UNIVERSITY OF
LEICESTER

DEPARTMENT OF INFORMATICS
UNIVERSITY OF LEICESTER

CO7201 Individual Project

A web tool for ChorGram

By

ZHAO LIU

Student Number: 149030494

Email: zl137@student.le.ac.uk

Supervisor: Emilio Tuosto

Second marker: Nervo Verdezoto

Final-report

Words Count: 10106

Submitted: 12/ 05/ 2017

UNIVERSITY OF LEICESTER

Contents

Abstract

The message-passing system is a subsystem of distributed system. A high efficient message-passing system could improve program execution efficiency and ensure data security. The choreographies [3] provide two types views which are global view and local view for presenting the states of system communication. However, the generating choreography graphs processing is complicated. The ChorGram is a tool for producing machines communication diagrams based on choreography. More precisely, the ChorGram through its special instructions to describe the messages exchanging between systems. And ChorGram also can generate global views automatically according to these instructions. Thus, this tool can be used to validate or analyze the interactive behaviors of distributed system for companies and professionals. In distribute system design, it always involves two or more machines to exchange information. The ChorGram could offer predicaments that the states and positions of messages in a time point for each machines by graphs. These graphs could be evidence for distributed system design. This feature could help developer to decrease bugs and ensure the security between machines during the project design stage. Although, the ChorGram is an excellent software, but the operation and configuration of this tool is not simple enough for beginners. Moreover, ChorGram is a desk application that reduces the spread of this tool. Therefore, this paper is highlighted the main features of WebChorGram, it is a web interface for ChorGram. In development of WebChorGram, it used several popular frameworks and tools, such as Django, HTML5 and JQuery. WebChorGram contains not only friendly GUIs for operation, but also powerful functionality for producing and analyzing choreography graphs. In addition, WebChorGram also suitable for students to learn the choreography and ChorGram.

1. Introduction

Web System and application has already becoming popular in recent years. Most desktop software also has released web version. The customers could directly use software without other work. Especially for the tool type of software, web version can greatly simplify the preparatory work such as downloading, installation and configuration environment. In addition, web system is deployed on the server, even if the user does not understand the system, it does not be crashed by wrong operation. For example, Graphviz [17] has a web version Webgraphviz, users simply input the text commands and press generation button, then, the outputs will show on the web pages. Furthermore, the trend of using cloud platform increase continually, the development of web system or application and maintenance has become progressively easy and low cost. However, change a desktop program to web is not easily work. Basically, the functions of desktop applications should be keep in web version. Moreover, developer should consider the characteristics of web system, and add some features to guarantee that the web system interfaces are friendly enough. Hence, to increase web interfaces to desktop software should fully understand the features, usages, and operation methods of the original software.

ChorGram [5] is tool for designing and analyzing distributed systems. Distributed systems and applications have been already used in the majority of industries such as financial services, travel services, and food industry. However, most of these systems or applications are developed not easy. As the system needs to cooperate with each other, it is critical to ensure that information is exchanged between them. Exchanging

message is the fundamental feature in distributed system. To guarantee the communication, the components should accept same format messages. [1] There are two main ways to sharing information between two or more systems. One is sharing data, and another is message-passing. However, in the current Internet, sharing data is not a sensible method. Thus, message-passing or copy sharing is the most popular using in distributed system. And message-passing system is an important part of distributed system. [2] Hence, programing a good message-passing system should think about a lot of factor such as the message is simple enough, the message is uniform, and efficiency. For possible problems, developers should try to solve them when designing.

For different systems, they could have been developed by different languages, platforms or operating systems. To make these systems exchange information normally, not only a common message format, but also to ensure that the system can suitable send and receive the correct messages. The Web Services Choreographic [3] is a way that could help developer to check the system's status and messages before coding the programs. Choreographic is able to describe the association between different kind of system unrelated which languages or scales and platforms. And this approach supplies a concept "global view" to show that each system or component should exchange the corresponding message under the specified conditions. Through global view, each part can use it to generate a standard for projecting system and test. Under this commonly standard, the local system can communicate with others smoothly. The global choreographies can use sequence diagrams to indicate [6]. Sequence diagrams describe the interactions between components with communication patterns and the lifelines. In choreographies, the details of components are the core portions which the graph should

be represented. Hence, a global view could not only analyze deeper level about sharing data but also the cooperation orders between each independent component or system.

In fact, the requirements and business logics become more and more complicate, the software architecture has change to research the ways that could develop complicate distribute systems and applications [4]. In modern system, the software architecture pays more attention to decompose whole system to three parts, components, connectors and configuration, and depend public APIs to coordinate with each other [5]. To define the states of each components is a new challenge for developers. Moreover, in order to describe the states, the communication form of the components necessity to be exposed. Therefore, the "global view" could be a reference for design a distributed system or application. It could help developers to clear what time the interfaces should send or receive messages, and if there is no suitable information the system should how to work. This analysis can be done at the beginning of the software design could help development team to save cost and be able to eliminate some of the potential bugs.

ChorGram is a tool to generate global views depending the theory of choreographic [5,7]. ChorGram has already finished a wonderful job for developers. It accesses a special extenuation format file and generates several types graphs for users. ChorGram has developed by Python [9], but as same as most desktop applications, users have to configure the software environment at first. In addition, the shell command is the only way to operate this tool, so it is difficult for beginners. In this paper, we introduce a web version tool of ChorGram, namely WebChorGram. WebChorGram not only a web interface of ChorGram, but also added few new functions for uses. It provides the standard examples for learners and multiple methods to generate graphs. In

WebChorGram, it offers three kinds choreographic graphs, global view, machines view and transmit view. In section 2, we review the web technologies which suitable to develop the web version tool. Section 3 is the details of requirements and design schemes and implementing approaches. Section 4 is conclusion part, it analyzed whole work.

2. Background work

Relevant Works

In the past, a lot of desktop applications have been redeveloped web versions. A web application tool could more easily accepted by users, and it also improve the populate by Internet. For improving the functionality and user experience, the system used a lot of advanced web technologies and tools in development. In this section, it not only is introductions for these web technologies, but also explaining the principle for tools.

What is ChorGram

ChorGram is a tool to generate global views depending the theory of choreographic [5,7]. This tool uses CFSMs to validate the state of components and messages, and if the state satisfies conditions then the tool can create graphs. The CFSM is communication finite-state machines [8] that to resolve the behavioral of distributed components. These components exchange messages via specific channels, and the states of components change as the information is exchanged. In order to be able to reflect the process, ChorGram has a uniform command format, for example, A·B!bWin means that machine A will receive a message bWin from action A·B?bWin , and the state of A will from a0 to a1[5]. After verify commands by tool, then the tool generates choreographic graphs, includes global graph and machines state graph and transmit state graphs. And ChorGram accepts two kind of files, FSA and SSG. Each one has special format for the same purpose. The output of ChorGram is DOT files. DOT [18] is a graph description language which could describe undirected or directed graphs. Undirected graphs always used to define the parallel relationship between nodes. And

directed graphs are able to describe a tree or flowchart. For example, the architecture of a company. The DOT can be conversed to PNG, PDF and SVG file. Hence, ChorGram not only produces DOT files, but also can transform the DOT to other format files for displaying. The ChorGram has two key steps for creating graphs. First is constructing the DOT and PDF files. And then, the tool could generate JPG or PNG images from DOT files. Thus, the final outputs of ChorGram usually are PNG images and DOT files.

What is Python

Python is an object-oriented computer programming language. [9] Python has a lot of features could help developer improve the efficiency. Firstly, Python is easy to write and read, it has a graceful syntax. Secondly, Python has a powerful library to support and it can use C or C++ to extend easily. And Python not only can cross platforms and run anywhere, but also is totally free. In general, Python is a smart language and it could build a software quickly.

