JOHANNES KEPLER
UNIVERSITY LINZ

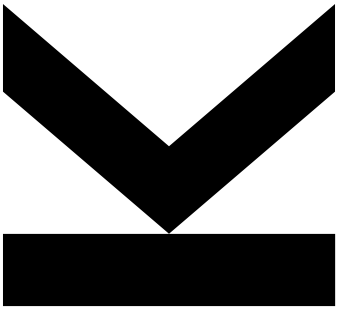
Submitted by
Hsuan-Ming Chen

Submitted at
Research Institute for Symbolic Computation

Supervisor
Prof. Wolfgang Schreiner

April 2019

Migrating Mathematical Programs to Web Interface Frameworks



Master Thesis
to obtain the academic degree of

Master of Science
in the Master's Program

Internationaler Universitätslehrgang: Informatics: Engineering & Management

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Str. 69
4040 Linz, Austria
www.jku.at
DVR 0093696

ABSTRACT

A mathematical software system, the RISC Algorithm Language (RISCAL), has been implemented in Java; however, it can be only executed on the local machine of the user. The aim of this master thesis is to migrate RISCAL to the web, such that users can access the software via a conventional web browser without needing a local installation of the software. In a preparatory phase, this thesis evaluates various web interface frameworks and how these can be executed on the web. Based in the result of this investigation which compares the advantages and disadvantages of the frameworks, one framework is selected as the most promising candidate for future work. The core of the thesis is then the migration of RISCAL to the web on the basis of this framework and the subsequent evaluation of how the demands have been met and how well all of the RISCAL programs are working after the migration.

ACKNOWLEDGMENT

First of all, I am grateful to Professor Bruno Buchberger for providing the opportunity to attend this program and giving every resource and help which he can support. Second, I am truly thankful to my advisor, Professor Wolfgang Schreiner, for offering me everything I need and solving the issues which I encountered during the research. Third, I appreciate having all the assistance from the assistant of the program, Betina Curtis, who assists me to deal with the administrative work. Finally, I want to express my sincere appreciation to my parents for always supporting me. The success of this thesis is indispensable without them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

As the progress of the Internet grows, handheld devices and personal computers become more and more popular. There are billions of programs on the Internet which help people to deal with their daily business. Many applications do exist, but most of these only run on specific devices. The problem arises that users have to install different software kits on their local devices to have a perfect system environment to use the software. One solution is Software as a Service (SaaS) [1] which allows users to use the programs on the cloud via the Internet or a web browser, for example Web Mail Service, Web Storage Service, etc. SaaS is now a necessary trend for programs.

An example of software which can only be executed on a local machine is the RISC Algorithm Language (RISCAL). RISCAL is “A specification language and associated software system which has been developed in Java for describing mathematical algorithms, formally specifying their behavior based on mathematical theories, and validating the correctness of algorithms, specifications, and theories by execution/evaluation” [2][3][4]. It is problematic that users have to locally install Java and the Standard Widget Toolkit (SWT) [5][6] libraries to run the RISCAL software.

RISCAL is one example for a large scope of mathematical/scientific software that has been originally developed for using on local machine, but which shall be nowadays used on the web. For mathematical software, five use cases, Mathematica [7][8][9], MATLAB [10][11][12], RStudio [13][14][15], Maple [16][17][18] and Geogebra [19][20][21] have been studied in this research. These cases are successful examples of originally desktop-based programs and which have then been implemented as web versions. While some features might not be operating when using the web version of these software, this still gives more convenience for users to solve the problems easily by using this web version.

To build a web-based software, plenty of frameworks, e.g. ASP.NET, Django, Laravel, etc., are available for developments. ASP.NET [23] is developed by Microsoft and designed to produce dynamic web pages. Django [24] is a high-level web framework which is based on Python for rapid development. Laravel [25] is a PHP-based web framework and designed for MVC architecture implementation. However, finding a suitable solution among countless frameworks is also one of the problems. To transfer a Java program to the web, there are various frameworks which can be used to solve the problem, e.g. Remote Application Platform (RAP) [6][26], Java Server Pages (JSP) [27][28], OpenXava [29][30][31], and Google Web Toolkit (GWT) [32][33].

To be able to understand above-mentioned frameworks more, this thesis has defined a set of requirements which is determined by the functionality of RISCAL and then developed a test program for evaluating various web interface frameworks according to these requirements. The best framework has then been chosen according to the result of the evaluation. On top of this framework, the web interface of RISCAL has been then implemented and deployed to a web platform. The difficulties to choose a good framework which meets the requirements and reduce the functionality loss during migration has been a main point of this research.

This thesis is organized as follows: In Chapter 2, five successful mathematical software systems and four web interface frameworks for Java are discussed. In Chapter 3, the functionalities of RISCAL are demonstrated in detail. In Chapter 4, criteria are defined and experiments are performed that evaluate the suitability of the frameworks according to these criteria. Chapter 5 discusses the use of the chosen framework for migrating RISCAL to the web: the system architecture, the implementation, the web deployment, known bugs, and unimplemented features are documented. In Chapter 6, the software is evaluated and the difference between the old system and the new one after porting to the web is elaborated. In Chapter 7, a conclusion indicates that the migration of RISCAL to the web has been successful. The appendix provides installation instructions for users and developers.

