belongs to this company only. This project is web based application provides user interface to help the customer to fulfil their requirement. This Webgen system is basically aim towards library generation task. Our team is working on memories providing layout and parameters of memory. And for checking this specification memory generation installation system has developed. This system is taking input which are variety of information about memory to generate library. This library is blueprint of memory which will be fabricated over the cheap. The parameter will be set on the basis of customer requirement for eg: size, number of pins, operating conditions and number words, etc. these are parameter which defines configuration of memory. Library is containing configuration, products and scripts to run all of these. These all parts are packed into one package and its called one library. When any requirement of library changed its primary configuration will change and that will generate memory installation requirement on Webgen. This installation process helps to generate config for memory. This config will be verified by verification platform and in the last stage this config will be imported into one of the library related to technology of memory and various parameter required to be set on the basis of customer requirement and at the last the library will be generated.

1) AIM of the Project

The aim of project is to make optimization and enhance efficiency of memory installation process. This memory installation and generation process is one of the important tasks to enhance for customer satisfaction. Webgen portal is providing web based interface for memory installation task. Automation of this task is available for limited inputs as more inputs are discovered. For satisfying more inputs need of enhancement of current automation system is necessary.

2) Bottlenecks of current system

Automation of memory generator system is not giving desired output for inputs discovered. This automation is fulfilling all basic requirements and inputs. The change in customer requirement some new inputs and requirements are discovered. To fulfill these requirements manual task is needed. Manual process is very time consuming and error prone. Enhancement of this automation is necessary to satisfy customer requirement within 48 hours. These enhancements are necessary to make system robust and universal to accept any kind of input.

Chapter 2

Literature Survey

2.1 System Overview

Memory generator installation system is used to generate various types of projects which are Memories, Bists, and Verification. This projects are having different purpose.Webgen is application which is representing the GUI for this installation system. Memory Generator is one of the functionality provided by this application portal. When customers are changing their libraries the primaries will be generated of this library and according to this primary one requirement of config deliverable is generated.[1]

Config deliverable is compiler installation request which is need to be satisfied by our team. Using some kind of manual work and some kind of automation this task needs to be completed.[1]

The scope and role of Webgen is as follows

- This memory installation system is only accessible to the customers who are working in our company or internal person.[1]
- Webgen is running over Webgen compute farm which are interlinked and exchanges inputs.
- It is web based portal we can use it using any web browser chrome, Mozilla or internet explorer.[1]

2.2 Architecture and modules of Webgen

The Webgen architecture is based on four main modules[1]

1) Compiler installation

2) Compatibility

3) Cut Generation

4) Publish

There are many other modules which are developed to support the functionality of these modules[1]

2.3 WebGen Compute Farm

[1]

Apart from these modules to support the functionality of Webgen and to give platform for running Webgen on more important part is compute farm. This compute farm is Cluster of interconnected computers provides service as server system. These computers are having big role in all functionality of Webgen. While doing compiler installation the job is queued into one of the request processor waiting for processor to serve them. This request processor is one of the part of the architecture of webgen other then this request processor other processing servers are also part of Webgen architecture.

2.4 Pre- requirements

[1]

Moving towards exploring functionality of webgen first is compiler installation, before initiating functionality of Compiler installation several task need to be performed on back end side which are as follows

- We need to have products installed on webgen which are required to generate particular compiler
- We need to have generated input files to fed the data necessary to install config

2.5 Exploring each modules and terminologies

[1]

1) Compiler installation The main terminologies related to compiler installation are as follows:

Product Repository(PRS):It stands for PRODUCT TRACKING. This system is central repository of The IP which are IO, standard cells. This IPs are ready to be used in production. We can get these IP deployed in UNICAD product tracking system using website of PRS or using SQL commands and Web Services.

Config : its package which holds products which is used to developed blueprint or schematics for a particular memory chip.

Maturity: this maturity levels represents levels of or sets of different validation system. By setting maturity level we can confirm the validity or maturity of the particular configuration or compiler.

There are more terminologies related to these modules:

PRS configuration name: This name uniquely identifies particular configuration.

Execution constraints: This is used by request processor to identify how to process particular compiler to complete compiler installation and to generate configuration.

2) Compiler compatibility Verification Environment: this represents different verification platforms. We need to set particular one to check validity of our compiler installation against this platform.

3) Cut Generation Cuts and operating condition: This represents the memory cuts (eg: numbers of pin, views) and operating conditions for memory for example: pressure, voltage, etc. These parameters are important to set while generating libraries which need to be fabricated on memory chip.

Apart from this WebGen Architecture contains some supportive modules: product management, Cut status, Technology management, Designflow management, Generated Libraries, News etc.

Details of above modules will be inside working of these modules.

2.6 WebGen Development Architecture

[1]

Webgen has been developed according to Model View Controller architecture. This architecture is having three main modules which are interconnected with each other as follows [2] As above figure by splitting the web application into 3 parts application will have modular and efficient collaboration between them. We can efficiently incorporate three goals development and design of application independently and integrating them. [2] MVC follows a complete cycle: when user triggers any action (eg: Modifying or Getting data) the controller transfers these request to particular model and model will request the data accordingly and gives response. The controller will decides views according to response by model and appropriate view will be delivered to the user.[2]

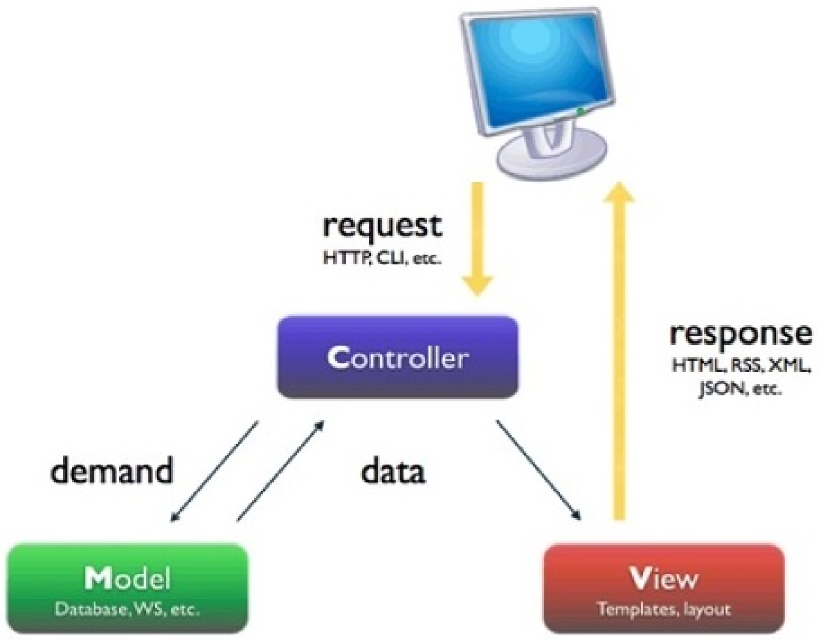


Figure 2.1: MVC Architecture

1) Model:

The main business logic will be incorporated here. The logic to modify or delete the data from database, It is direct communication point with database. This module will separate the business logic form presentation logic so the developer can separately work on the module allocated no changes will be required to made on presentation side. Other than this data management task more logic can be implemented here which are Session management, Authentication and Security. This part could also serve as helpers for other part. As this is separate logic from other parts this could serve as reusable model.[2]

2) View:

The presentation logic of web application will reside here. The developer now has control over the way how the data represented against user. User interface will be decided by this layer. This layer can be frequently changed according to the user interest or for other reasons (e.g. Logo changes) will not affect the other logic of application.[2]

3) Controller:

This embeds functionality to handle the user request. When user triggers any action using View (user interface) this request would come to the controller. Controller will have logic to decide which logic to implement here. The controller sends necessary data and requests appropriate model to perform necessary action or logic. Model will give response to controller based on this response controller would prepare particular response and decides which view will be given to the user.[2]

2.7 Flow chart of MVC incorporated in Webgen

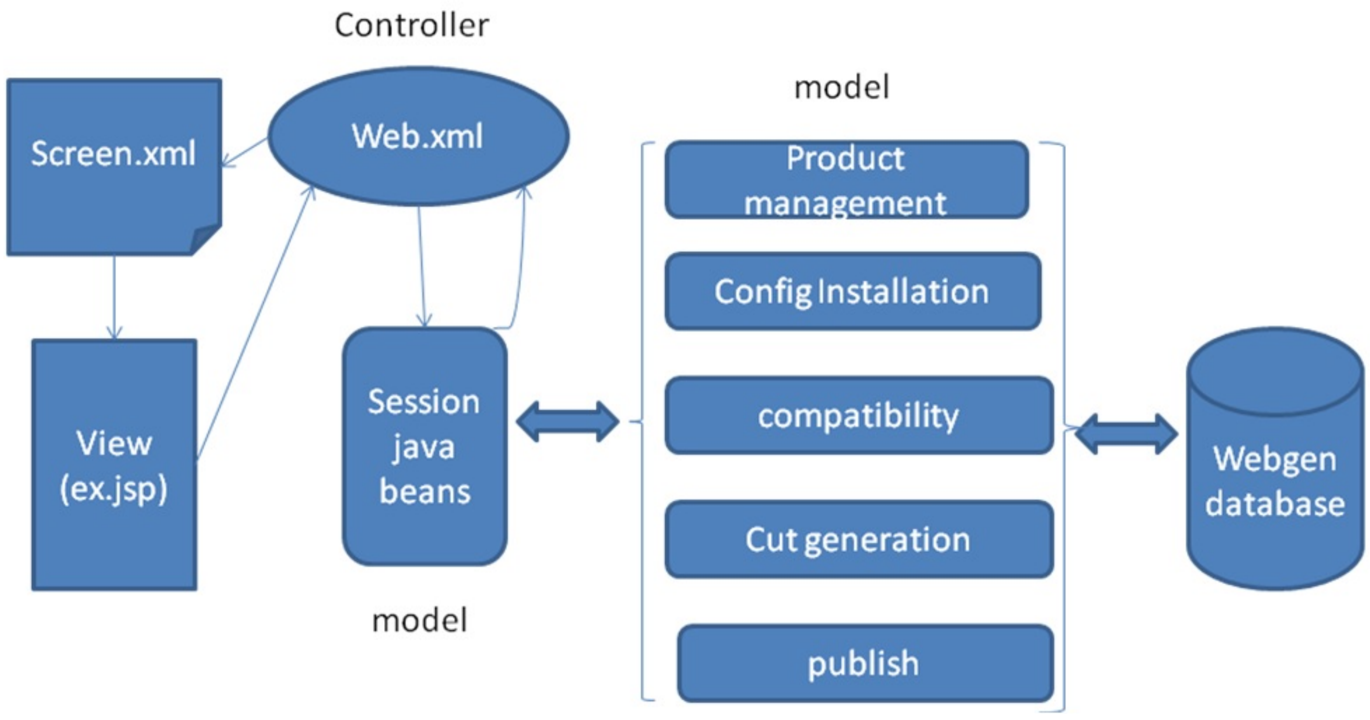


Figure 2.2: MVC in WEBGEN Architecture

For developing Webgen using MVC java based model has been implemented. The JAVA components used are:

JSP: java server pages which used for developing user interface. The page represents view part of webgen.[2]

Servlet: This is used to implement controller part of Webgen. it will runs at server side which is web server.[2]

JavaBeans: Beans are container to give us the base to create our business logic inside. The business logic to access database and function to communicate with database is contained within this Bean. This EJB container (Enterprise java beans) provides some inbuilt functionality to implement business logic.[2]

In WebGen implementation of main business logic which is config installation, compatibility, etc. are implemented in model part.

2.8 Working of WEBGEN MVC Architecture

[1]

The application content represented to user side using jsp pages. All the presentation logic resides here. When user makes any request or triggers any action using jsp page that request would be handled by their action classes which are implemented using servlet technology.

The mapping of jsp to its corresponding action class is handled by controller which is xml file contains mapping between these jsp to action class. When corresponding action class found by controller this request is further goes to its EJB class based on which action needs to be performed and EJB will execute their corresponding business logic. Response of this process handled by controller and corresponding view will be identified by the controller class and at the end user will get their desired response.

2.9 Webgen Development Stages

[1]

Webgen development process is divided into three hierarchies as follows.

Webgen Development stage: When developer starts Webgen development or modification the code will be deployed on to Webgen development server. The code which is in this stage will be used by developer only not by customer.

WebGen Quality Assurance Stage: When development is finished without any compilations and exceptions the code will be deployed for QA test here code is tested. The test is conducted to identify whether it meets customer criteria or not.

Webgen Production stage: when Webgen code passes all kind of test the final product is ready to be deployed for customer use. This WebGen when deployed onto production server it can be use for live customer requirement.

2.10 Current working

WebGen manual working for installation request:

The starting point of Webgen installation process is not using user interface. There are several numbers of tasks needs to be performed before installation actually starts.