Why Choosing Django

In the current, there are serval web frameworks based on Python, such as Pyramid, Bottle, Tornado, Flask, Django and Web2py. However, Django has complete documentations, and it also has free and rich APIs [9]. Developer could easily implement functions according to these APIs. Nowadays, the Model View Controller (MVC) design pattern has becomes a popular approach for constructing web system [12]. M stands for "Model", it uses to access data layer, the data layer can implement the data model, if the system has database, the model layer can exchange data with it. V stands for "View" that shows data in web pages, and sends requests to server. C stands for "Controller". Controller layer contains logical methods and exchange data from data

layer and view layer. After the web pages sent requests, the controller will receive the requests and makes responses based on logical methods. Django has its special way to support the MVC which called MTV. M is as same as model layer that contains data information. V stands for "view", it is the business layer, this layer contains the logical codes, and it also processes data according the model layer. T stands for "Template", it uses to display information by web pages normally. In addition, Django not only provides various tools for building web application such as caching, logging, sending email and messages framework, but also supports editing URL format, developer could manage the URLs by themselves. Generally, Django is a feature rich framework and it also is a light framework for developers.

Why using HTML5 and JQuery

In the 2014, the W3C published HTML5 that hopes HTML5 can instead HTML4 and HTML1.1. In HTML5 added multiple features [13]. Firstly, the DOCTYPE declaration and the charset declaration has been simplified. Moreover, HTML 5 increased several tags to strengthen the page performance such as <svg>, <section> and <audio>. In addition, HTML5 expanded the API library to improve the interactivity with users. Although HTML5 provides very detailed tools to help developers to create beautiful pages. However, the job of page editing still cost time and patient. Thus, choosing a mature web framework could be a best way to handle page editing. Bootstrap is a famous web framework. It can quickly complete the page layout tasks, and can be compatible with a variety of terminals, not only web pages but also can be displayed on the mobile side [14]. Furthermore, Bootstrap perfectly supports HTML5 and is still free.

Web system wants to have a better user experience, it needs to improve the display effect and increase the interaction between users. In web design, there are three languages should be learnt, HTML, CSS and JavaScript. HTML and CSS are responsible for displaying, but JavaScript can manage the behaviors of pages [15]. In other words, JavaScript is a simple language to control the elements from HTML and web pages. However, when the JavaScript processing complex tasks, the amount of code line is large. Thus, in this project, using JQuery is preferred. JQuery [16] is a library which includes complete JavaScript methods, and developer can directly invoke these functions in web pages.

What is Graphviz

Graphviz [17] is a graph visualization software which generates diagrams in different formats such as JPG, PNG and SVG [22] for web pages and PDF for other documents. More precisely, Graphviz accepts the description which created by text commands, then transmits these commands to graph files. The format of text command has several types like DOT, NEATO and FDP. Additionally, in the final results, the color of nodes, or the fonts of messages or the line style can be changed with different instructions. Graphviz has a JavaScript version Viz.js. In web pages, developer could import Viz.js to implement the visualization function. However, the Viz.js only access DOT language.

3. Contribution

3.1 Requirements of project

Users can use ChorGram to read the FSA or SSG file by the specified path and generate PNG diagrams into another specified path. Although, the ChorGram has provided a great job for distribute system designers. However, the operating of ChorGram could not suitable for beginner, especially for students who are not familiar with Linux instructions. In WebChorGram, the first goal is that keeps original functions which as same as ChorGram. But, instead of using terminal, users could upload file directly by web pages. After user uploading the text commands, the WebChorGram can generate chorographic graphs and displaying on the web pages. Hence, the new web tool becomes not only more friendly for users but also more focus on producing graphs. In addition, WebChorGram could help students who are new for software design to understands the messaging passing system without coding.

In order to make ChorGram easier for beginners to understand. In WebChorGram, it offers the introduction of ChorGram in home page. Moreover, WebChorGram should provide few standard examples. this function could help users learn how to write FSA file in text commands. Furthermore, students could use these standard FSA files for producing the choreographic diagrams. And then, according to these pictures, students could analyze the relationship between each line of commands and graphics. As a result, the WebChorGram should provide a friendly method to understand ChorGram and chorography.

Based on the FAS file which has prepared by users, ChorGram produces not only one type of choreographic. However, these pictures all have been stored in a folder, users have to check them one by one. Thus, the new web tool should straightly display the three kinds

of graphs, such as global graph, communication machines graph and transition systems / execution graphs.

For users analyzing the choreography graphs conveniently, the WebChorGram should increase a function to help users to find out the connection between the text commands and graphs. For example, there is a new function which could highlight the paths and nodes after user selecting a line of command.

3.2 Design

According to analyze the first requirement, the basic processing can be described by figure 1 under blow:

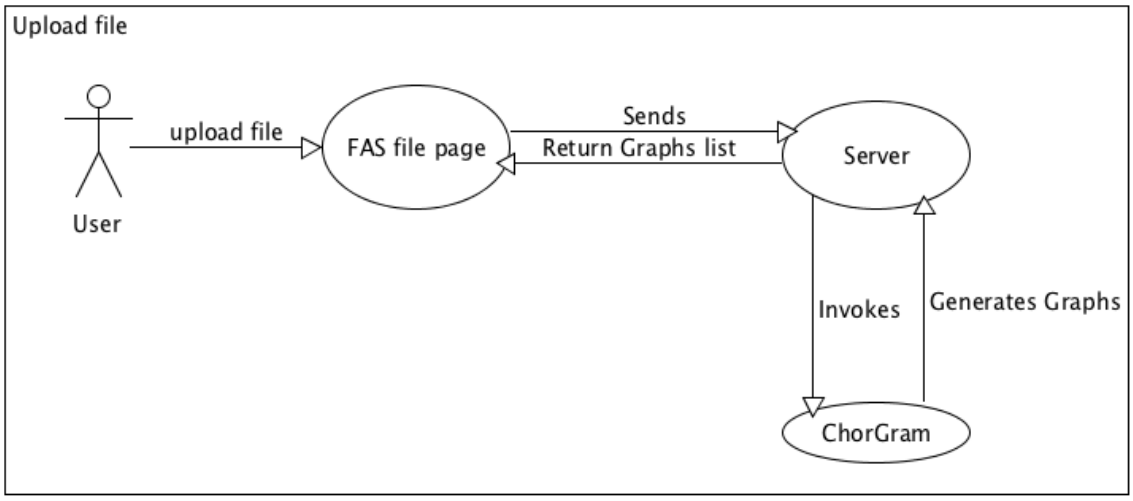

figure 1 Upload file process

After user uploading a FSA file, the upload page sends this file to server. Then, the server invokes ChorGram with passing the file path. If the file content is validated successfully, the ChorGram generates graphs into a specific document which the name of document as same as the file. After that, the server searches global graph, machines graph and transition graphs by the file name. In the next step, the server returns the image path list to page, and the upload page could load pictures by each path in list. However, if the FAS file has any problems that

the ChorGram cannot creates any image. And the list of path will be empty, and the page should display an error message for noticing user.

For second requirement, the WebChorGram should contain an example page. In this page, there are several standard FAS files in a list, users could choose any one for generating graphs. The details about the process is shown below figure 2:

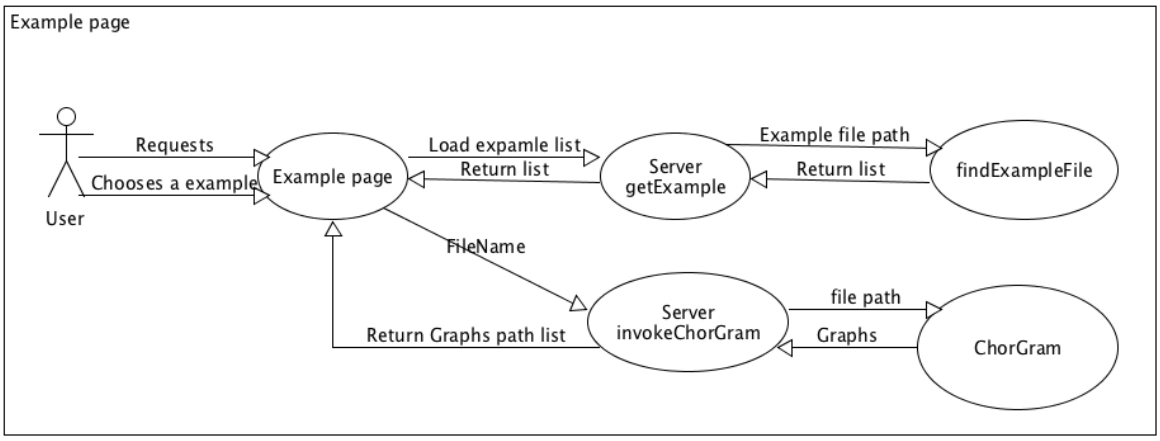

figure 2 Example process

When users visit the example page, the example list should have been prepared by server already. Therefore, the server should have a method for finging all example files by specific directions which have been prefabricated in the system. And the page can display all file names. Then, when user selects any one of file list, there is a page element shows all contents of the file. For example, using text area controls to present file content. If user generates graphs based on this file, the server can receive the file name by sending a request from page client. Furthermore, there is a function can invoke the ChorGram with the path of the files. After ChorGram finished its work, the server should return a list of all image paths to page, and then, the example page could load pictures.