2. State of the Art

In this chapter, Section 2.1 will introduce some existing mathematical software systems and how they can be accessed via the web. Section 2.2 describes various web interface frameworks. Finally, Section 2.3 draws some conclusions about these frameworks.

2.1. Mathematical Software Systems

Mathematics is the power of problem solving, abstract or creative thinking, and plays an important role in our daily life, without which the world cannot be operated orderly. Therefore, there are countless examples of successful scientific software which help human beings to deal with problems and make decisions, for example, Mathematica, MATLAB, RStudio, Maple and GeoGebra. Furthermore, RISCAL is similar to above of programs. Following will briefly give an introduction to the aforementioned programs.

Mathematica

Mathematica [7][8][9] is a technical computing program for scientific, mathematic and computing fields, which is developed by Wolfram Research and can be programmed in the Wolfram Language. It provides a notebook interface which allows users to implement features including machine learning, visualization, statistics, etc. to interact with the system.

Figure 1 shows a simple polynomial expansion example on the web interface of Mathematica. The web version of Mathematica only provides built-in codes to demonstrate the basic function, which means the users cannot write the code by themselves on the web; instead, by using the code provided by Wolfram to see how the code interacts with the function and finally output the results. Compared with the desktop version, the web version is more like a helping document.

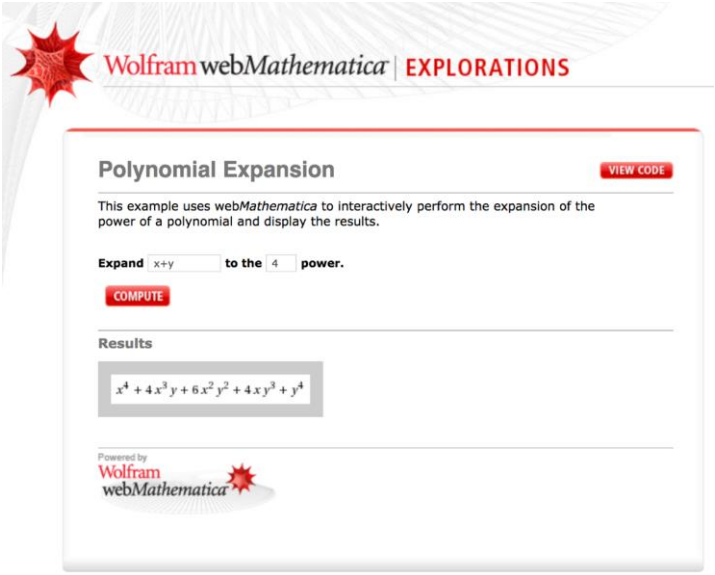


Figure 1: Web interface of Mathematica

MATLAB

MATLAB [10][11][12] developed by MathWorks is a commercial software focusing on numerical computing. It allows users to manipulate matrices, plot functions, create interfaces and interact with other programming languages, including Java, Python, C++, etc.

Figure 2 shows a basic matrix operations example on the web interface of MATLAB. The web version of MATLAB provides a live editor which allows user to insert/modify/delete the sample code and execute the code by using the web browser. Furthermore, the notebook style interface gives users a clean description of how the code is operated. However, users can not download the sample code by using the live editor on the web site, only with the desktop version installed in the local machine they are able to do so. The web version provides minor features, compared to the desktop version.