As depicted in diagram the starting point of the Webgen installation process is when

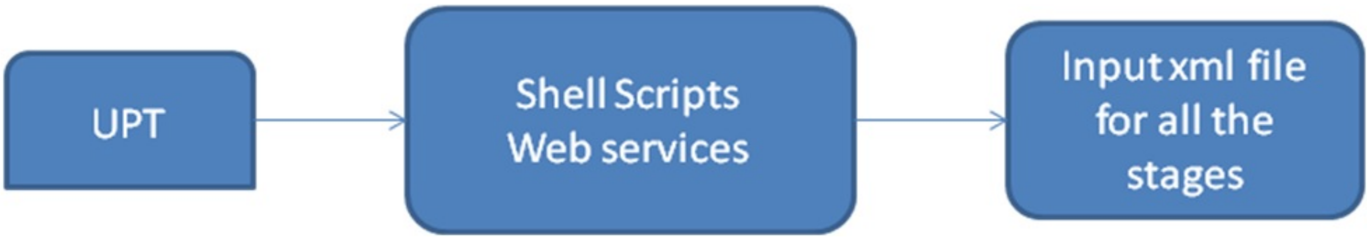


Figure 2.3: product installation

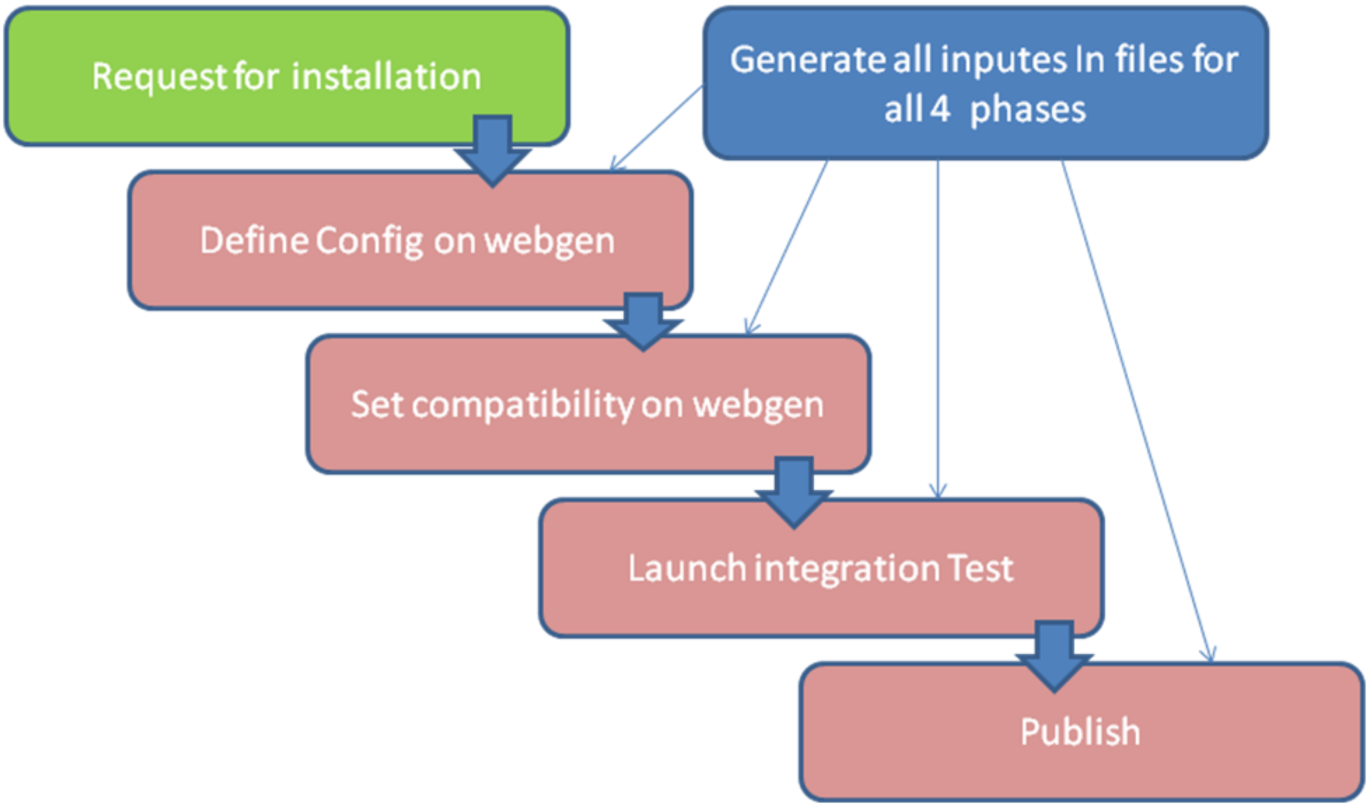


Figure 2.4: Stages of Webgen with input file

some customer changes product or primaries on PRS.

When primary on PRS changes it will trigger the scripts written in shell that initiates the process of xml creation. These scripts are responsible for all the checks required to be made. The scripts first checks primaries changed after some pre decided date. If any changes are there then this scripts initiates very important process.

This process is product installation from PRS to Webgen corresponding to its primary. This product triggers next step which is creating directories for logging the activity, setup area, then making on file for product declaration on Webgen which contains name and version of products. Once this process completed then next step would be creating input files for each and every stages at once.

As depicted on below figure Webgen config installation process will take place after input in xml format is available. When process of input generation completed we get mail which is a request for compiler installation and it also contains path where all of these input files stored.

2.11 Manual Process Description

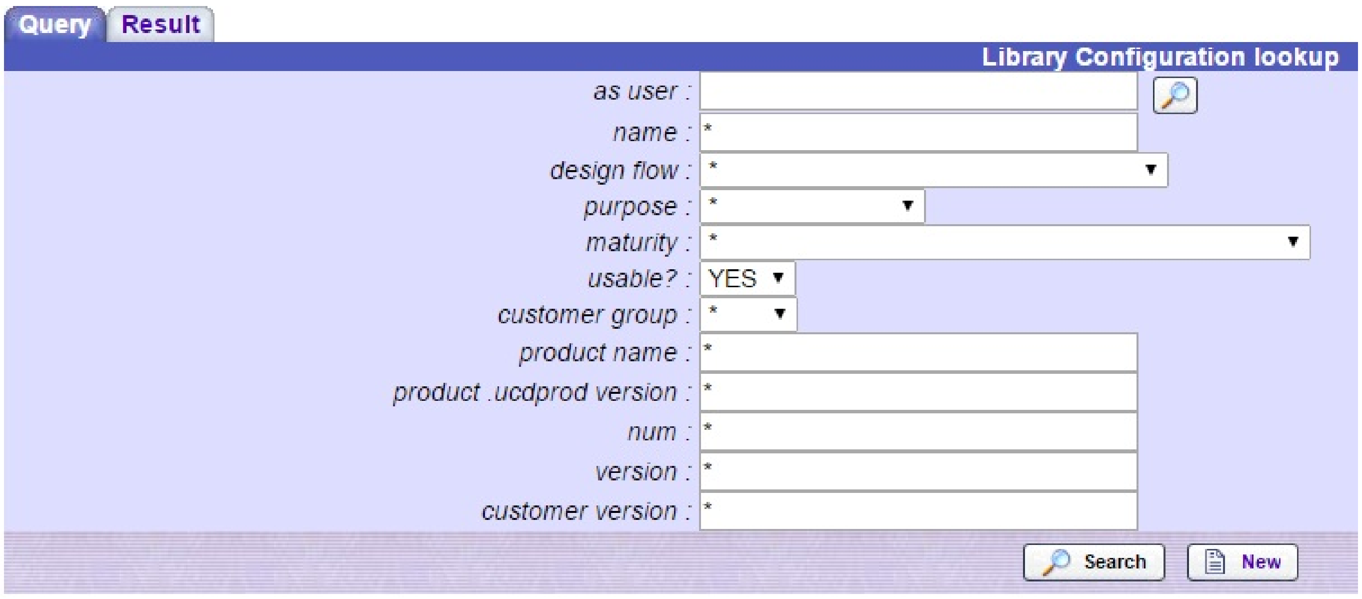
[1] **1. Config Installation**

Expectation:

The necessary things should be taken care while installing config on webgen are as follows:

- Asik kit, Library filter and flow type are mandatory, views filter is optional
- Technology present in PRS should match with webgen
- Maturity model of both PRS and WebGen should be aligned.

First step of this process is as follows:



as user :	
name :	*
design flow :	*
purpose :	*
maturity :	*
usable? :	YES
customer group :	*
product name :	*
product .ucdprod version :	*
num :	*
version :	*
customer version :	*

Figure 2.5: Installation Config Search

1) Search the library config: To check config is already exists or not we need to search it using name, and design flow other fields are optional to specify. In interface we need to specify necessary fields and then click on search button.

The result of this action would be either empty or list of config that are already installed.

2) Edit or create: of no result found we need to create new config using new button. If there is desired entry in the list then choose the latest one and start editing by clicking EDIT button.

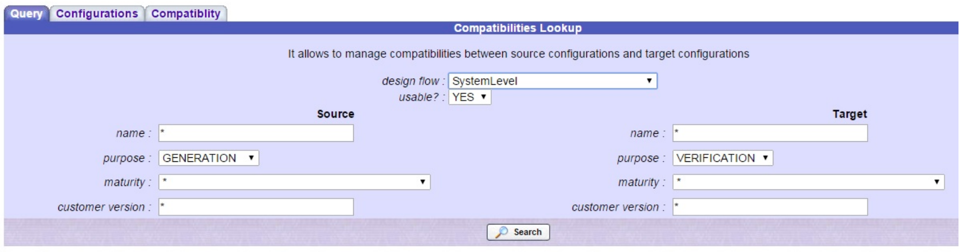
The first tab will be config info, the fields required to fill are config Name, PRS name, Design flow, Maturity, Purpose, exec constraints.

The next one will be identifier they are flow type, view filter, asic kit and library filter, must edit field is the value of PRS parameter generator parameter changed.

The last tab is to specify products, we can just put the products in webgen by using .productfile which is generated when we have installed the products on webgen. Click .productfile button and copy this product file content and just past in popup box. SAVE your config.

After saving config you will be redirected to another page where we can check whether or config is properly installed or not. Here our request is processed by request processor and then result of generation will be displayed as a field GENERATION. The answer ok means generation of config is successful and we can use it into our later stages.

2. Configuration Compatibility Check



Compatibilities Lookup

It allows to manage compatibilities between source configurations and target configurations

design flow : SystemLevel
usable? : YES

	Source		Target
name :	*	name :	*
purpose :	GENERATION	purpose :	VERIFICATION
maturity :	*	maturity :	*
customer version :	*	customer version :	*

Figure 2.6: set Compatibility

This check is performed to identify compatibility of our configuration with other config version.

The link config compact is used to open this lookup page. First we need to set design flow according to our primary config. The next one is the config name which is our primary product name for which we are checking compatibility. Set source and target purpose and choose search.

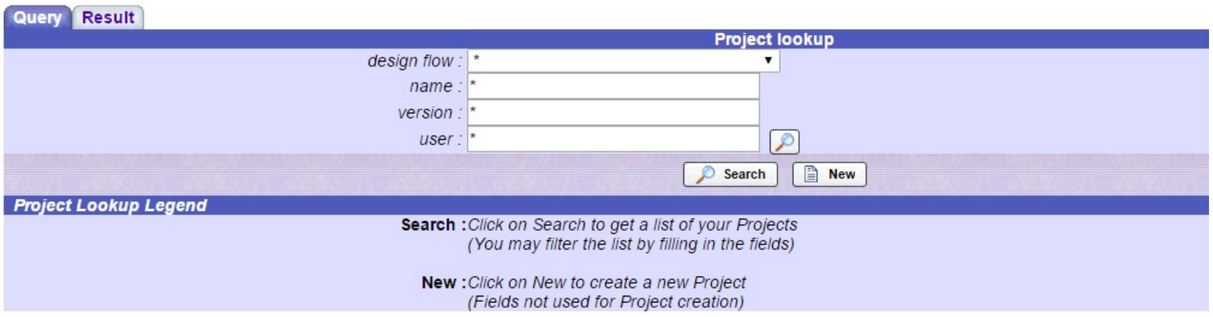
If result of search is empty then next go for integration test where we are generating libraries. Else we need to set compatibility by selecting version and config name. The values are needed to be fetched from PRS. Search deliverable name on PRS site and select primary form the list. Then click on the field coordinator ok for checking verification platform name. Set yes if any platform name specified on PRS.

3. Cut Generation

The name of the project should be: TESTS cut $(con_gurationdesignowname)$

Select design flow and enter name of project as *TESTS cut* then search for project. If result is empty then create new project. Go to memories tab if purpose is generation. If project is new then add library and select appropriate on from them if present. Else the select one of the existing library for editing purpose.

If library already exists then choose appropriate library and click on the version specified. By clicking on it one pop up box will be open it contains list of library config with version choose the version which we have installed recently.



Project lookup

design flow : *
name : *
version : *
user : *

Project Lookup Legend

Search : *Click on Search to get a list of your Projects*
(You may filter the list by filling in the fields)

New : *Click on New to create a new Project*
(Fields not used for Project creation)

Figure 2.7: Cut Generation

2.12 WEB SERVICES

SOAP: It is based on three parts service provider, service registry and service requester.[3]

- Service provider will develop web service as soap based, and then after the description of service will be published over the registry comprises the all web service repository.[3]
- Service registry is the central repository of web service this enables online discovery of web service created.[3]
- Service requester is the consumers of web service who can find the web service from the central repository using query and fetches the description. This description is used to bind with the implementation of service. After binding this service description consumer can start communicating with service.[3]

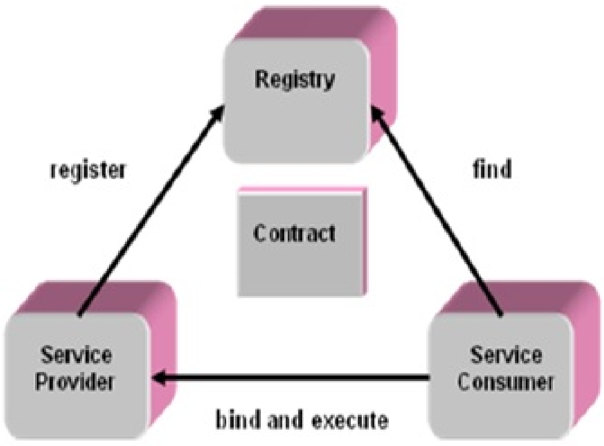


Figure 2.8: SOAP Architecture

[3]

These entities are communicating with each other using XML and SOAP protocol. This communication is held over HTTP. Web service Description language is used for binding describing web service. The description includes many things like method usage its purpose and information that how to bind it with implementation.[3]

REST: It is not protocol it is Representational state transfer and based on client server Architecture. Web Application Description Language is used to describe web service. The content it includes is URI of the resource and how to use it.[4]

It is based on request response style, here client is sending request and server first process its request and gives response.[4]

Main principles of REST are

Address ability, Uniform interface, statelessness. They are as follows[4]

- Address ability is implemented with the help of URI (uniform resource identification). Data will be directly represented as resources and addressed by URI. The resources can be variety of information but should be referenced by URI.[4]

  In other term resource is document which encapsulates current state or desired state of resource.[4]

- Uniform Interface: Rest uses HTTP protocol for communication so interface would be HTTP based and for accessing resource it uses HTTP methods which are GET, POST, PUT, and DELETE. Based on which operation is required the method is chosen and given as request.[4]

- Statelessness is the ability of rest that it will uses only fresh request for Resource identification and for method to perform. Old request is not having any kind of validity.[4]

  To retrieve, create and modify resource Restful web service uses four methods:[4] [5]

  Comparing performance of web services:

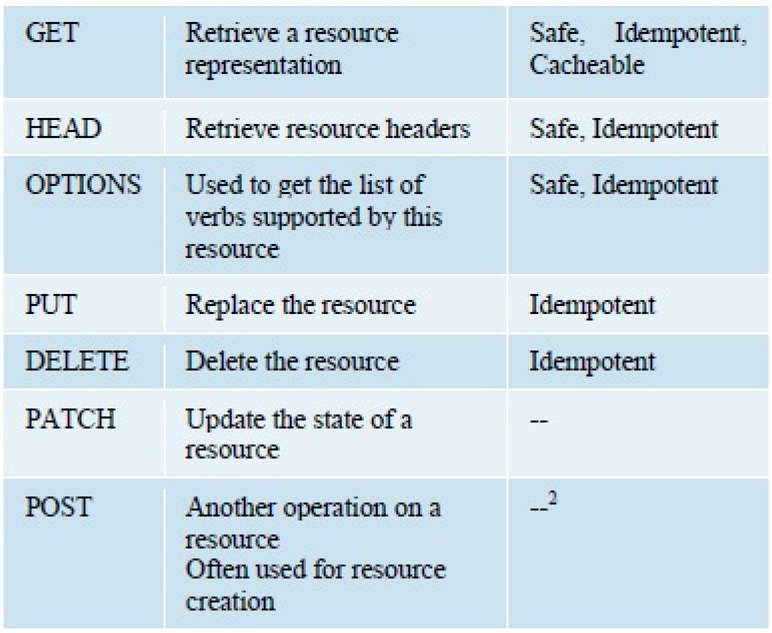
  Bandwidth: SOAP consumes more bandwidth because it uses xml for message communication. Encoding and decoding of xml messages consumes large bandwidth. Whereas restful is communicating over http so it give us better performance. [4]

  Throughput and response time is better in case of REST.[6]

  As per conclusion Restful web service giving better performance in terms of throughput, bandwidth, development time and response time we are moving forward to implement this Restful web service for our functionality.