WebChorGram should display three kinds of choreography graphs like global view, machines communication graph and transition graph. Although, the ChorGram product not only three types graphs, but these three types graphs are the most core graphs. According to introduction of ChorGram [5] that there are two steps for transforming a FAS file into graphs, the first step is generating the PDF and DOT files in a document which is under the path of "results". In the next step, the ChorGram created a document under the path of out document and the name of document is the name as same as the file. The global graphs, machines graphs and transition graphs all in this document. And when the page displays these PNG files, it should distinguish the specific graphs by key words like "global", "machine" and "ts". This part of requirements can be implemented by JavaScript on page.

Increasing interaction between commands and graphs is the last requirement. After user selected a line of command, the corresponding paths and nodes should be highlighted in the three types of graphs. And in the previous analysis of design, the final outputs should be several images. Furthermore, these images are created by transforming the DOT files. Hence, increasing a dimension in an image is changing the contents of DOT files. However, these DOT files have created in server, when a line of commands has been selected, the page should notice the server that there is which line of commands should be changed. Then, the server should produce new graphs for page by new DOT files. This approach which is through changing DOT file could implement the requirement, but it is not efficient and it could lead the load of server increasing. However, there is another solution that the HTML5 can display SVG file directly. SVG [22] stands for Scalable Vector Graphics, it is a xml based file for illustration graphs. Therefore, the server no needs to save PNG images instead of passing complete DOT files to page-client, and the page-client could use JavaScript for transforming DOT files to svg and displaying. As a result, when a line of commands has been

selected, the page should change DOT file directly without sending request to server. In addition, displaying graphs by svg could increase other interactions, for example the interaction from graphs to commands. The svg supports a lot of functions for controlling graphs. The details of this process is showing like figure 3:

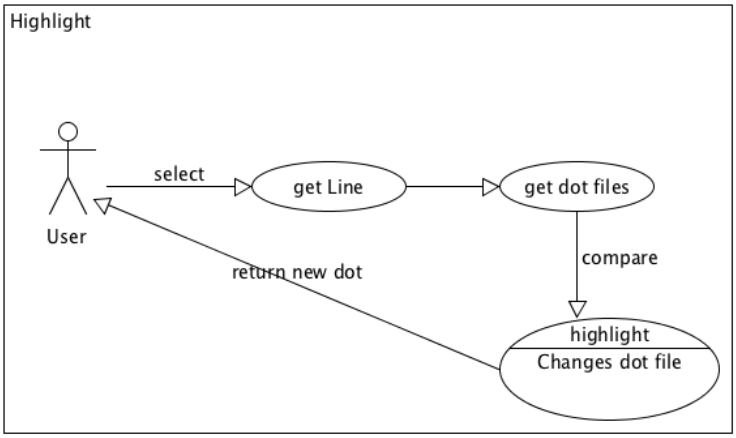

figure 3 Highlight

After user selecting, the JS method should get the line of commands and the original DOT files. Then, the method should send the line and DOTs to the highlight method, this method could create new DOT files by comparing operation. The last step is showing the DOT file by transforming with svgs.

In general, WebChorGram should include a short introduction of ChorGram, such as the wording principle and the rule of FSA. Secondly, WebChorGram should has some examples for beginners or students. Moreover, the web tool should display three types graphs and it also has uploading function to analyze the file which provides by user. Furthermore, the tool should support an interaction function for analyzing the graph. Totally, the web system contains four parts, home page, example page, upload file page and input page. In user case diagram like figure 4:

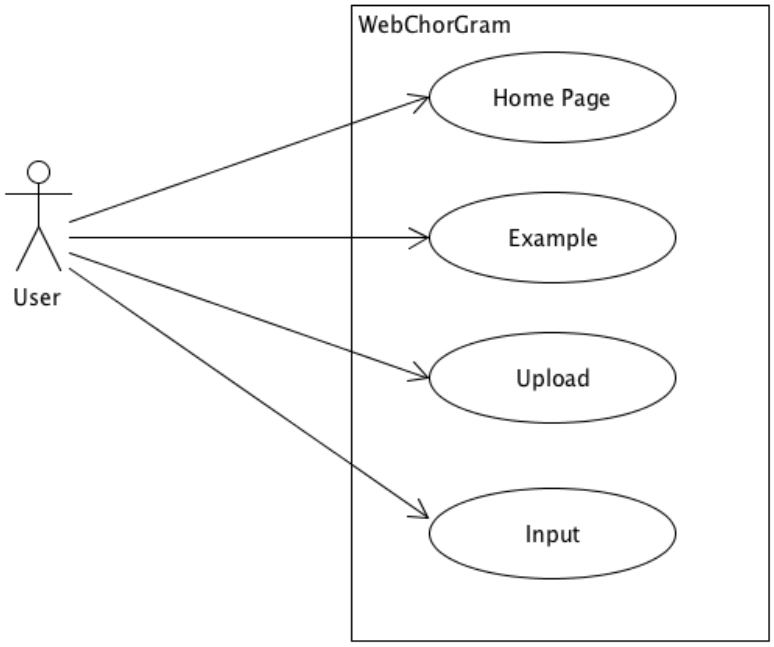

Figure 4 User Case Diagram

The content of home page is from introduction page of ChorGram for user understanding ChorGram easily. In example page, there are few different FSA file examples and these examples have been prefabricated in server. For upload page, user could upload any FSA file to server and get the outputs on this page. Input page has functions mostly the same as upload page, user could input any commands for generating graphs, but if these commands have errors, the sever will notice user. furthermore, each page should have a navigate for users linking to other functions.

3.3 Implementation

Initial work

The first job is configuration the development environment. Django has two versions, one is based on Python 2.7 and another is for Python 3. Because of the ChorGram is developed by Python 2.7 version, so we should choose the same one for keeping compatible. The

development is under Mac, the macOS has installed Python 2.7. That is simplify the configuration working. There are a lot of IDEs (Integrated Development Environment) can use to program the python project, but the PyCharm provides excellent features for development and it is free for education. Using PyCharm could create a Django project comfortably, and Django offers a simple server for access web project. After created a project, it includes an initial page for checking that the project is running.

In a Django project, there are several essential parts, such as init.py, settings.py, usrls.py and wsgi.py. Each one has its special functions. The init.py is an initial file which is empty for noticing Python that this project should be a Python package. Settings file is configuring project. Urls file is using to edit the access url for project. And the last file wigi.py offers an entry-point for web servers [11].