  Webgen installation process using web services

  As depicted in figure 2.4 we can get this input file from path stored in mail. The process of Webgen using web services is very easy one. There is script provided



GET	Retrieve a resource representation	Safe, Idempotent, Cacheable
HEAD	Retrieve resource headers	Safe, Idempotent
OPTIONS	Used to get the list of verbs supported by this resource	Safe, Idempotent
PUT	Replace the resource	Idempotent
DELETE	Delete the resource	Idempotent
PATCH	Update the state of a resource	--
POST	Another operation on a resource Often used for resource creation	--[2]

Figure 2.9: REST Methods

with us which contains URL of the web service and all the necessary input checks which is requires for smooth processing.

The URL contained within file lead us to Webgen application and the process is handle by the Webgen as per they implemented web service. The script file is also contained name of the resource required and method (GET, POST)

Chapter 3

Current Automation

The whole installation process is divided into 5 stages. These stages are automated for config installation to publishing of library using RESTful webservice and they are working properly. This automation has successfully reduced installation time and also the burden of manual process. These stages are individually implemented with java Restful webservice.

Install products on Webgen was already implemented before automation process and is able to run successfully with new requirement. The second stage which is config creation is taking input as xml file which contains parameter for configuration and generates config only if it is successfully completed then next stage we can run set compatibility webservice which also takes xml as input and gives output if it is successful. The other stage will be used in same way.

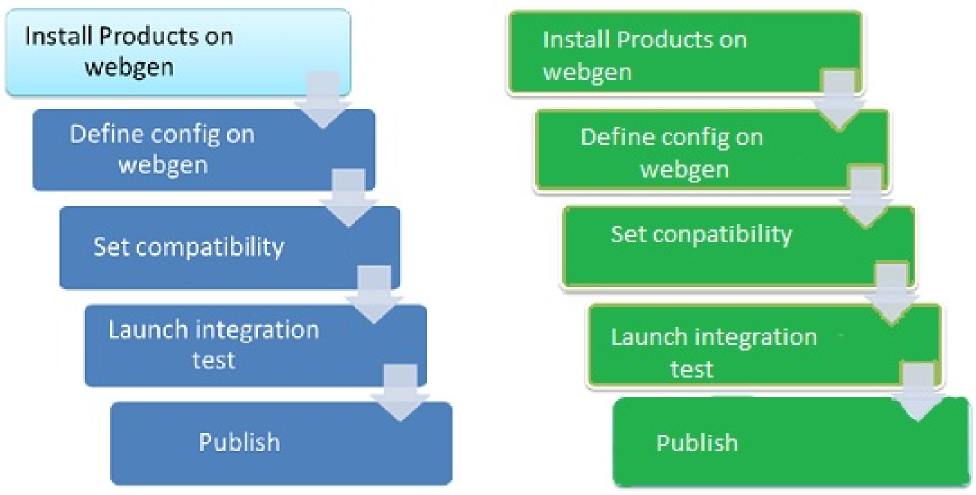


Figure 3.1: installation stages

Problem in current automation:

The main problem in current automation is that it could not work with new or modified input. The automation is compatible with old inputs. The customer requirement for specification of memory Ex: specifying numbers of pins and some more constraints have been incorporated for memory installation system. This requirement is fulfilled by manual process because the web service implemented is highly specific for some set of inputs. As the input has changed there is need to modify this automation.

There is some modification still needed in this web service for reducing error rate and for increasing efficiency so some of the functionality is required to be implemented which is not present in current automation process. This automation is failing for new inputs so we are not currently using this automation because when customer requirement for installation will come we need to fulfill it on urgent basis. This installation process should be completed within 48 hours after the requirement has submitted to us.

Chapter 4

New Automation Ideas and Implementation

The new automation idea for this installation process to increase efficiency and decrease error rate leads to change in some functionality. The change in functionality is that instead of making xml file prior to whole installation automation process this file could be generated stage wise. The reason behind this modification is that the in some cases the inputs of next stage is depended on the output of the previous process.

The flow chart of this idea is as follows

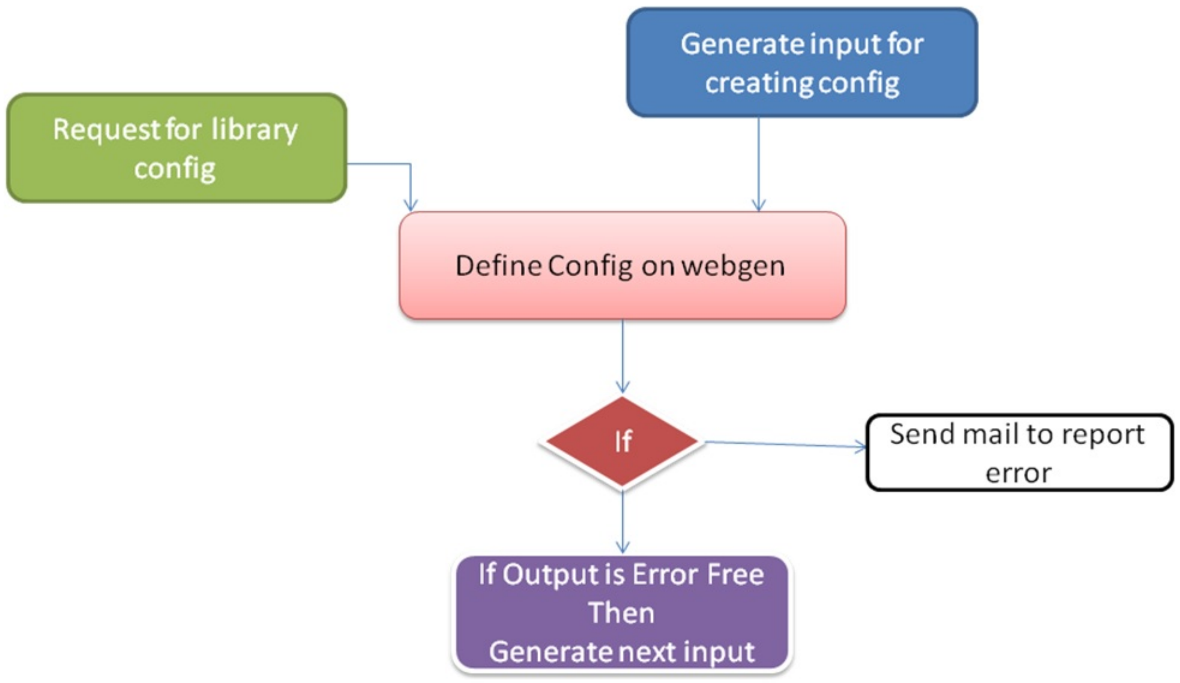

Generate input for creating config
Request for library config
Define Config on webgen
If
Send mail to report error
If Output is Error Free Then Generate next input

Figure 4.1: New Automation Idea

Efficiency: efficiency is increased by reducing work done by the functionality. If one stage is giving error we will not compute the input file for the next one so it increases performance and reduces time.

Error rate reduction

For error rate reduction some functionality was required to be implemented. There is need of implementing two new functionality.

Config search functionality: this task is required to be checked properly before we are going to create config automatically, because manually we can check for every new scenario comes but when it is in automation everything we need to take care for reducing error possibility which can directly affect customer experience and satisfaction for completion of whole task. The automation would be used directly on the live system. The error can propagate from one stage to other and if it is passing all stage can lead to major problem or error which is very hard to correct or modify.

Product search: this functionality is needed when webservice is going to start installing products from UPT to webgen. This is starting point of installation process which will reduce the error rate and gives better performance. If product is already present then no need to install it second time it will save time also.

The config search functionality implementation is given step wise.

4.1 Modeling restful application:[4]

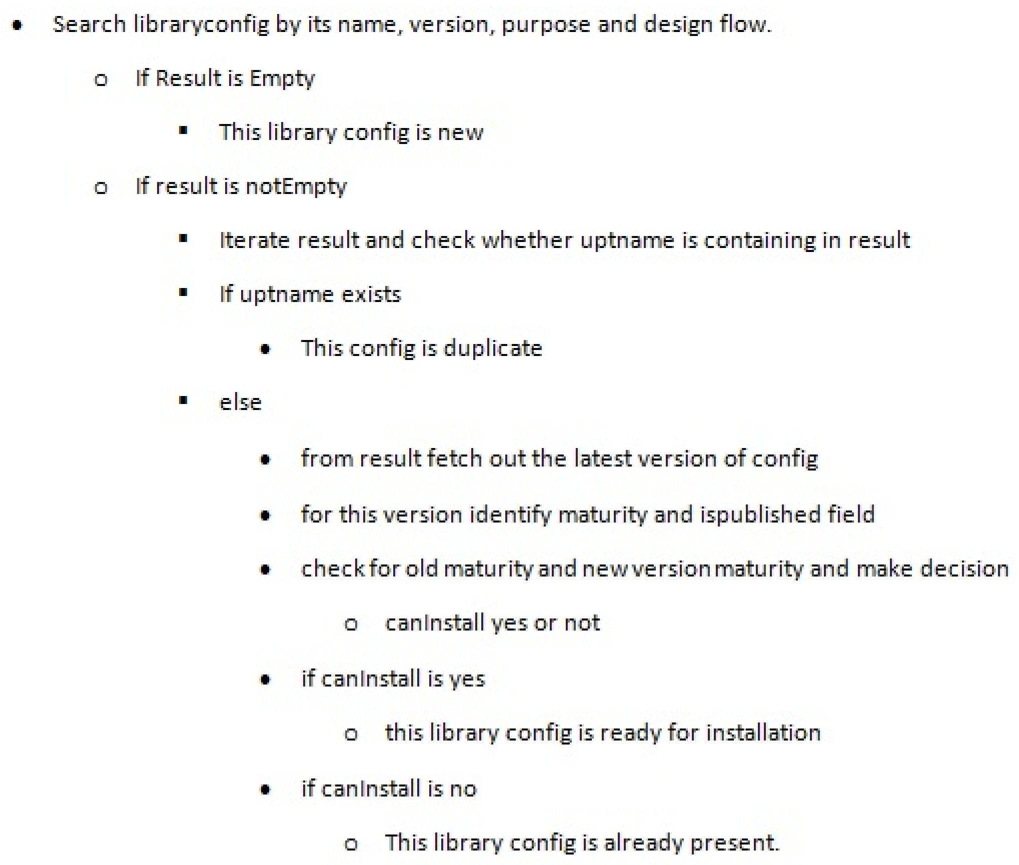
Complete framework of Restful web service for library search functionality.

Whenever any library config request will come to use we are first searching for this library whether it is already present or not. We are checking by using search functionality of web application and if it is already there we select latest version and we try to modify it.

The main logic behind this is two libraries with same version and uptconfiguration name should not present in the Webgen. Other than this if two requested config version is already present in the Webgen then we need to make several checks.

While installing config with already present version we need to take care about whether it is already published.For some types of maturity library config of already existing version could be installed and for some it cannot.

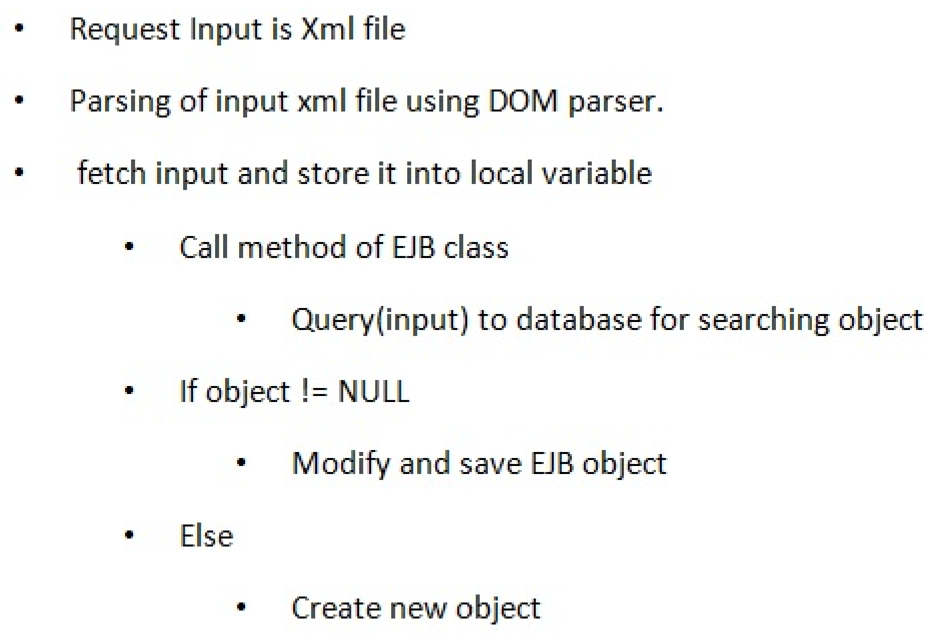
The algorithm for searching is as follows



```
• Search libraryconfig by its name, version, purpose and design flow.
    o If Result is Empty
        ▪ This library config is new
    o If result is notEmpty
        ▪ Iterate result and check whether uptname is containing in result
        ▪ If uptname exists
            • This config is duplicate
        ▪ else
            • from result fetch out the latest version of config
            • for this version identify maturity and ispublished field
            • check for old maturity and new version maturity and make decision
                o canInstall yes or not
            • if canInstall is yes
                o this library config is ready for installation
            • if canInstall is no
                o This library config is already present.
```

Figure 4.2: algorithm for config search

Web service framework implementation algorithm:



```
• Request Input is Xml file
• Parsing of input xml file using DOM parser.
• fetch input and store it into local variable
    • Call method of EJB class
        • Query(input) to database for searching object
    • If object != NULL
        • Modify and save EJB object
    • Else
        • Create new object
```

Figure 4.3: algorithm of Webgen development architecture

The first step is to implement the web service functionality into our application. I want to implement the library search functionality so first I have to make an algorithm for based on several test cases I have conducted.

Flow chart of web service architecture of Libraryconfig search functionality

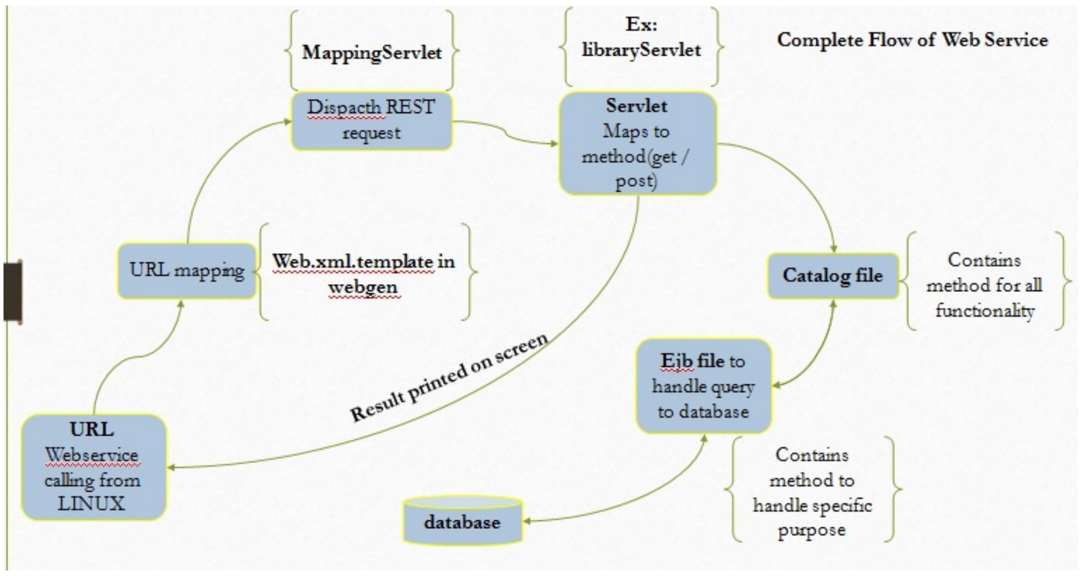

MappingServlet
Ex: libraryServlet
Complete Flow of Web Service
Dispacth REST request
Servlet Maps to method(get / post)
URL mapping
Web.xml.template in webgen
Catalog file
Contains method for all functionality
Result printed on screen
Ejb file to handle query to database
URL Webservice calling from LINUX
Contains method to handle specific purpose
database

Figure 4.4: architecture of Libraryconfig search functionality

The first and major step was making logic which is to be implemented is over. The next is configure our application so that it can be used by other application for that we need to implement several steps.

URL Mapping: web.xml is the mapping file which contains mapping for all of the servlet present in the application. For mapping search functionality with URL we need to make one entry as below.

Web.xml.tempalete

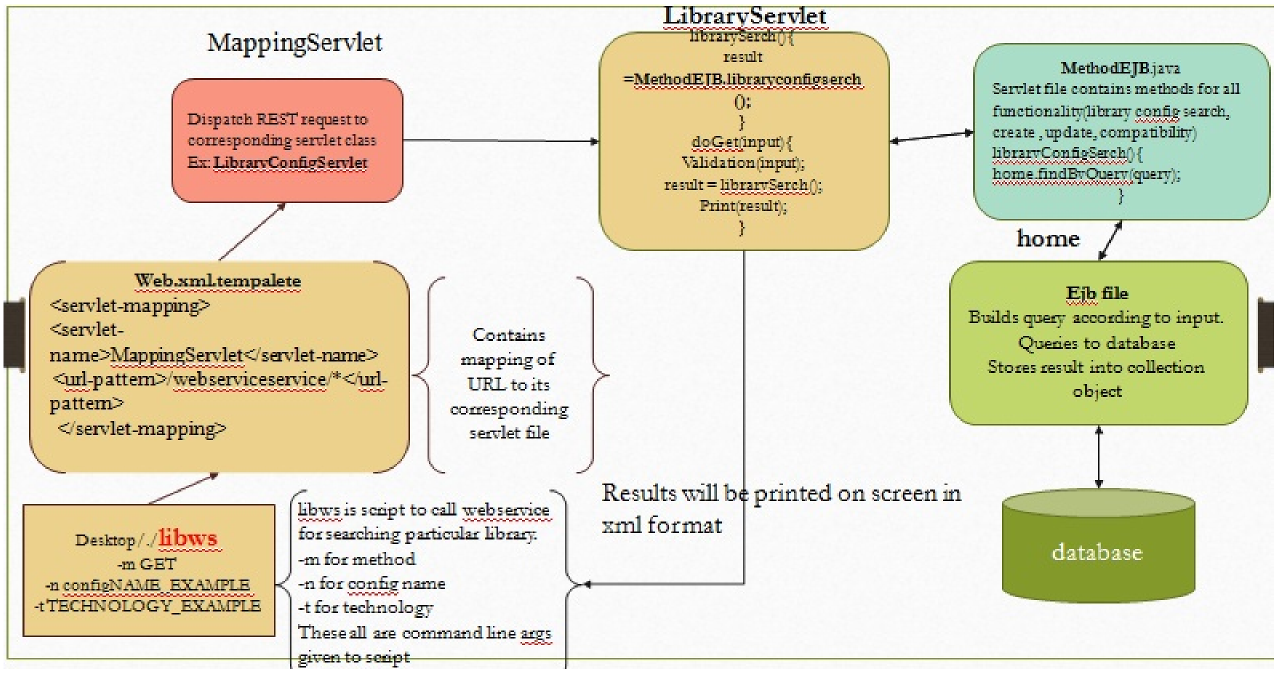
```
<servlet-mapping>
        <servlet-name>MappingServlet</servlet-name>
        <url-pattern>/webservice/*</url-pattern>
</servlet-mapping>
```

- Configure file which contains URL resource name to class name. Here Dispatch servlet is the class contains mapping of resource name to its appropriate class name for all the web service functionality present in the Webgen.

- Third thing is to make URL of web service. Now we have resource name and entry into xml file we can proceed with this step by making URL containing address of our web service. The url pattern could be as follows:

http://example.com/WebgenApplication/webservice/LibrarySearch?name="*",Design_flow
="*"

This URL is generalized we can use this URL directly but better way is to make one file which contains all necessary checks. This file can be a shell script or tickle script. We can call this script for web service functionality by just providing inputs. This script would check whether the necessary inputs are present or not and format of input fields as well as input file is proper or not if it is not the it will put default value or it will give error before executing this web service.



MappingServlet
Dispatch REST request to corresponding servlet class Ex: LibraryConfigServlet
LibraryServlet
librarySerch(){ result =MethodEJB.libraryconfigserch (); } doGet(input){ Validation(input); result = librarySerch(); Print(result); }
MethodEJB.java
Servlet file contains methods for all functionality(library config search, create , update, compatibility) libraryConfigSerch(){ home.findByQuery(query); }
home
Ejb file
Builds query according to input. Queries to database Stores result into collection object
Web.xml.tempalete
<servlet-mapping> <servlet-name>MappingServlet</servlet-name> <url-pattern>/webserviceservice/*</url-pattern> </servlet-mapping>
Contains mapping of URL to its corresponding servlet file
Results will be printed on screen in xml format
database
Desktop/./libws -m GET -n configNAME_EXAMPLE -t TECHNOLOGY_EXAMPLE
libws is script to call webservice for searching particular library. -m for method -n for config name -t for technology These all are command line args given to script

Figure 4.5: architecture of Libraryconfig search example

Error rate reduction in automation process using this functionality:

Before this search functionality the error rate was 65 percent using automation process now after implementation the error rate possibility is 20 percent. Because we are getting installation which comes with new input or modified scenario in 60 percent of cases, new functionality has reduced error rate because it is implemented with every possible scenario. This is probable rate because this functionality is tested on development server. Current working is configuring Quality Assurance platform for testing this functionality. After making all tests this functionality would be directly used by customer so that the error rate can be calculated on bases of live platform.

Chapter 5

Testing Automation Functionality

The current automation process is required to be tested for variety of set of inputs, because it is failing to give desired output for some set of input. For this purpose regression testing is used because the purpose is to test system for some input, modify it and test it again.

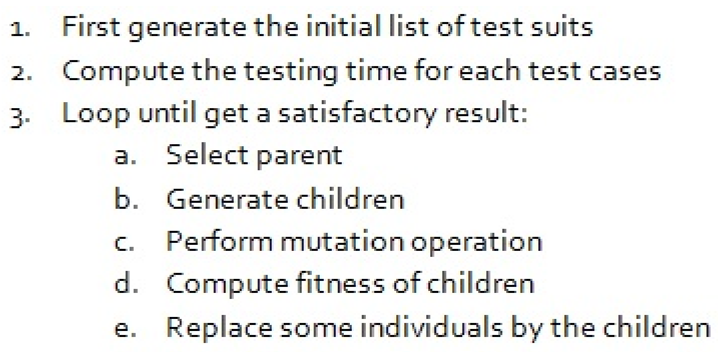
5.1 REGRESSION TESTING

5.1.1 Techniques for Improved Regression Testing

[7]

Regression testing is one of the most widely used testing used for project testing; many of the researchers are focused on the efficiency of the regression testing to make it more efficient. This research elaborates the technique and useful tools used in the regression testing to make regression testing more comfortable and efficient for the user to in the testing phase.[7]

There are four main areas in which the efficiency of the testing can be increased. First technique is to how we can reduce the regression testing time. This can be done by testing those parts which are modified. So we can reduce the number of attempts and scope which will relatively reduce the time to test. Second is to identify the test cases that used to rerun on the subsequent versions of the software and compute the order in which they are effective. Third we have to recycle the test cases by monitoring the executions to gather test inputs those of which can be used for retesting and creating unit test cases. Last technique that could recover test cases from the obsolete test cases is by generating new test cases from the old ones, and by repairing some test cases in



1. First generate the initial list of test suits
2. Compute the testing time for each test cases
3. Loop until get a satisfactory result:
    a. Select parent
    b. Generate children
    c. Perform mutation operation
    d. Compute fitness of children
    e. Replace some individuals by the children

Figure 5.1: algorithm

terms of modification by making it compatible to the modified module.[7]

To increasing efficiency of my testing method for validating Automation process I have implemented some of these techniques.[7]

1) Testing modified part: I have modified automation part library config create and incorporated functionality of library config search. If I test only this file then testing time would be considerably reduced. I need 30 or 40 second for testing the whole automation process rather I would require 1 or 2 second to test only library config search.

2) Identify test case for subsequent version: for creating config I was giving full input file instead here for validating search functionality only need name, version and UPT configuration name and maturity

3) Repairing or recovering test cases is of library config search by identifying necessary input step wise. I would require name for config search then version for checking duplicate config or not.