The second work is adding the ChorGram into the web project. Coping the folder of ChorGram into project. And there are few configuration works for ChorGram. Firstly, the ChorGram requires few libraries and tools to work together, such as Graphviz, Haskell platform, MissingH, HKC and Petrity. For each one the detailed introduction is in the introduction page of ChorGram [5]. The PyCharm supports test shell commands in Terminal function. After installing tools and libraries that using the testing FSA file to test whether the ChorGram is already to work. The project directory is follow figure 5:

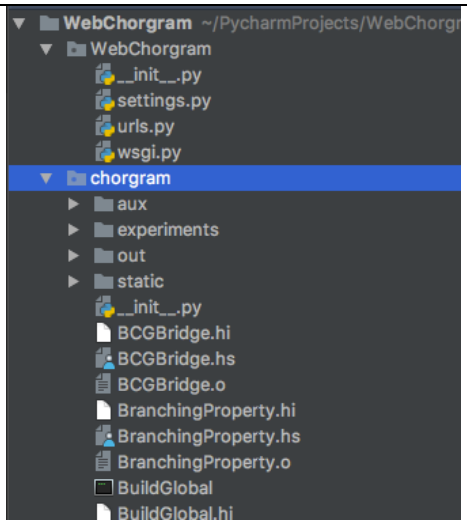

figure 5 project directory

Creating Home Page

Django project is consisted with several Polls apps. Polls app could implement the actual business logics. The WebChorGram included an webContext app to handle requests and responses. Polls app also has few core files for implementing its functions. To create a directory polls, there are few basically files to be created at same time. The admin.py is for registration models. Apps.py is recording the name of app to project. Furthermore, developer should create models in models.py. And the test.py is for adding test cases. View.py is the core file in an app, it is the business layer, and it could exchange information between pages and models. App also has a urls.py file, it includes all accessing information. In directory of webContext, it has a folder templates, the home page should be added in this folder. The Django accepts the page file is the html format. In the introduction of Django section, Django supports a special MVC design patterns MTV. Thus, the M is model.py, the V is view.py and the T is template folder. For example, if developer added a page into Django project, there are few setting works should be done. At first, we should understand the working process about Django in figure 6.

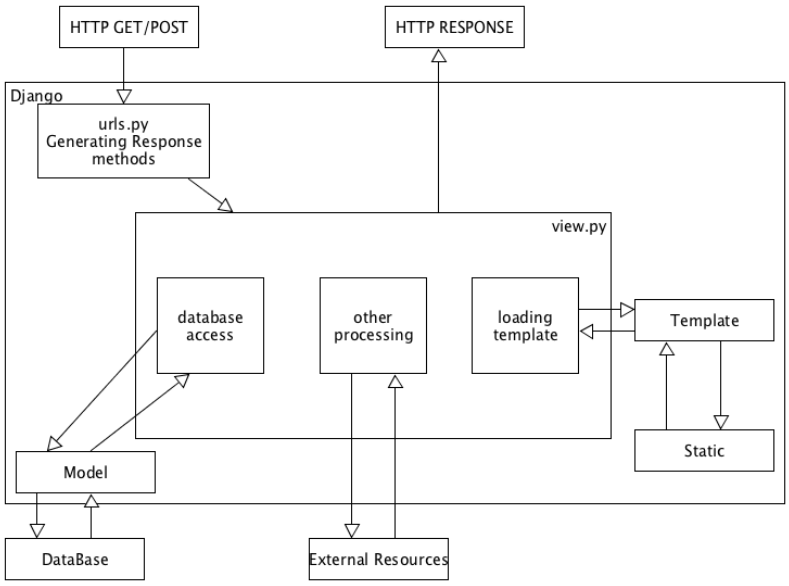

figure 6 Django Processing

User sends a HTTP request with browser by inputting a URL. When Django accepts the request, it checks which views include the specific methods. For example, if the codes of urls.py is like figure 7

```
url(r'^$',index,name='index'),
```

figure 7 url example in urls.py

when user accesses a link like 127.0.0.1:8000/index, Django will search which views including index method, and the method should produce the necessary data with models and prepare to load the relative template. In the next step, the method packages the data by HttpResponse and returns to browser. In general, when developer adds a page into the app, there are four steps should be done. First is editing url in urls.py. Then, creating model if it is necessary. Next step is writing codes for implementing the business logics. The last step is editing the template.

For building the home page, the first stage is adding a url in urls.py file like figure 7. The second step is creating model, but there is no necessary to create model in this function. Next step is adding a method in view.py. In this example, the name of method is index. In addition, the function of index method is finding index.html and linking to page. The last step is creating index page in template folder. After that, the system has built an index page already. However, the content of this page is empty, and according to design section said that the home page is an introduction of ChorGram. Moreover, the index page includes not only the introduction, but also a navigate for leading to other functions. Hence, the page should have layout. CSS is a language for controlling the styles of html page. However, using CSS to layout html page is a time costing job. Consequently, the WebChorGram adopted Bootstrap to control the styles of page.

Bootstrap has a grid system that divide the page into 12 columns [14]. This system is used for adding page layouts through sequences of rows and columns. The pages of WebChorGram have been spited two parts, one is side bar for navigating, another is content part. The Bootstrap offers a lot of css templates for user who is not profession on the layout of pages. The WebChorGram is using Material Dashboard [19] template, this template separates whole page into two parts, navigate part and content part. Moreover, this CSS template is not only based on Bootstrap, but also supporting HTML 5. On the other hands, Material Dashboard template has complete documentation, and it is free. For using the CSS template, before linking the css file into html, there is a preparation should be finished. That is creating a static folder under template directory in project and put the CSS file into it. When running the server of Django, the Django will scan all static files under the directory of static folder. If the page wants to load static resources such as images, JavaScript files and CSS files, that the page should add a line of code "{% load staticfiles %}" at beginning. Then, the

page could link to the specific css file. For using the components of Material Dashboard, it should use specific tags which has been defined in css file. After setting the style of index page, the content of page is copying from ChorGram introduction page. Although, the home page is a simple function in WebChorGram, but it reflects the basic understand of Django framework.

Upload file and generating graphs

The fundamental function of ChorGram is reading specific files and generating graphs. In WebChorGram, uploading file and generating graphs also are the basic jobs. The processing of these jobs could decompose two blocks, the first is building upload function. In the documentations of Django [11] said that Django receives file data from Request.FILES by keys for file fields. And this approach always handles the data from <form> component which is posted from pages. In the next step, Django could read data and save it into a particular path. However, using <form> tag for sending data has a problem that the page need refresh one time, then graphs can display in page. There is another way that could post data from page to server. That is using AJAX. AJAX is an asynchronous refresh tool for page, and it is offered by JQuery. AJAX helps page to improve user experience by seeing operating effects without waiting refresh. Therefore, WebChorGram chosen AJAX for posting data to server.

The steps of building upload page mostly are same as index page. The different part is the page content. In the index page, the content simply is a text for introducing ChorGram. However, the upload page not only includes upload module but also contains display module. Moreover, the display module should show three types graph separately, such as global graph part, machines graph part and transition graph part. As a result, the content of upload page should be divided to four parts. The basic component is an input tag for upload. However, the

style of HTML native file component is simple, and this style is not coordinated with whole page styles. Therefore, the system chooses a plugin Bootstrap File Input [20] to instead of original file component. This plugin is based on Bootstrap and HTML 5. More precisely, this plugin provides not only preview files but also multiple choice and dragging upload. Furthermore, File Input supports AJAX based uploads and showing upload progress, and more. In general, using this plugin is a simple way that leads developer to handle a technical file upload component for building web page with Bootstrap CSS3 styles. For example, there are several simple steps to add this plugin in upload page. At first, building the links for the File Input CSS file and JavaScript file into page. Then, add an input element and setting the value "input-709" of the id property. The following step is setting the values for AJAX and the main property is the URL address. Thus, the page work has finished.

In the back-end, there is a method upload_ajax for handling upload request in veiw. The details follow figure 8:

```
# ajax upload file
# if the user uploads the file then get the name,
# else if user inputs commands then generate a fsa file in upload file
@csrf_exempt
def upload_ajax(request):
    if request.method == 'POST':
        file = request.FILES.get('file')
        print(file.name)
        f = open(os.path.join(BASE_DIR, 'chorgram/static/uploadfile', file.name), 'wb')
        for chunk in file.chunks():
            f.write(chunk)
        f.close()
    else:
        file = request.FILES.get('file')
        print (file.name)
        # return HttpResponse({'filename':file.name})
        # return render(request,'pages/upload.html',{'filename':file.name})
    filename = file.name
    json_str = json.dumps(filename)
    return HttpResponse(json_str)
```

figure 8 upload method in view.py

As mentioned above that the file is sent by request.FILES. After the method got the file, Django should read the file content and save it into a defined path with file name. This path is

pointing to a folder which used to store uploaded files. At last, the upload method returns file name to page with JSON format.