5.1.2 Time-Aware Regression Testing

[8]

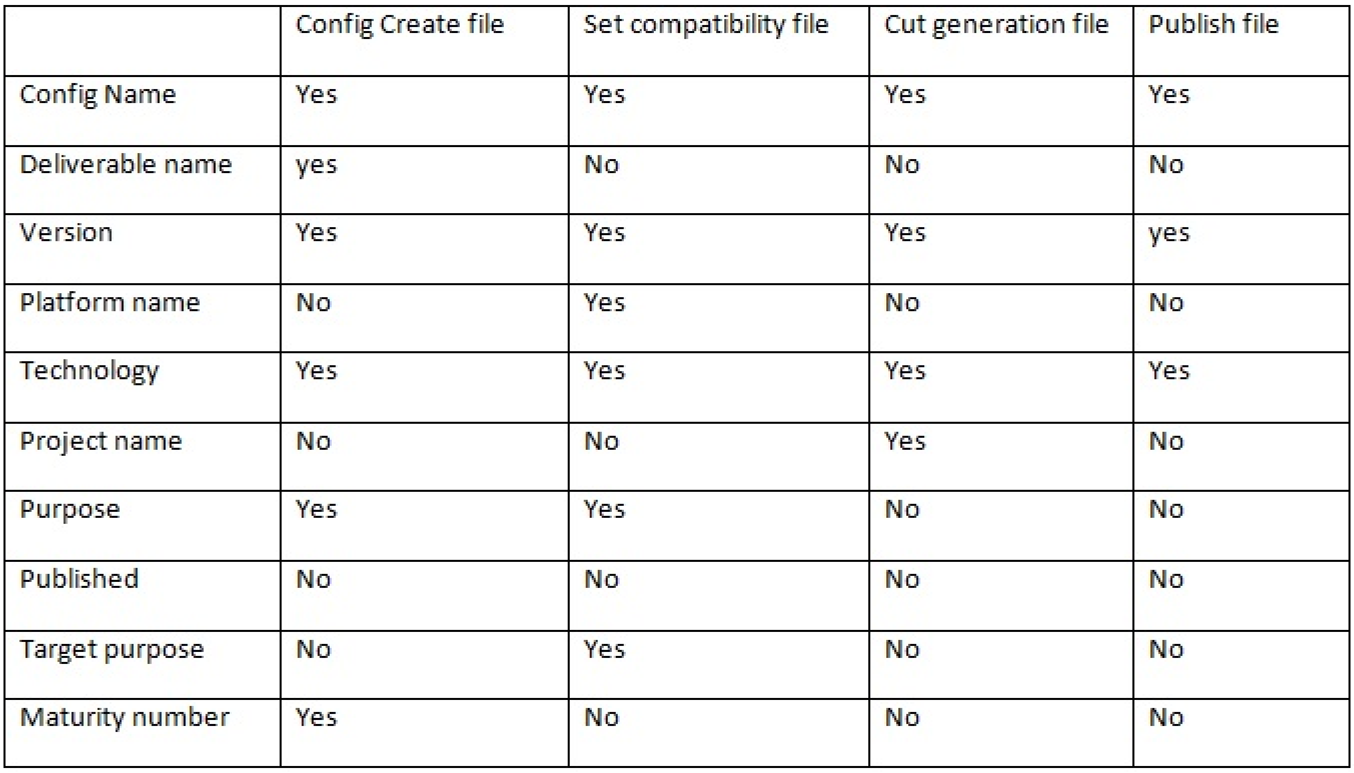
As regression testing can solve the problem of redundant test cases for particular suit. They may be redundant overall by considering the different test suits.[8]

The time aware regression testing algorithm has a better approach than heuristic algorithmic approach.It can be use genetic algorithm for time aware regression testing.[8]

The genetic algorithm[8]

Step 1: here we first generate the initial list in the memory set so we can eliminate the redundant test cases from different test suits.[8]

Step 2: in this step we will compute the testing time for the different test cases.[8]



	Config Create file	Set compatibility file	Cut generation file	Publish file
Config Name	Yes	Yes	Yes	Yes
Deliverable name	yes	No	No	No
Version	Yes	Yes	Yes	yes
Platform name	No	Yes	No	No
Technology	Yes	Yes	Yes	Yes
Project name	No	No	Yes	No
Purpose	Yes	Yes	No	No
Published	No	No	No	No
Target purpose	No	Yes	No	No
Maturity number	Yes	No	No	No

Figure 5.2: Table of test input files(column) and its fields(row)

Step 3: this step is a loop includes multiple steps.[8]

- First we will select the parents from individual two test suits[8]
- Than the cross operation is performed in which redundant test cases are eliminated[8]
- Mutation step includes the addition of bits in the test case which give information about the test cases whether they are used earlier.[8]
- Generating children will be performed by evaluating the computation time of the test cases.[8]
- Replacing parent with the individual children will make new test case to evaluate.[8]

This step 3 will be evaluated until we get the satisfactory result in term of less redundancy of the test cases.

Testing automation process using Time-Aware technique:

1) Initial list of test cases:

Input file of library config create, compatibility, cut generation and publish.

By observing this test cases it is clear that many of the test cases are repeated as well as duplicate so we can eliminate duplicate from input xml file because we can consider this input file as one parent test case and make the reduce set of test case which includes its children only

Test case 1: config name, version, technology

Test case 2: config name, technology, purpose

Test case 3: config name, deliverable name, purpose

Test case 4: config name, platform name, version, purpose

Test case 5: config name, deliverable name, technology

Test case 6: config name, version, maturity number, technology

Test case 7: config name, project name, technology

Test case 8: config name, purpose, design flow, version

Test case 9: config name, version, technology, deliverable name

These are example test cases generated from algorithm. Now form previous methods we can further reduce testing time by reducing the scenario which is required to be tested. If only modified part is only considered then test case according to that part to test them can be further reduced. If concentrate only library config search module then It is required to take some test cases only. The possible candidate for these test suits are as follows.

Test case 1: config name, version, technology

Test case 2: config name, technology, purpose

Test case 3: config name, deliverable name, purpose

Test case 4: config name, platform name, version, purpose

Test case 5: config name, deliverable name, technology

Test case 6: config name, version, maturity number, technology

We can deduce the exact test case for validating this task as follows:

Test case: config name, version, technology, deliverable name, purpose, maturity number This test case is validating this functionality perfectly. This covers every scenario needed to validate the search functionality.

Chapter 6

Input generation Automation and optimization for webservice

The current automation process is required to be tested for variety of set of inputs, because it is failing to give desired output for some set of input. For this purpose regression testing is used because the purpose is to test system for some input, modify it and test it again.

6.1 Overview

Memory installation process aim for generating library with some cuts and conditions, these are the inputs given while installing them. These inputs are available from central repository system. The central repository system is collection of products, deliverable and primaries. To get required input from all these items it has to be fetch and collected.

6.2 Need for Automation

The central repository system is providing input for memory installation system. This system is no longer available and required inputs are also changed. This central repository system has been replaced and there is no automation existing to fetch input from this new system because as the system has been changed the input also no longer same as old one and the way of fetching input is also changed. There is need to develop some automated system which will fetch input from this system and help to build input file in required format because all the webservice will run properly only if enough numbers of inputs are provided with proper format.

6.3 Proposed solution

There are multiple solutions proposed.

1) Fetch input one by one by querying and check for formatting and generating output file.

2) Collect all the inputs at one place and check for proper formatting while generating output files.

There is huge discussion which approach to be taken into consideration finally based on following criteria

-speed

-performance

-time needed and effort required for the development

-repetition of task

These are important factors for quality of automation to be delivered to the customers. One by one each criterion has been observed and final product schema has derived.

1) Speed: Time to execute each of module and collectively of whole system. The task should take minimum time and gives output as fast as possible.

2) Performance: Automation performance is dependent on many criteria which are line of code, error handling and how each conditions has been evaluated so that it would take less time.

3) Time and effort required for development: both approached has been tasted for time and effort required and graphs have been plotted.

In effort vs query graph the time and efforts are considered on the basis of queries, however instead of queries there are many factors that affects a lot for example functions we defined and code written. To complete the process in less time we need to give more efforts as depicted making one complex query the effort needed is very high as compared to effort needed to write 5 simple queries which does same work as one query is very high. According to graph analysis we need more time and lots of effort to develop the complex queries but the advantage is that we get our work done in less time.

4) Repetition of task The first approach will query inputs one by one and the find out answers as well as syntax checking is also performed one this step, again for second input the same task would be repeated. This affects performance and response time. So

the first approach would be slow as compared to second one because the approach would less time and effort for development but it would have repeated task.

This four criteria has been observed and from that the conclusion can be derived that first approach will require less development time less efforts while the time and speed factors are not good for this approach as well as it contains repeated tasks so performance will be degraded. The second approach is very hard to implement but it has 3 factors in favor which are time speed and performance.

The second approach is now better to implement, before going towards this implementation part several terms needs to be explored.

Defining Central repository system

The system stores all the kind of products, deliverable and primaries and plug-in and gives the information needed for different purpose like library generation, product creation. This central repository system has been changed and we now the newer central repository system which is enhanced one but problem with this is its no longer providing the input in older format or some inputs has been changed, some inputs path or the package which contains it has been changed.

Chapter 7

Automation and Implementation

7.1 Introduction

There are two parts of automation which is automation of installation process and second one is automation of input generation which will be given to the installation process. The one part has been completed we eliminated the manual process on portal by replacing it with webservice which is performing same task as required for installation. The overall idea is as depicted in Automation Idea figure 7.1.

Need of Automation

There are still manual efforts needed to have installation of library and cut generation. The webservice is providing the automation of web portal task but inputs are still required to be filled manually into xml files. This manual process needs to be automated to make whole system automation to have good performance.

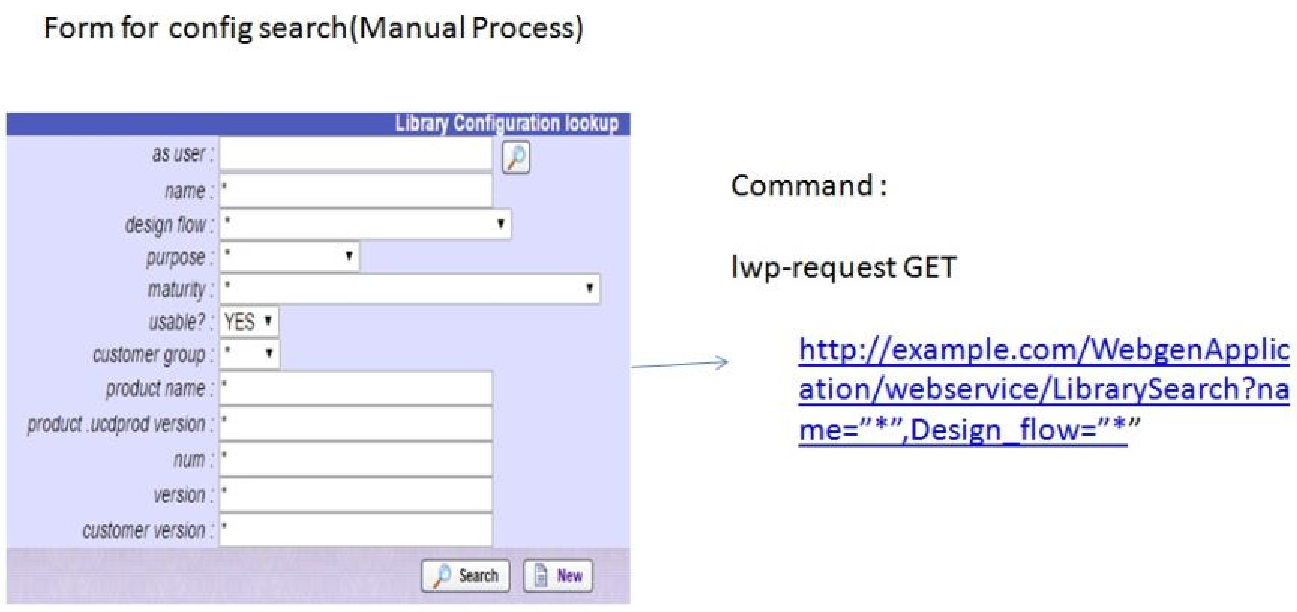
7.2 Automation Idea

Proposed automation flow is depicted into figure 7.2. This is abstract view of automation of input generation.

The flow has been explored step by step that described in following points.

1) The second approach proposed earlier is depicted above. First we query inputs or defines inputs form the path or some other packages and collects in one common file. This file is containing the final input set.

2) Create xml tags lists. This tag list will contain the tags for input required for the automation process. This tag list defines structure of the xml file.


Form for config search(Manual Process)
Library Configuration lookup
Search
New

as user :	
name :	*
design flow :	*
purpose :	*
maturity :	*
usable? :	YES
customer group :	*
product name :	*
product .ucdprod version :	*
num :	*
version :	*
customer version :	*

Command :

```
lwp-request GET
http://example.com/WebgenApplication/webservice/LibrarySearch?name="*",Design_flow="*"
```

Figure 7.1: Automation Idea

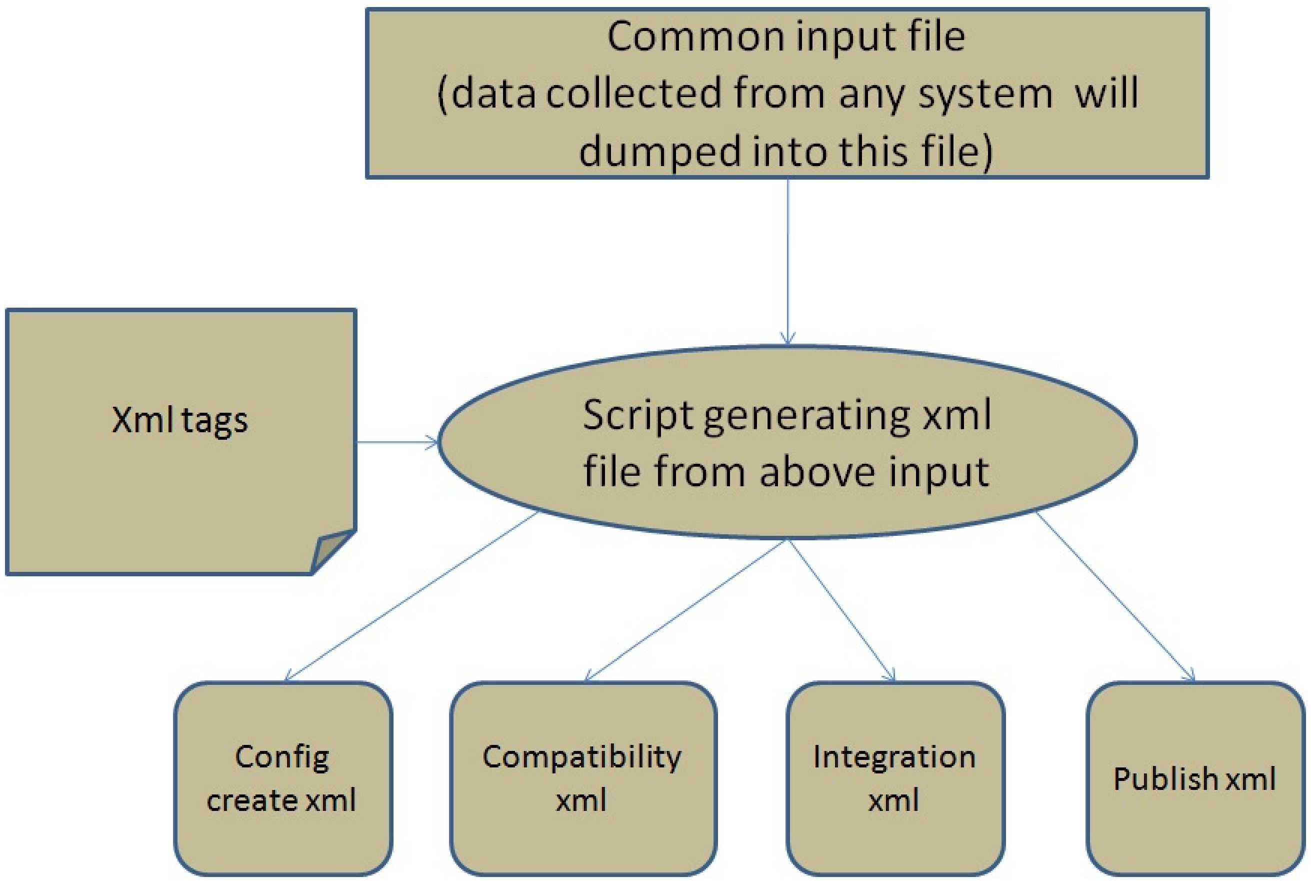

Common input file
(data collected from any system will dumped into this file)
Xml tags
Script generating xml file from above input
Config create xml
Compatibility xml
Integration xml
Publish xml