The second block is producing graphs by ChorGram. As the ChorGram introduction said that using this tool needs two Shell instructions. Therefore, after the server received generation request, it should invoke ChorGram with two commands. The first command is building basic files like PDF and DOT, and the second is transforming DOT files into PNG images. Python provides a library subprocess [9] to launch new processes for invoking other input or output pipes. The particular function is call method. It has two main parameters for building connection with ChorGram, one is String type value which contains the details of order, and another is Boolean type value which is to demonstrate whether the method running in the Shell. The key codes are shown below figure 9:

```
def runByShell(name):
    messages = 0
    command1 = 'cd chorgram\n python cfsm2gg.py -nc -dw 0 -df pdf -dir static/results/ static/uploadfile/'
    operate1 = command1 + name
    splitname = name.split('.', 1)
    command2 = 'cd chorgram\n python cfsm2gg.py -df png -b 1 --dot "Nshape=parallelogram" --dot "Gnodesep=.5"
                -p "w *" -tp "'+splitname[0]+' * *" -cp "* * '+splitname[0]+'" static/uploadfile/'
    operate2 = command2+name;
    try:
        subprocess.call(operate1,shell=True)

    except:
        messages = 1
        print messages
        print 'operate1 failed!!!'
        return messages
    try:
        subprocess.call(operate2, shell=True)
    except:
        messages = 2
        print messages
        print 'operate2 failed!!!'
        return messages
    return messages
```

figure 9 runByShell() in view.py

At beginning, the method defines two variables for storing Shell instructions. However, the ending of input file direction is the name of file without extension. Thus, the method has to split name to two strings. Besides, using try-except module is to ensure that whether the

subprocess.call() is execution normally. If any exception happens, the except module should send a notice in the terminal. The return values of Call method usually are two types, if the commands to complete normally, the method returns 1 and else returns zero. Therefore, there is a potential bug could be existed in system that through call method the system could invoke ChorGram successfully, but if any error happens during the execution of ChorGram, the program will not return an illegal value. However, this problem is not unresolved. The system could check if the ChorGram has produced the required files. Although this approach still cannot get the detail of error, but it has already solved the bug. After execution ChorGram, there are several PNG images in the specific directory. However, the result folder includes not only images but also DOT files. Thus, before server returns value to page, the graphs generating method should filter all images. The different between images and DOT files is that they have dissimilar file extensions. The last operation of server is returning image path list to page. Because of the ChorGram changes image names by adding the key words for distinguishing the graph types, such as global, machines and ts. When the page received the image path list, it could filter graphs by these key words and display pitcutes to relevant positions. In order to improve the performance of the upload page, when the page is showing images, the original module which is displaying the upload controls will be instead of a text area element, and the content of text area will be filled with contents of the uploaded file. Then, users could watch the different graphs and commands at same time. This changing could improve the efficacy of analyzing the design for users.

Example page development

The example page is showing instances which are prefabricated in WebChorGram. As mentioned in system design section that user can select a case for generating graphs. The

layout of this page is similar with upload page that the page content should be divide to four parts. One parts is a text area for showing commands and others are illustration graphs. However, the different between upload page is that there is a file selection function for users to choose example. In page editing, Bootstrap provides dropdowns component for building value selection function. This component is consisted of two parts, first is a button for showing which value has been selected. Second part is a dropdown menu. It contains all values for users. However, for WebChorGram these values are instances. In addition, these instances load from server. Therefore, when user access example page, the server should prepare a file list for page. In view.py, there is an example method for handling this request. It using a listdir() function to get a list which includes all file from specific paths. And it packages the list and return to page by HttpResponse. Then, the page sends each value into dropdown menus from the list. However, there only the file names in the list. After the user chosen a file, the text area should display all commands by file. Because of the page cannot access the content of a file which stored in server directly. Hence, this function was implemented by JavaScript and AJAX. At first, an on click function should be added in dropdown component, it could invoke AJAX to get commands through server by file name. The view has a method could read the contents according this file name, and return the texts to example page. Then, the page loads the texts into text area and sets the read only property is true. As a result, user could see all commands from the file which is selected.

The next step is displaying graphs. There are two ways to implement. The first approach is that the system should prefabricate not only example files, but also the corresponding graphs. Then, the example page could show these graphs directly after user selected file without computing. The another approach is that the page posts commands to server, and then the server invokes ChorGram for producing images. These two methods have their own

advantages. First method could reduce the number of server operations, but it has to development a special function for returning images, and if the developers wants to add new example, they should also add graphs. Thus, the difficulty of system maintenance should be increased. Although, the second function may regenerate graphs for same example, but for increasing new examples, it is simple to increases the files in system. Moreover, if the original examples have to modify, it also simple operations for changing or replacing files in example folder directly. Furthermore, the second function could improve code reusing, it can call the methods which have been achieved for other completing functions. Therefore, the WebChorGram used the second way to implement graphs production. And the display is same way as upload page that according to distinguish key words for presentation images.

Increasing interaction in page

In the requirements, an important demand is adding the interaction between commands with graphs. In the design section said that after user selected a line of instructions in text area, the page should reload new graphs, and these graphs were produced from server, but the system should transform DOT files into SVG files for showing. Thus, in the development of input page, the server should return DOT file to instead of the path list of images. The layout of input page is similar with example page, but the distinct is users could input commands in text area. After inputting, the page posts all commands to server. For server, the view.py contains a method receiving these commands and writing them as a file for ChorGram. Then, the get_file_return_dot(request) is invoked, and this method calls ChorGram for producing DOT files. One thing should be noticed that the global DOT file is stored in the different folder from DOT files. Thus, get_file_return_dot(request) has to check global DOT file alone, and adds the name into returning list. Although, the values of returning list only are the file name, but

that could prove that the required DOT files have been generated successfully. The details method code is under blow figure 10:

```
# Testing according the file to generate dot and return to page
def get_file_return_dot(request):
    filename = request.GET.get('filename')
    splitname = filename.split('.', 1)
    #print (splitname[0])
    name = filename
    outPath = 'chorgram/static/out/' + splitname[0]
    paths = [os.path.join(BASE_DIR, 'chorgram/static/out/' + splitname[0]),
             os.path.join(BASE_DIR, 'chorgram/static/results/' + splitname[0])]

    cleanMessage = cleanFile(paths)
    if (cleanMessage == 0):
        print 'file clean done'
    exeMessage = runByShell(name)  # run(name)
    print exeMessage

    PICS = os.listdir(os.path.join(BASE_DIR, outPath))
    if (len(PICS) == 1):
        result_list = 1
        return HttpResponse(
            json.dumps(result_list),
            content_type='application/json'
        )
    # chorgram/static/out/ no global.dot file , need add
    pathForGlobal = 'chorgram/static/results/' + splitname[0]
    globalDotFile = searchFile(os.path.join(BASE_DIR, pathForGlobal), 'global', '.dot')
    PICS.append(globalDotFile)

    result_list = PICS
    # filter(lambda x: x.find(".png"),PICS)
    # print 'result_list', result_list
    return HttpResponse(
        json.dumps(result_list),
        content_type='application/json'
    )
```

figure 10 get_file_return_dot method in view.py

At first, the variable filename is for getting the file name which posted from client-side, then the method needs splitting the name between name and extension. The reason of splitting operation is the folder name is same as filename without extension, but for searching the target file operation, it needs the whole name including the extension. Before invoke ChorGram in runByShell() method, the output document should keep empty state. It could ensure the results of runByShell() that are not be affected by the previous operation. Then, searching the DOT files in result directory, but the global file will be not stored in same path of other DOT files, so it should be searched alone. The final step is packaging all results in a list and returning the list to client-side.

There is a new problem that how transform DOT file to SVG format and displaying. To solve this problem, the web system has used Viz [21]. In the background section mentioned that Viz is the JavaScript version of Graphviz. Viz can convert DOT files into PNG, SVG or JSON format by JavaScript. Using viz is simple that links the viz.js file in web page and then calls Viz() function. More precisely, the core parameters of this function are two, one is string value stands for the source, and another is the target of format, such as svg, png and json. For original displaying in JavaScript methods, after filter the key words, the method loads the path of images directly. However, for showing method in input page, it only contains the DOT file name, no image and path. Therefore, it is necessary to obtain the contents of the DOT files. The AJAX should be used, through AJAX could get data from server. For server that received the request of getting content by DOT file name. The method get_dot(request) is working. It could search the file and read it line by line, then package the data for returning. In addition, the global DOT file needs to find from special path. After the client-side received the content of DOT file, placing the content into Viz() and setting the format is svg. Then, the svg file should be generated. Although this svg file could display on HTML5 page, but it cannot be control the size. When the size of svg file is larger than the screen resolution, the effect of display could be very poor. svg is a xml based file for displaying graphs and it is scalable, but the svg which generated by Viz is not a file but a package of string values. Thus, HTML controls cannot modify these values. For fixing this problem, it is necessary that converts these strings to a xml document. The DOMParser [23] object has a function for building XMLDocument object from XML formatted text. Through this function, the package of string type values could change to a xml document object, then the HTML controls could correct the size of svg automatically that based on the CSS style. In addition, using svg file to instead image could get another advantage. There are several HTML components can adding animation for svg, such as <object>, <embed> and <iframe> tags.

Through different animation effects, the page could increase different functions. This is a foundation for further expansion for this system. For example, the new requirement is click the part of graph, the relation commands could be annotated. The input page core codes for using Viz and DOMParser are following figure 11:

```
if(item.toString().indexOf("dot")!=-1)
{
    if(item.toString().indexOf("machines")!=-1)
    {
        var filename = item.toString();
        var dirName = name;
         $.get("{% url 'webContext:get_dot' %}", {'filename':filename,'dirName':dirName}, function(ret){

         }).done(function(data) {
                //push dot data for highlight
                machinesDot.push(data);
                //display svg
                var parser = new DOMParser();
                 var img = Viz(data,"svg");
                 var svg = parser.parseFromString(img,"image/svg+xml");
                 addZoomforSvg(svg,"#result_machines");
                 $('#machines').show();
                });
    }
```

figure 11 display graph with svg in input page

The first two if statements are checking that if the filename which returned by server is includes "dot" extension, and the key word "machines" is distinguishing the file whether a machine graphs. After that, the program obtains the DOT content by AJAX. Then, the program converts the DOT file into a svg, then transforms svg to a XMLDocument object for display.

After changing the way to load the graphs by svg, the preparation of the interaction has been completed. In the design section pointed out that changing diagram is changing DOT. For highlight specific paths and nodes function, the next step is analyzing the relationship between FSA commands and the DOT. For example, there is a FSA file like figure 12:

```
.outputs A
.state graph
q0 1 ! hello q1
q0 1 ! world q1
.marking q0
.end

.outputs
.state graph
q0 0 ? hello q1
q0 0 ? world q1
.marking q0
.end
```

figure 12 a case of fsa

After ChorGram, the machines DOT file like figure 13:

```
digraph CFSMs {
 graph [color=white ratio=compress margin=0];
   subgraph cluster_A{
   label = A;
    Aq0 [style=filled, color=cornflowerblue]
    Aq0     [label = "q0"];
    Aq1     [label = "q1"];
    Aq0 -> Aq1  [label = "A&middot;M1 ! hello"];
    Aq0 -> Aq1  [label = "A&middot;M1 ! world"];
   }

   subgraph cluster_M1{
   label = M1;
    M1q0    [style=filled, color=cornflowerblue]
    M1q0        [label = "q0"];
    M1q1        [label = "q1"];
    M1q0 -> M1q1    [label = "A&middot;M1 ? hello"];
    M1q0 -> M1q1    [label = "A&middot;M1 ? world"];
   }

}
```

figure 13 the machines DOT file

For comparing these two figures that all codes should be divided to two parts. In the upon part, that is the story of machine A. In FSA file, each machine has two states q0 and q1, after A sends messages "hello" or "world", the state of A will change from q0 to q1. Thus, a result could be speculated that the DOT file should include two nodes and two paths. And in DOT

file, the line 6 and 7 are presenting two nodes q0 and q1. In addition, the line 8 and 9 are standing for two paths from q0 to q1 and the labels are "! hello" and "! world". According to observer these two graphs that there are several rules in them. For the line of FSA commands q0 1 ! hello q1, it includes two nodes and one path which is from q0 to q1 and the label is ! hello. And for the machines DOT file, the 6 line is presenting a node q0, and it contains key words q0 and [label = "q0"]. Moreover, the path from q0 to q1 contains "->" symbol and message "! hello". Thus, in the program, these features are the evidence for comparing with other instructions.

In general, the development of highlight function could be decomposed into 5 steps. First step, according to user selection that the system should confirm the selection is which line of commands. However, when user clicking the text area controls, the component only will return an order number. This number stands for the position of current clicking in all characters including the line feed. Therefore, if the system requests to realize which line command is selected, it has to recognize how numerous characters the entire command has, and the count of line breaks are there before the clicking position. After that, when the system obtains the line of commands. The program should extract key words for definitude the targets with searching in DOT file. Fortunately, the syntax of FSA is fixed. Normally, like figure 12, the first and second lines present that there is a machine A existing. And the third line explains that the state of machine A will change to q1 from q0 by sending or receiving a message. Certainly, the content of message can be changed. More specifically, the third line is composed of five parts, and they are connection by space character. In development, the system could decompose a line of command by space and get the length of character array, then, the extracting keywords operation could have based on position in array. For example, for the commands "q0 0 ? world q1", the positions of nodes is 0 and 4, the message position

is 2 and 3 in array. The third step is comparing each line of DOT file with key information which achieved in step two. If there is a line that satisfies the conditions, then the system has to change the ending of this line command with adding ", color=red]". Thus, this nodes or path will fill by red color for highlight. The key codes are following figure 14:

```
function changeMachine(dot,command)
{
    var dotSliptArra = dot.split('\n');
    var commandArra = command.split(" ");
    if(commandArra.length!=5)
    {
        return null;
    }

    for (var i = 0; i< dotSliptArra.length;i++) {
        var tempDot = dotSliptArra[i];
        if(tempDot.indexOf(commandArra[0])!=-1 && tempDot.indexOf(commandArra[4])!=-1
            && tempDot.indexOf(commandArra[2]) !=-1&&tempDot.indexOf(commandArra[3])!=-1){
            tempDot = tempDot.substring(0,tempDot.length-2);
            dotSliptArra[i] = tempDot+',color=red];';
        }

    }
    var newdot = dotSliptArra.join('\n');
    var parser = new DOMParser();
    var img = Viz(newdot,"svg");
    var svg = parser.parseFromString(img,"image/svg+xml")

    $('#result_machines').children().remove();
    addZoomforSvg(svg,"#result_machines");

}
```

figure 14 changing DOT file

At beginning of the code is splitting DOT file and FSA commands by line break and space respectively. Then, if the length of character array is 5 that the method compares the key information with every line of DOT file and adds a highlight sign in line of DOT. The next step is regenerating the new DOT file. The last step is using Viz() for recreating graphs and showing. However, before showing new graphs, the page should clear the old graphs. As a result, the highlight function should be completed, but the original page presenting three different type graphs. Although, the above section only analyzed how to highlight in machines graphs, but the principles are same for other graphs, the important part is getting key information for comparison.

In complicated FSA case, because of the page layout has a constraint on the size of modules, the figures show that they will be smaller and difficult to see the details. The WebChorGram added graphic zoom function for each graph. There is a JavaScript library svg-pan-zoom [24] for svg zoom behavior. For using this library, it needs to get the id value to confirm which component has loaded the svg. Then, the library has a svgPanZoom() method to implement the zoom function. This method includes several key parameters should be configured, such as zoomEnabled stands for if the zoom function opening, controlIconsEnabled means that whether the control buttons should be presented and fit property is the graph filling automatically. Svg-pan-zoom is an outstanding library for zooming svg file.