Figure 7.2: Automation Idea

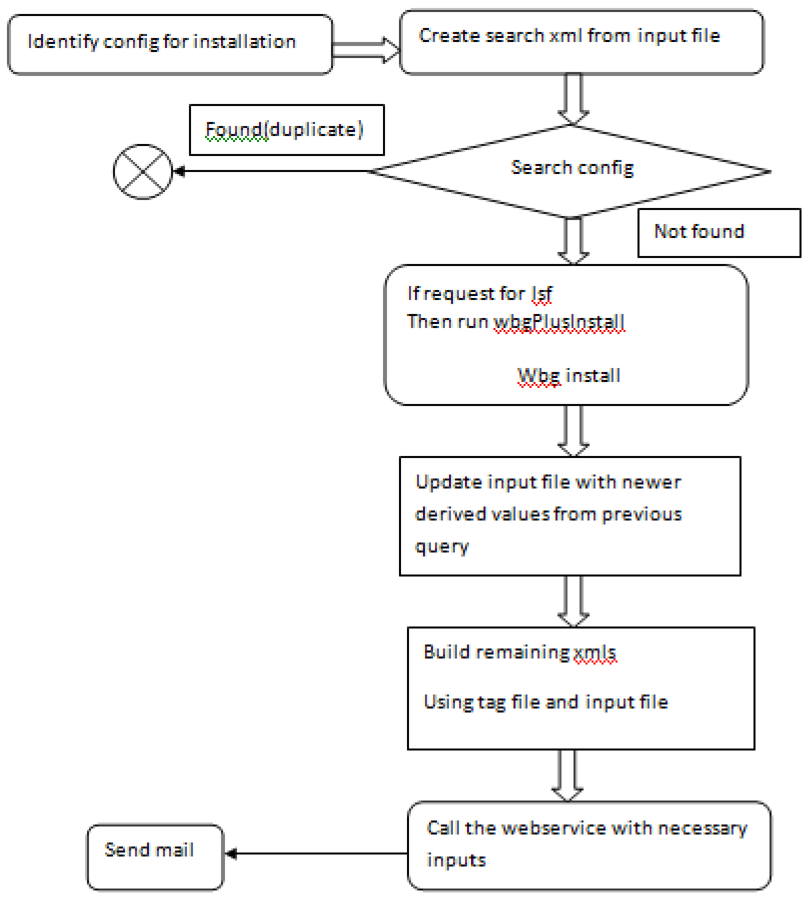

Identify config for installation
Create search xml from input file
Found(duplicate)
Search config
Not found
If request for lsf
Then run wbgPlusInstall
Wbg install
Update input file with newer derived values from previous query
Build remaining xmls
Using tag file and input file
Send mail
Call the webservice with necessary inputs

Figure 7.3: Flowchart or algorithm of implemented system

3) Script: This is main part of automation process this script will handle all the things. Collecting and modifying inputs, Setting up a global variable and functions, Calling scripts for xml generation, Error handling, Call webservice for installation process and gives us an output. This script is the main controller of whole automation process.

4) Last step of whole picture is generation of xml files from tag list as well as input set. The generation process is also takes the tag list and finally come out with xmls

7.3 Implementation Flow chart

Main script is collection of small task as well as its controller of this automation. So the proposed flow has been depicted into the 7.3 figure. The steps included here are processed by the script only, and to execute these all process many supporting scripts has been developed. Different kind of task is handled by scripts individually and this main script handles the tasks how to call the scripts,when to call the scripts,input for this scripts,output from this scripts,finally error handling.

In this way this script will act as a controller for the whole automation process. This flow is modular so that we can easily move the steps inside script whenever we want in this way the dependency between the tasks has been removed.

1) The script which is calling the main script will set the environmental variable and paths as well as it will set the logging of the process by creating the file

2) Main script will first source the files which contains global variable and functions needed throughout the process.

3) Querying inputs from repository. Convert this input in necessary format if needed then check for syntax and value

3) The request of config installation can be duplicated so first it is necessary to check the duplicity of the product so calling of webservice search config is necessity here. The problem here is this file also needs some input in xml format. So first generate input set from common input file for the searching the config and the after the process of search will be carried out by giving xml input to them.

4)Output from the above step gives idea whether it can proceed further or the config is duplicate so stop the process with message describing config name, technology and whether it can be installed or not.

5) The further process carried out if config is new. So now the very important task which is product declaration on to webgen needs to be started this also can be done by single script which will be called with proper inputs.

6)if product installation goes without any error the next task is to generate input xmls for rest of the webservice. This will call the convert xml script and convert xml would source input file and tag file then generates output xmls for each of the installation process

7) final step is to call web service which would take the input xmls and generates the library. It also gives final answer whether its successful or error if any.

7.4 Installation process analysis and input ideas (find out necessary inputs)

The central system has changed so we no longer get the proper inputs as same as the older one. So to build input from newer system we need to analyze the inputs and system which was fetching inputs from older version of central repository. The analysis has to be carry out to identify and differentiate the inputs and categorize it as: 1)Manual 2)constant 3)derived 3)from central system

The intense analysis has been carried out and finally the conclusion was made that Our input set is divided into this four categories. There is need to analyze each of the input where it is coming from and where it is used , these all criteria has been taken into account to analyze inputs and after that each input is categorized. Some inputs were unnecessary some were repeated and so these inputs have been eliminated and final unique input set has been build which will be queried from the central repository.

7.5 Xmls analysis and addition/deletion

The central system was queried by the old process followed for fetching inputs they were also generating inputs in the form of xml files. However there are four different processes followed one after another for installation, their inputs were overlapping an. Even though they are over lapping they were repeatedly queried and they were used in different xml tags. This system will be replaced by the input generation automation in which will derive the inputs or just make the tag name same as in other files to have unique value, the other way is do not change the value of tag just fetch the value one time collect all input as unique key value pair and put this value inside tag.

7.5.1 analysis and modification

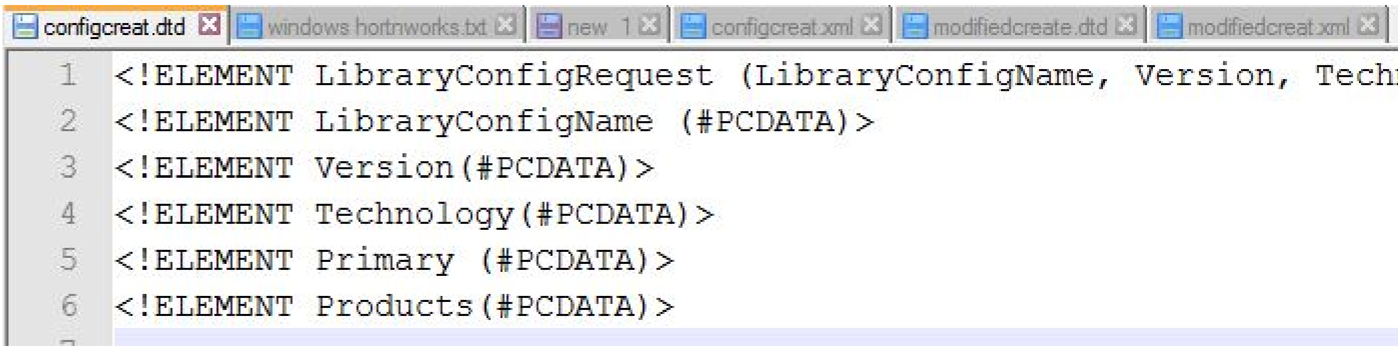
[1]

Introduction to xml

The xml can be used in similar fashion as HTML(Hypertext Markup Language). Extensible Markup Language is considered as sub part f Standardized General Markup Language(SGML) and used in web. Xml Technology is made up by combining two things 1) DTD: document type Definition file, 2) The actual data file in form of xml.[9]

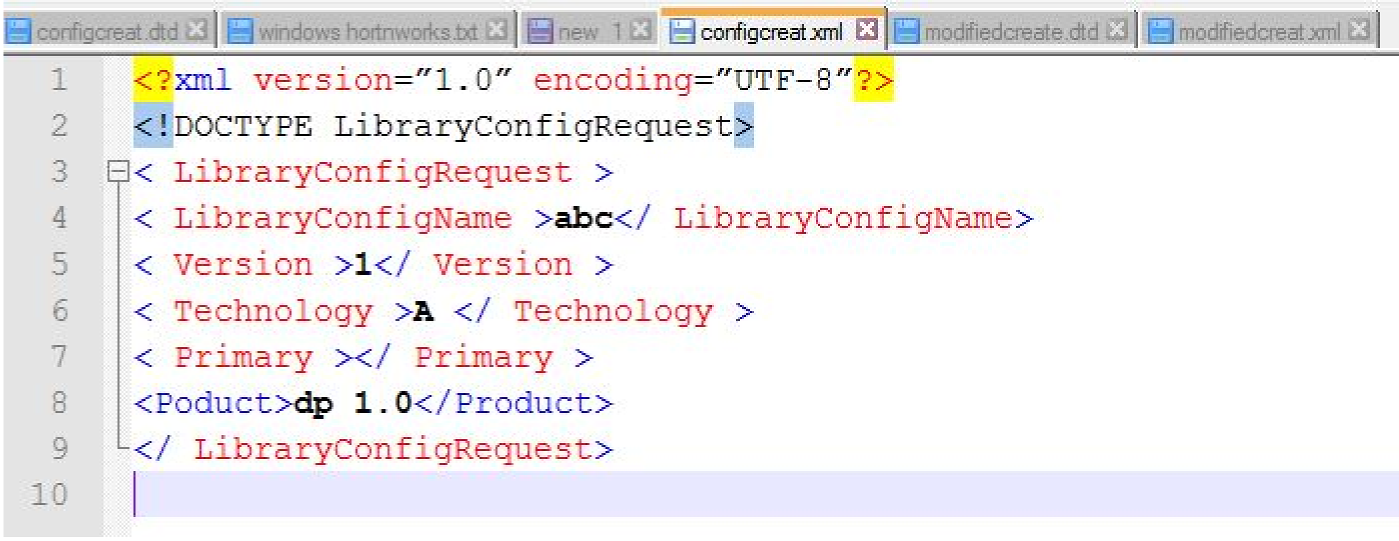
DTD Definition or in simple terms type of particular element residing inside the xml data file will be described by DTD file. The structure or hierarchy can be described with the help of xml DTD.[9] Following is the example dtd file:

The !ELEMENT is the keyword which describes top level tag helps in describing hierarchy. The (#PCDATA) element is defining the type of element. The element resided within parenthesis is to be taken as in hierarchy of top level element for example (Li-



```
<!ELEMENT LibraryConfigRequest (LibraryConfigName, Version, Tech
<!ELEMENT LibraryConfigName (#PCDATA)>
<!ELEMENT Version(#PCDATA)>
<!ELEMENT Technology(#PCDATA)>
<!ELEMENT Primary (#PCDATA)>
<!ELEMENT Products(#PCDATA)>
```

Figure 7.4: DTD file of ConfigCreate



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LibraryConfigRequest>
< LibraryConfigRequest >
< LibraryConfigName >abc</ LibraryConfigName>
< Version >1</ Version >
< Technology >A </ Technology >
< Primary ></ Primary >
<Poduct>dp 1.0</Product>
</ LibraryConfigRequest>
```

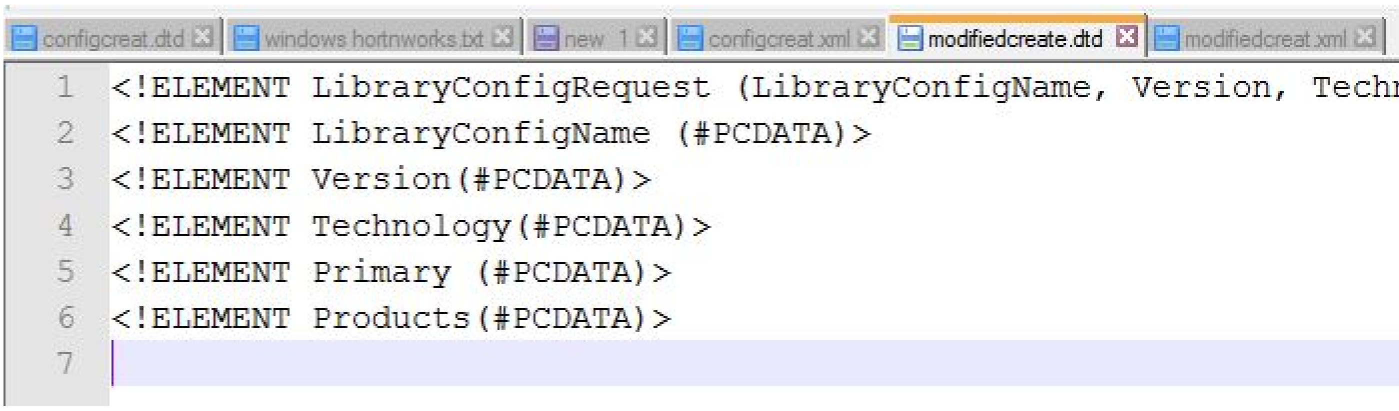
Figure 7.5: xml file of ConfigCreate

braryConfigName, Version, Technology, Primary, Products) is child element of top level element LibraryConfigRequest. There are many more options available to de define data element for example(?) after the element defines it as an optional element.[9]

7.5.2 Modifications needed for newer webservice

AIM The modification requirement is raised because the automation is not generating all the inputs before searching. Time taken by product installation on webgen and generation of products file as well as their list is very high comparatively other tasks. The main goal of automation is to reduce the time taken by the whole manual process. The first step is to search library config from the webgen and if it is duplicate then no need to go ahead and install products it reduces considerable amount of time taken by the webservice, so there is no need to include product detail inside the input set of library config search.[9]

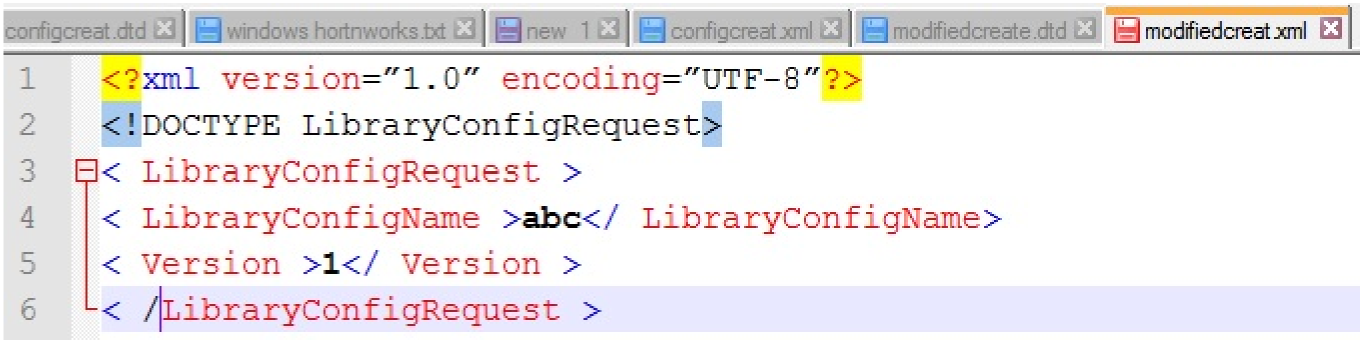
This is the example xml file for the library config request xml file, here if I want to apply this xml as an input to the webservice search library config then It should have



configcreat.dtd | windows hortnworks.txt | new 1 | configcreat.xml | modifiedcreate.dtd | modifiedcreat.xml

```
<!ELEMENT LibraryConfigRequest (LibraryConfigName, Version, Techn
<!ELEMENT LibraryConfigName (#PCDATA)>
<!ELEMENT Version(#PCDATA)>
<!ELEMENT Technology(#PCDATA)>
<!ELEMENT Primary (#PCDATA)>
<!ELEMENT Products(#PCDATA)>

```

Figure 7.6: modified DTD file of ConfigCreate



configcreat.dtd | windows hortnworks.txt | new 1 | configcreat.xml | modifiedcreate.dtd | modifiedcreat.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LibraryConfigRequest>
< LibraryConfigRequest >
< LibraryConfigName >abc</ LibraryConfigName>
< Version >1</ Version >
< /LibraryConfigRequest >
```

Figure 7.7: modified xml file of ConfigCreate

reduced set of xml elements which are really needed for the searching and other ones which are to be made as an optional.[9]

Primary is already optional and The element needed for the searching are library config name and version other should be made as an optional elements so newer dtd and input set in the form of xml is as following:[9]

1)library.dtd

2)library.xml

This is very abstract view of the input xml file the original one is large with plenty of inputs which has to be analyzed. This analysis will lead addition or deletion of the elements from input set.[9]

7.5.3 Query building and planning

The input will be either queried from the repository or will be derived from other inputs so we need to fetch the inputs from repository in optimal way.

First the inputs will be provided by from newer system so that this is necessary to understand newer systems architecture and database system, other than this it is also necessary to understand in what way we can derive inputs.

Some query building practices has been carried out on our webgen database they are as follows: There are two types of approach have been proposed to derive inputs

1) we install all the products on webgen and then after we fetch the inputs from repository

2) we can have the inputs at time of product installation because it also making request for same inputs and we can derive our inputs from them.

Query planning is still open point of discussion so even though we can make the further system ready.

7.5.4 Input validation

The planning of from where to derive input has been decided then validating inputs are the most important part of automated installation because in manual system we the portal will give error or manually we can change the input in required format. Here In automation it is necessary to teach the system what is proper format for inputs.

7.6 implementation with screen shots

The system implementation is made up with the help of following files

Configfile

Configfunctionfile

InputTagforXml

Installation

productToInstall

convertToXml

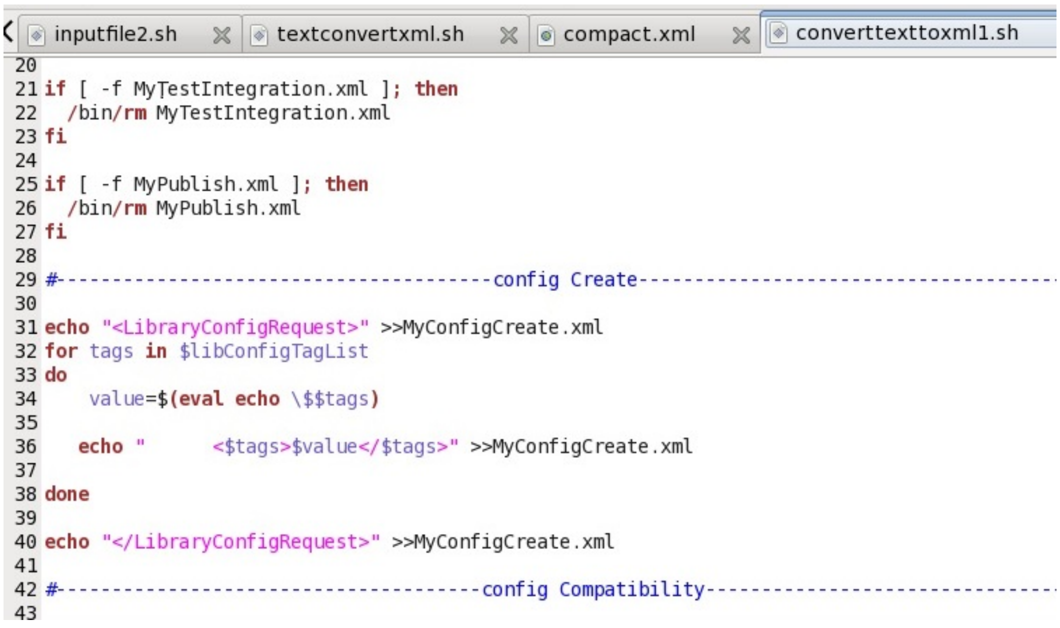
inputFile

ParentFile

wsAll

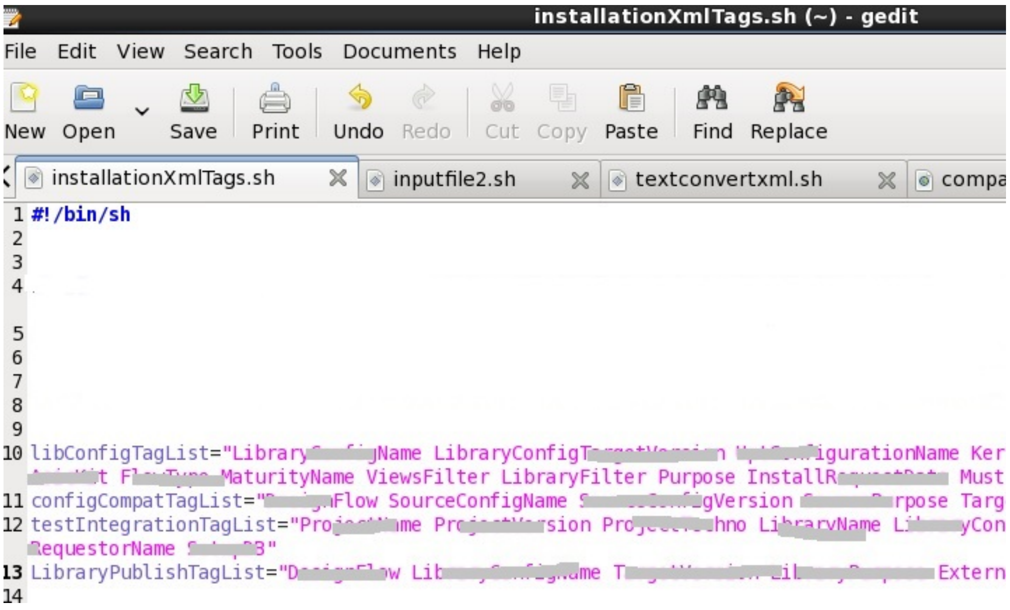
This all files interact with each other; take inputs and gives output to the main file which is installation. This main file is script which would be called by parentfile script

ParentFile: It will set up environment and global variables. It also set up the log file generation so that whatever the logs generated by running the following scripts would be stored here and we can refer to it whenever any error comes. This parent file is the base



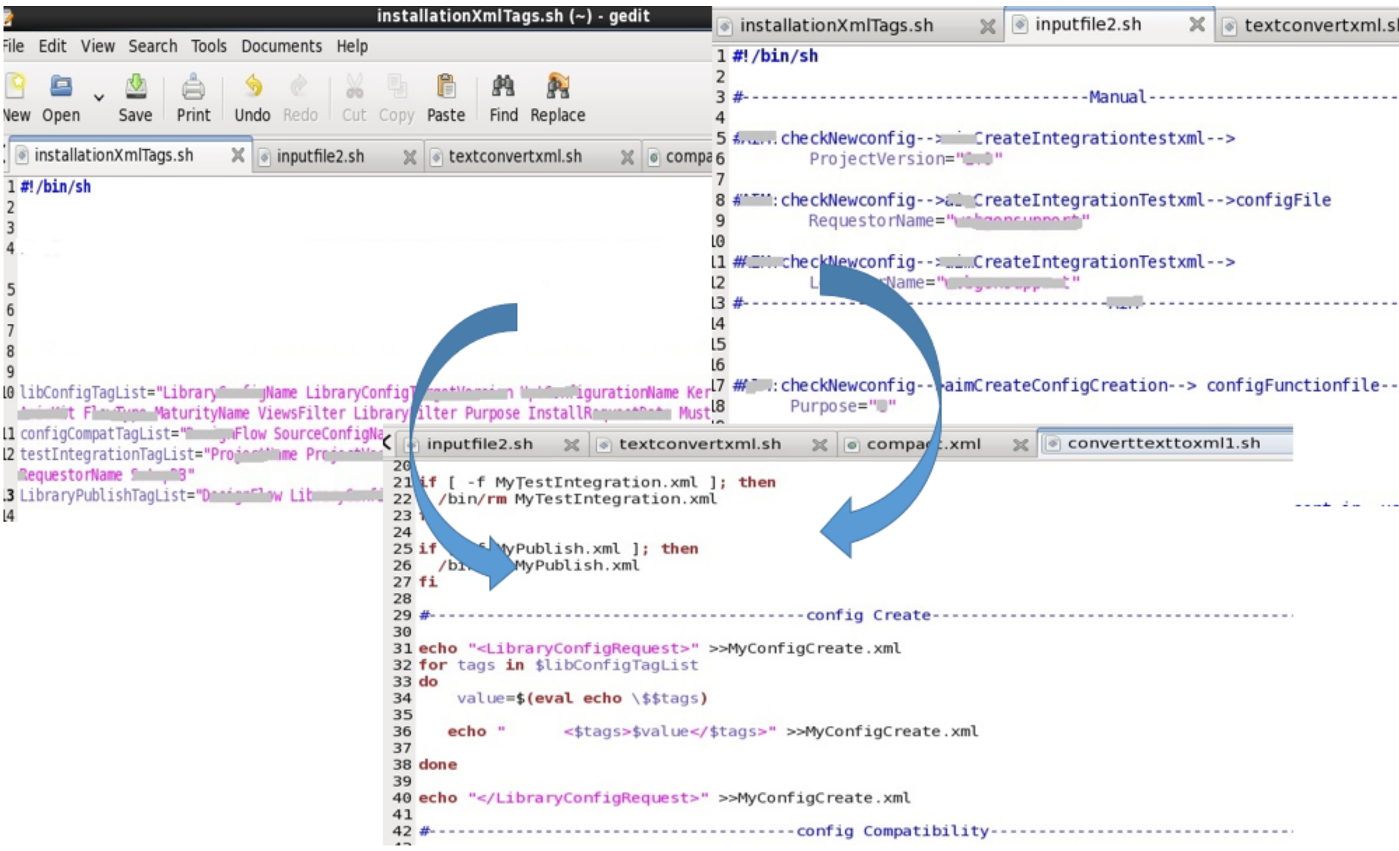
```
if [ -f MyTestIntegration.xml ]; then
  /bin/rm MyTestIntegration.xml
fi

if [ -f MyPublish.xml ]; then
  /bin/rm MyPublish.xml
fi

#--------------------------------------config Create-----------------------------------

echo "<LibraryConfigRequest>" >>MyConfigCreate.xml
for tags in $libConfigTagList
do
    value=$(eval echo \$$tags)

   echo "      <$tags>$value</$tags>" >>MyConfigCreate.xml

done

echo "</LibraryConfigRequest>" >>MyConfigCreate.xml

#-------------------------------------config Compatibility-------------------------------
```

Figure 7.8: converts the inputs into xml


installationXmlTags.sh (~) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
installationXmlTags.sh
inputfile2.sh
textconvertxml.sh
compa

```
#!/bin/sh









libConfigTagList="Library[redacted]Name LibraryConfigT[redacted]n [redacted]igurationName Ker
[redacted]t F[redacted]MaturityName ViewsFilter LibraryFilter Purpose InstallR[redacted] Must
configCompatTagList="[redacted]Flow SourceConfigName [redacted]gVersion [redacted]rpose Targ
testIntegrationTagList="Pro[redacted]me Pr[redacted]sion Pro[redacted]hno LibraryName Li[redacted]yCon
[redacted]equestorName [redacted]3"
LibraryPublishTagList="D[redacted]w Lib[redacted]me T[redacted]il[redacted] Extern