Generally, after adding zoom function in WebChorGram, this web tool has already implemented basically. However, in development of input page, the system has changed the way for presenting graphs from using images to svgs. Thus, the input page gained several useful functions than other pages, for instance the highlight function. The next development direction is sharing these new features to other pages. The JavaScript can code into the HTML or become an independent JavaScript file for pages to link. In order to optimize the functionality of the entire system, the system should reedit all the JavaScript methods for building a new file for pages. And the except home page, other pages through linking main.js to implement all functions. Thus, not only these pages have the same approach to loading graph, but also fixing bugs simply.

3.4 Test

White Box Testing

White box testing is focus on the internal logic of codes. Therefore, the server programs part should use white box testing. Because of the core method is invoking ChorGram, hence the first test is on runByShell() method. The flowchart about this function is figure 15:

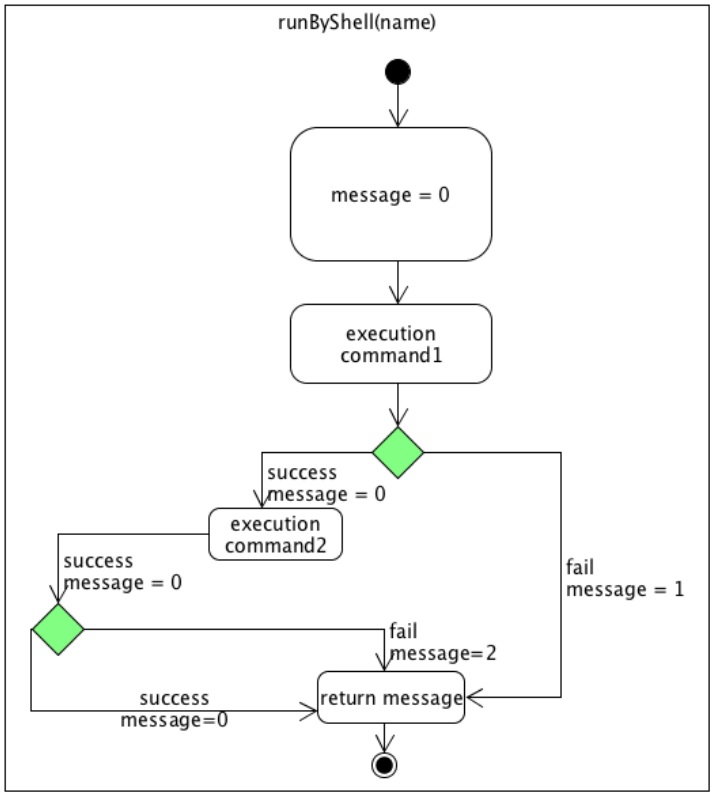

figure 15 runByShell method

In order to have the full coverage, the following are test cases:

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Two instructions	command1 has errors, command2 is right.	1
2	Two instructions	command1 is right, command2 has errors.	2
3	Two instructions	command1 has errors, command2 has errors.	1
4	Two instructions	command1 is right, command1 is right.	0

The results of this test:

Through inputting two commands to test the method was running correctly. The after inputting command 1, if command 1 not has errors that the program should execute command 2 continually. If command 2 is right that the program should return 0. However, if command 1 or command 2 has errors the program should return 1 or 2 immediately.

After testing, the returning values all of tests is 0. The reason is execution command by subprocess.call(), this method only create a channel for program executing command in Shell, and it is no matter what happened in Shell. Hence, the final value always is zero.

The next test is for searching file method, this method could find the file based on the given path, file name and extension. The search depth includes all subfolders by recursive method. The method flow chart is following figure 16:

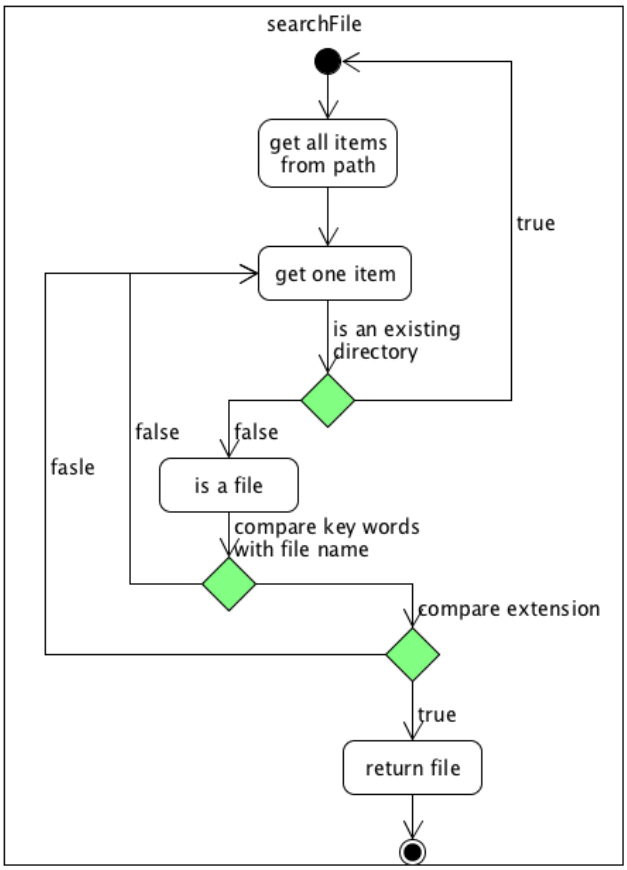

figure 16 searchFile processing

In order to have the full coverage, the following are test cases:

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Path: "home" File name: "test" Extension: "doc"	wrong path wrong file name wrong extension.	Exception message: 'Search file failed!!'
2	Path: "static" File name: "pingpong" Extension: "fsa"	wrong path, right file name right extension.	Exception message: 'Search file failed!!'
3	Path: "lib/js" File name: "pingpong" Extension: "png"	wrong path right file name wrong extension.	Exception message: 'Search file failed!!'
4	Path: "lib/js" File name: "ttt" Extension: "fsa"	wrong path wrong file name right extension.	Exception message: 'Search file failed!!'
5	Path: "chorgram/static/results " File name: "ttt" Extension: "txt"	right path wrong file name wrong extension.	Exception message: 'Search file failed!!'
6	Path: " chorgram/static/results " File name: "pingpong" Extension: "exe"	right path right file name wrong extension.	Exception message: 'Search file failed!!'
7	Path: " chorgram/static/results " File name: "ttt" Extension: "fsa"	right path wrong file name right extension.	Exception message: 'Search file failed!!'
8	Path: " chorgram/static/results " File name: "pingpong" Extension: "fsa"	right path right file name right extension.	Return " pingpong.fsa "

The approach and results:

The test approach is call searchFile method with parameters from each test case. In each test case, there are three attributes path, filename and extension. And each attribute has two types value wrong or right. Thus, there are eight test cases should be executed. The results of these test cases that for test case 1 to 7 the program executed with exception, and returned null value, but for test case 8, it returned the file name normally. Thus, only the path, name and extension are all existing in system, then the system should return normal value. As a result, the test for searchFile method was passed.

Integration Testing

Integration testing belongs to black box testing. It is a useful test strategy for web project. According to requirements the first test is upload file for generating three kinds graphs and

displaying in web page. This test should be divided into two parts. The first part is testing the whether the file can be upload into server.

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Txt file	Upload	Fail
2	PNG file	Upload	Fail
3	FSA file	Upload	Fail
4	Doc file	Upload	Fail

The approach and results:

The WebChorGram should accept two types file, FSA and SSG. Hence, there are prepared several extensions file for uploading, such as "test1.txt", "test2.png", "test3.fsa" and "text4.doc". When using page to upload txt file, the page appeared a waring message for noticing user that this upload function only supports "fsa" file, and the upload process does not continue. Furthermore, for test2, test4 files, the system has send same message to user, but the test3 file can upload successful. As a result, the upload function is running normally.