```

Figure 7.9: converts the inputs into xml


installationXmlTags.sh (~) - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
installationXmlTags.sh
inputfile2.sh
textconvertxml.sh
compa
compact.xml
converttexttoxml1.sh

```
#!/bin/sh




libConfigTagList="Library[redacted]Name LibraryConfig[redacted]n [redacted]igurationName Ker
[redacted]t F[redacted]MaturityName ViewsFilter Library filter Purpose InstallR[redacted] Must
configCompatTagList="[redacted]Flow SourceConfigNa
testIntegrationTagList="Pro[redacted]me Pr[redacted]
RequestorName [redacted]3"
LibraryPublishTagList="D[redacted]w Lib[redacted]
```

```
#!/bin/sh

#--------------------------------Manual-----------------------

#[redacted]checkNewconfig-->[redacted]CreateIntegrationtestxml-->
        ProjectVersion="[redacted]"

#[redacted]:checkNewconfig-->[redacted]CreateIntegrationTestxml-->configFile
        RequestorName="[redacted]"

#[redacted]checkNewconfig-->[redacted]CreateIntegrationTestxml-->
        L    rName="[redacted]"
#---------------------------------[redacted]-----------------------



#[redacted]:checkNewconfig-- aimCreateConfigCreation--> configFunctionfile--
        Purpose="[redacted]"
```

```

if [ -f MyTestIntegration.xml ]; then
  /bin/rm MyTestIntegration.xml


if     MyPublish.xml ]; then
  /bi     MyPublish.xml
fi

#-----------------------------------config Create-----------------------------------

echo "<LibraryConfigRequest>" >>MyConfigCreate.xml
for tags in $libConfigTagList
do
    value=$(eval echo \$$tags)

    echo "       <$tags>$value</$tags>" >>MyConfigCreate.xml

done

echo "</LibraryConfigRequest>" >>MyConfigCreate.xml

#----------------------------------config Compatibility----------------------------
```

Figure 7.10: generation of xml

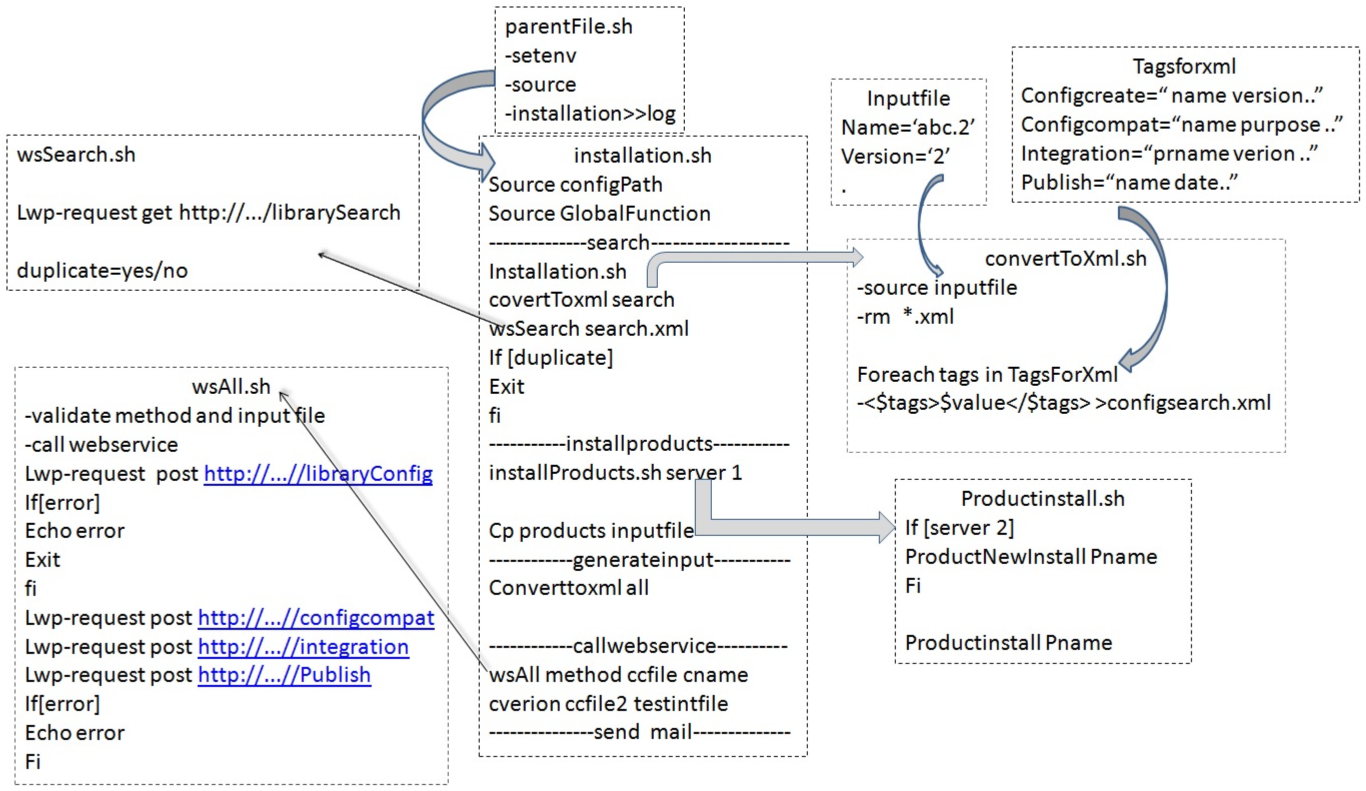

parentFile.sh
-setenv
-source
-installation>>log
wsSearch.sh
Lwp-request get http://.../librarySearch
duplicate=yes/no
installation.sh
Source configPath
Source GlobalFunction
-------------search-------------------
Installation.sh
covertToxml search
wsSearch search.xml
If [duplicate]
Exit
fi
-----------installproducts----------
installProducts.sh server 1
Cp products inputfile
------------generateinput-----------
Converttoxml all
------------callwebservice---------
wsAll method ccfile cname
cverion ccfile2 testintfile
--------------send mail-------------
Inputfile
Name='abc.2'
Version='2'
.
Tagsforxml
Configcreate="name version.."
Configcompat="name purpose.."
Integration="prname verion.."
Publish="name date.."
convertToXml.sh
-source inputfile
-rm *.xml
Foreach tags in TagsForXml
-<$tags>$value</$tags> >configsearch.xml
wsAll.sh
-validate method and input file
-call webservice
Lwp-request post http://...//libraryConfig
If[error]
Echo error
Exit
fi
Lwp-request post http://...//configcompat
Lwp-request post http://...//integration
Lwp-request post http://...//Publish
If[error]
Echo error
Fi
Productinstall.sh
If [server 2]
ProductNewInstall Pname
Fi
Productinstall Pname

Figure 7.11: Automaton System Implementation whole picture

file of main script. The main script will be initiated by the parentfile.

Installation: This script is a main controller of whole the automation which handles when to call which script and what input should be given to the particular script , what are the output of the particular script.

ProductToInstall: It handles two type of product installation and which type of products should be installed will be decided based on which server it is running. The main script would give instruction to this script for deciding about installation.

ConverToXml : It would take input file and xml tags, with the help of this both and some conditions it will generate output xmls.

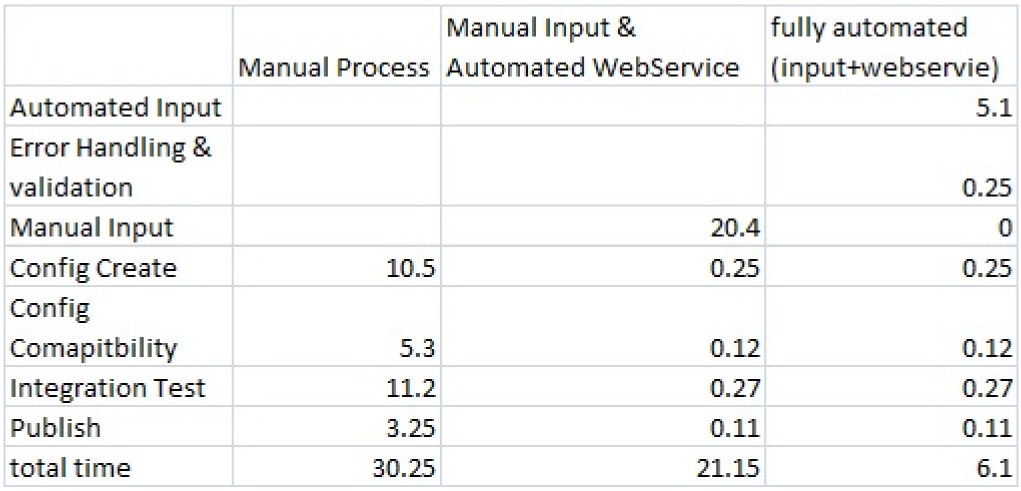
Inputfile: The input which will be collected in one place and this file doesnt contain all inputs. The information of products would be given to it when all products have been declared on to the webgen.

InpitTagForXml:Tags specification for particular input xml file specified here.

wsAll : It is huge script which binds the calling of all the webservice at one place, it handles webservice would take which inputs and at last the handling of error, logging accomplished by this script.

7.7 Results

There are two development stages included in the process of whole system automation task. First Was webservice development and second was automated input generation. The output product which is automation of the installation gives great advantage in terms of time, effort and performance.



	Manual Process	Manual Input & Automated WebService	fully automated (input+webservie)
Automated Input			5.1
Error Handling & validation			0.25
Manual Input		20.4	0
Config Create	10.5	0.25	0.25
Config Comapitbility	5.3	0.12	0.12
Integration Test	11.2	0.27	0.27
Publish	3.25	0.11	0.11
total time	30.25	21.15	6.1

Figure 7.12: Execution Time Data

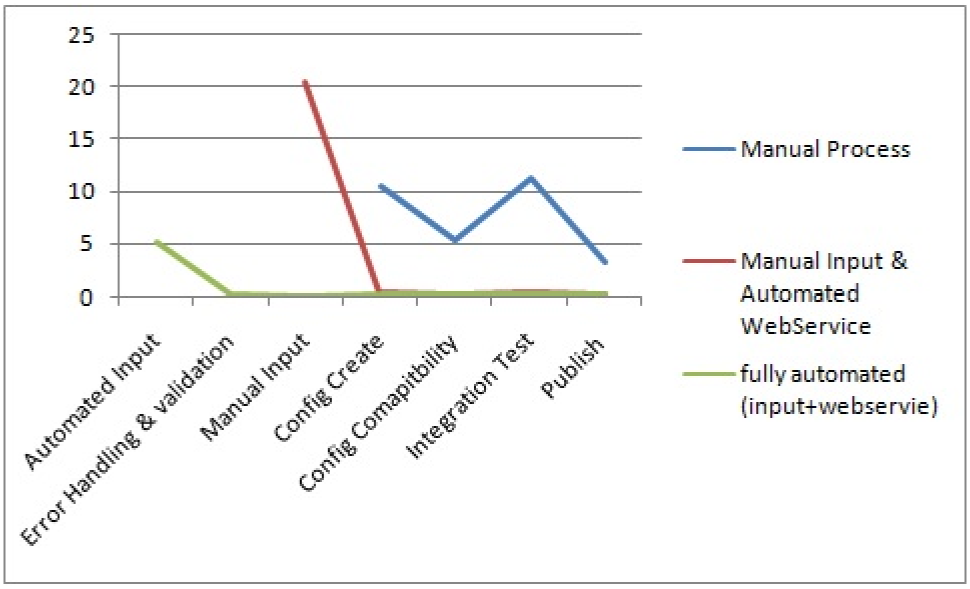


Figure 7.13: Execution Time Graph

The manual process was the tedious task, it was taking about half an hour in any ideal conditions to accomplish the installation process. The first target was to achieve automation of manual process carried out on the portal, so the webservice have been developed to replace on manual part which was time consuming. In second stage there is the process of building the input file which was still manual and has to be automated, so for that one solution has been made to generate input automatically and give xmls as output. The combined system was giving very optimized version which was saving huge amount of time.

The graph showing here three stages from manual to fully automated system here the y-axis representing time taken by each process described in x-axis. The graph shows many more hikes in case of manual process and totally takes time more than other two. Semi- Automated system would take comparatively less time than manual and lesser time compared to all others taken by fully automated system.

7.7.1 Optimization / Speed Up

Gain from the whole system has been calculated.

1)manual input automated webservice speed up is about 30%.

2)whole automation speedup is about 80%.

This considerable gain would helps to reduce effort and time taken by the system.

Chapter 8

Conclusion and Future Work

Conclusion

Webgen automation process is still in progress, as requirement or input has changed now the old automation process is not working as desired for some new set of inputs. For making automation compatible with new and broad set of input some part has been changed. Libraryconfig search functionality is implemented for this purpose as well as some new functionality which is product search which was incorporated in product installation work. This functionality has reduced the error rate by 40 percent and increased the performance. Now for testing this functionality various testing methods which are regression testing and some other enhancement over regression testing has been used for reducing the testing time and making testing more appropriate to test this modified module. Testing time after incorporating has been reduced 20 to 30 percent.

The automation of manual installation process is followed by the automation of input generation so this two parts combining one single automation from generating input, making xml file, calling webservice which performs the installation process. this task can be accomplished with the help of one single command. This automation has saved the effort and time required for the whole installation process. The overall gain in any ideal condition is about 80%.

Future work

Currently working for configuring the Quality assurance platform for further testing after completion of this task the future direction is to handle issues regarding to automation process.

Bibliography

[1] "St webgen manual,"

[2] E. Tutorial Java, "da oracle," 6.

[3] F. Belqasmi, J. Singh, S. Bani Melhem, and R. Glitho, "Soap-based vs. restful web services: A case study for multimedia conferencing," *Internet Computing, IEEE*, vol. 16, pp. 54–63, July 2012.

[4] S. Mumbaikar, P. Padiya, *et al.*, "Web services based on soap and rest principles," *International Journal of Scientific and Research Publications*, vol. 3, no. 5, 2013.

[5] C. Davis, "What if the web were not restful?," in *Proceedings of the Third International Workshop on RESTful Design*, pp. 3–10, ACM, 2012.

[6] P. K. Potti, S. Ahuja, K. Umapathy, and Z. Prodanoff, "Comparing performance of web service interaction styles: Soap vs. rest," in *Proceedings of the Conference on Information Systems Applied Research ISSN*, vol. 2167, p. 1508, 2012.

[7] M. J. Harrold, "Reduce, reuse, recycle, recover: Techniques for improved regression testing," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, IEEE, 2009.

[8] L. You and Y. Lu, "A genetic algorithm for the time-aware regression testing reduction problem," in *Natural Computation (ICNC), 2012 Eighth International Conference on*, pp. 596–599, IEEE, 2012.

[9] K. Nyberg, "Automatically generating dtd-specific xml parsers," *ACM SIGAda Ada Letters*, vol. 30, no. 2, pp. 13–18, 2010.