Second part is whether the system can produce correct graphs for showing:

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Normal file	Generating Graphs	successful
2	File has error	Generating Graphs	Fail

The approach and results:

There are two test cases, test case 1 is a correct FSA file, and test case 2 has few errors. The result of test case 1 is normal, the page can display graphs successfully. And for the test case 2, the page showed a caution message that prompts user the graphs were not generated. According to these evidence can be proved that the upload and graphs generating function is working correctly.

However, during the test processing, a special bug has been found. After uploading file, if user press generation button twice, the system will produce same graphs twice, and display on same modules in page. In other words, the system presents the results twice on the page. The reason of this bug is that before displaying new graphs, the page should clear up the results from previous operation. For fixing this problem, it needs to add a cleanup function in JS file for resetting the page content.

The next test is for example page:

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Average-new-comm.fsa	It can be found in example page, and showing content in text area controls	successful
2	Average-new-comm-nosend.fsa	It can be found in example page, and showing content in text area controls	successful

The approach and results:

The test details are increasing new example files in system, then checking whether the dropdown menus and the text area controls can load these examples. The test processing is that increases two new files in example direction in server. The next step is visiting the example page to search these two files in the dropdown menus, then select each one and observe if the file content appears in text area component. The final result was the example has passed this test.

The following test is testing the highlight function.

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	q0 1 ! hello q1	Selecting this command, and check the graphs has right highlight paths and nodes	Find out the paths q0 to q1 with message "! hello" in each graphs
2	.outputs A	Selecting this command, and	Find out all nodes name

		check the graphs has right highlight paths and nodes	including A in all graphs
3	.marking q0	Selecting this command, and check the graphs has right highlight paths and nodes	Find out q0 nodes and the node has filled the color in all graphs

The approach and results:

The test strategy is through clicking the commands to check the highlight paths and nodes are correct. The test case is similar with figure 12. For results of test case 1, the path q0 to q1 with message "! hello" was be marked by red in machines graphs. And in global graph, the node label showing "A -> M1: hello" was be marked by red color. For ts0 graph, the path labeled "A->M1: hello" was be highlighted, and for ts1 graph, the path which has the same label as ts0 graph highlighted. According to these results the test case 1 passed.

Test case 2 is finding nodes. In machines graph, the nodes q0 and q1 under title A were marked. In the global graph, both two nodes were highlighted. In ts0 graph, both two nodes were highlighted. In ts1 graph, the two nodes which are in the middle were marked. The results of test case 2 were correct.

For test case3, in machines graph, the nodes that at top of graphs were marked. In global graph, there no any node was marked. In ts0 and ts1 graphs, there was no node marked. The results of test case 3 were right.

According to each result of test case, the highlight function is running normally.

In general, the core functions in WebChorGram have passed test successfully, but it still has few bugs not found yet. During operating these test, there are few other small bugs have been fixed. For instance, when user selects line of commands, there is a bug that if the click position in the middle of line, the page has possible determined the wrong number of lines.

4. Conclusions

Through a period of effort, the WebChorGram not only has most of the functions of ChorGram, but also adds new features for users. It could help students to understand ChorGram simply. The home page and example page could produce a way to study chorography efficiently. In addition, the students also could input themselves commands through input page or upload page to confirm their conjectures. Furthermore, this web system also could be used in analyzing distribute system design. WebChorGram could decrease the complexity of using ChorGram, the WebChorGram can create chorography graphs without configuration and Shell instructions. Moreover, the highlight function is a useful feature for analyzing. Although, the highlight function only offers top-bottom approach currently, but still could bring convenience with system designers. And, the WebChorGram could improve the rate of using by Internet.

In the development, the WebChorGram tried to use several the latest web technologies, such as HTML5, Bootstrap and JQuery. These web technologies bring powerful functionality and user experience for this system. More importantly, the WebChorGram is based on Python and Django web framework. The Django is the most popular web framework in recent years, and Python is the fastest growing programing language. There are a lot of advanced system and ideas were implemented by Python like machine learning and big data. Thus, the WebChorGram has a good scalability, and it can be seamlessly docked with other advanced systems. The layout and operations of WebChorGram have consulted several the mature same type web sites, such as viz-js.com and webgraphviz.com. They are good examples that development web interface for a tool. Therefore, the intuitive and operation simplify are the standards of designing WebChorGram.

The WebChorGram could support diverse features for user. However, it still has several disadvantages. The first of all, if the FSA file has errors the system could only feedback error messages without details. Thus, the users fix errors difficultly. The second point is that the WebChorgram only provides interaction from commands to graphs, but the interaction between graphs to commands is not achieved. Furthermore, the system only could generate graphs with FAS file which the file is not containing grammar errors, but if the file has logical bugs, the system cannot notice users. In addition, the WebChorGram cannot set the styles of graphs. Hence, the display effect is not variety enough.

In general, the WebChorGram has built a completed web system and established the foundation for the expansion in the future. Although, the system has insufficient points, but it has already prepared for user to use chorography analyzing distributed system.

References

[1] Paulo Veríssimo and Luís Rodrigues. Distributed Systems for System Archites. Springer Science & Business Media, 2012., pages 26-27, 2012.

[2] Pradeep K. Sinha. Distributed Operating System: Concepts and Design. Wiley-IEEE Press, 1997., pages 114-166.

[3] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, and Y. Lafon. http://www.w3.org/TR/2004/ WD-ws-cdl-10-20041217. Working Draft 17 December 2004.

[4] Issarny V, Bellissard L, Riveill M, et al. Component-based programming of distributed applications[M]//Advances in Distributed Systems. Springer Berlin Heidelberg, 2000: 327-353.

[5] Julien L, Emilio T, Nobuko Y. A Tool for Choreography-based Analysis of Message-passing Software. River Publishers, 2017.

[6] Z. Micskei and H. Waeselynck. Uml 2.0 sequence diagrams' semantics. http://home.mit.bme.hu/~micskeiz/ sdreport/uml-sd-semantics.pdf.

[7] Julien Lange and Emilio Tuosto. ChorGram: tool support for choreographic deveelop- ment. Available at https://bitbucket.org/emlio_tuosto/chorgram/wiki/Home, 2015.

[8] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. JACM, 30(2):323–342, 1983.

[9] Python Documents. https://wiki.python.org/moin/BeginnersGuide/Overview (accessed 02/05/2017)

[10] Django Introduction. https://docs.djangoproject.com/en/1.11/intro/overview/ (accessed 02/05/2017)

[11] Django Document. https://media.readthedocs.org/pdf/django/1.11.x/django.pdf (accessed 02/05/2017)

[12] The Django Book. http://djangobook.com/model-view-controller-design-pattern/ (accessed 02/05/2017)

[13] HTML5 Introduction. https://www.w3schools.com/html/html5_intro.asp (accessed 02/05/2017)

[14] Bootstrap. http://getbootstrap.com (accessed 02/05/2017)

[15] JavaScript Tutorial. https://www.w3schools.com/js/default.asp (accessed 03/05/2017)

[16] JQuery API. https://api.jquery.com (accessed 05/05/2017)

[17] Graphviz. http://www.graphviz.org (accessed 03/05/2017)

[18] DOT file. https://en.wikipedia.org/wiki/DOT_(graph_description_language) (accessed 04/05/2017)

[19] Material Dashboard. http://www.creative-tim.com/product/material-dashboard (accessed 02/05/2017)

[20] Bootstrap File Input. http://plugins.krajee.com/file-input (accessed 03/05/2017)

[21] VIZ. https://github.com/mdaines/viz.js/ (accessed 06/05/2017)

[22] SVG. https://svgontheweb.com (accessed 03/05/2017)

[23] ParseFromString Introduction. http://help.dottoro.com/ljcilrao.php (accessed 02/05/2017)

[24] SVG-pan-zoom library. https://github.com/ariutta/svg-pan-zoom (accessed 02/05/2017)