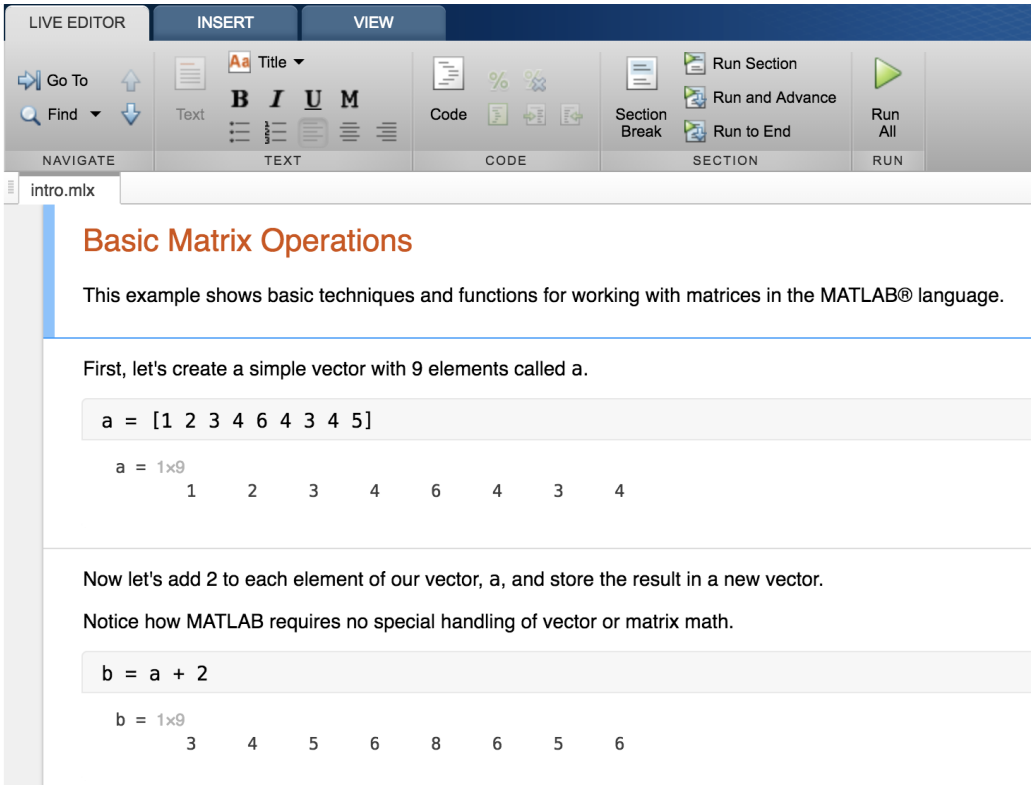


Figure 2: Web interface of MATLAB

RStudio

RStudio [13][14][15] is an IDE for the R language, a programming language for statistical computing. It is popularly used in data science field such as data analysis and data mining. R libraries provide a variety of graphical features including classification, clustering, etc. RStudio can also be manipulated by the programming languages C++, Java, Python, etc.

For the web interface of RStudio, Figure 3 presents the screenshot of the RStudio cloud. The web version of RStudio offers most of the functions and the user interface seems no different with working on the desktop version. It is capable to create/delete script files on the cloud, insert/modify/delete any code which is produced by the user, execute the code in real-time, and download anything which the user has done on the web. The only limitation will be executing the code which requires much of operating time. RStudio is an open-source software and allows developers to construct their own cloud in order to build a multi-processing platform to process more complicated problem faster.

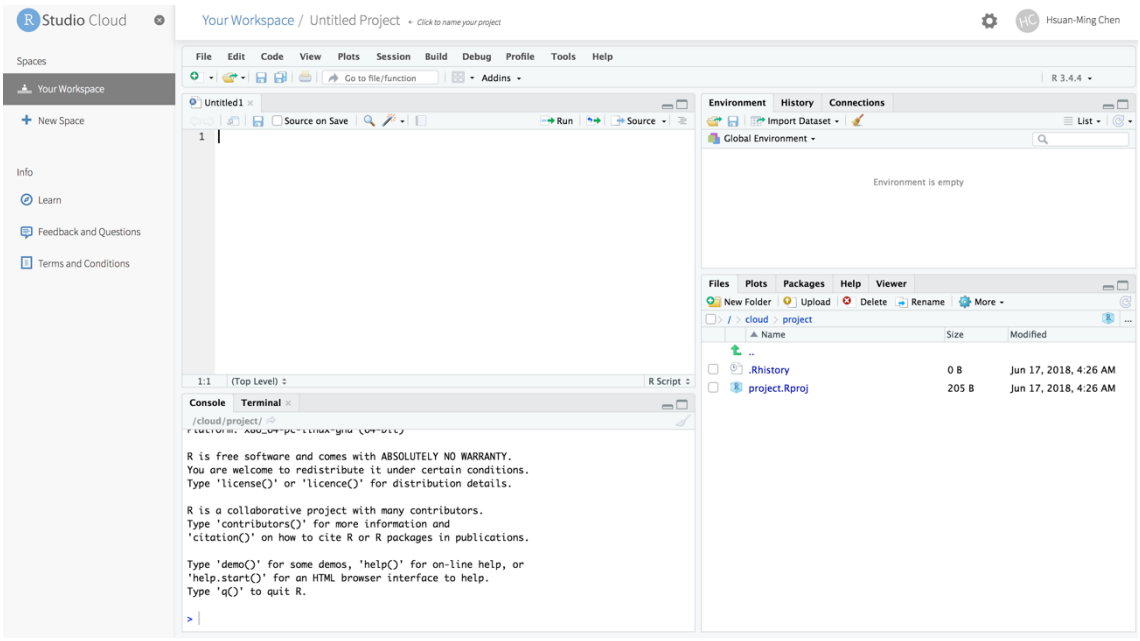


Figure 3: Web interface of RStudio

Maple

Maple [16][17][18] is a software of symbolic and numerical computing which has been developed by Maplesoft. It combines various aspects in technical computing, which easily allows users to analyze data, visualize mathematic functions, and solve scientific problems. It uses the Maple language and libraries to perform mathematical computations.

Figure 4 gives an example on the web interface of Maple by one interactive vehicle ride and handling tool. In this sample, the user can modify the numeric value of each parameter and run the simulation to calculate the overall evaluation. By using the web version of Maple, users are allowed to utilize the app which is developed and published on the platform by Maple or other users. However, users need to install the desktop version of Maple to download and operate the code. The web version of Maple is more like a platform which is combined with mathematical programs.

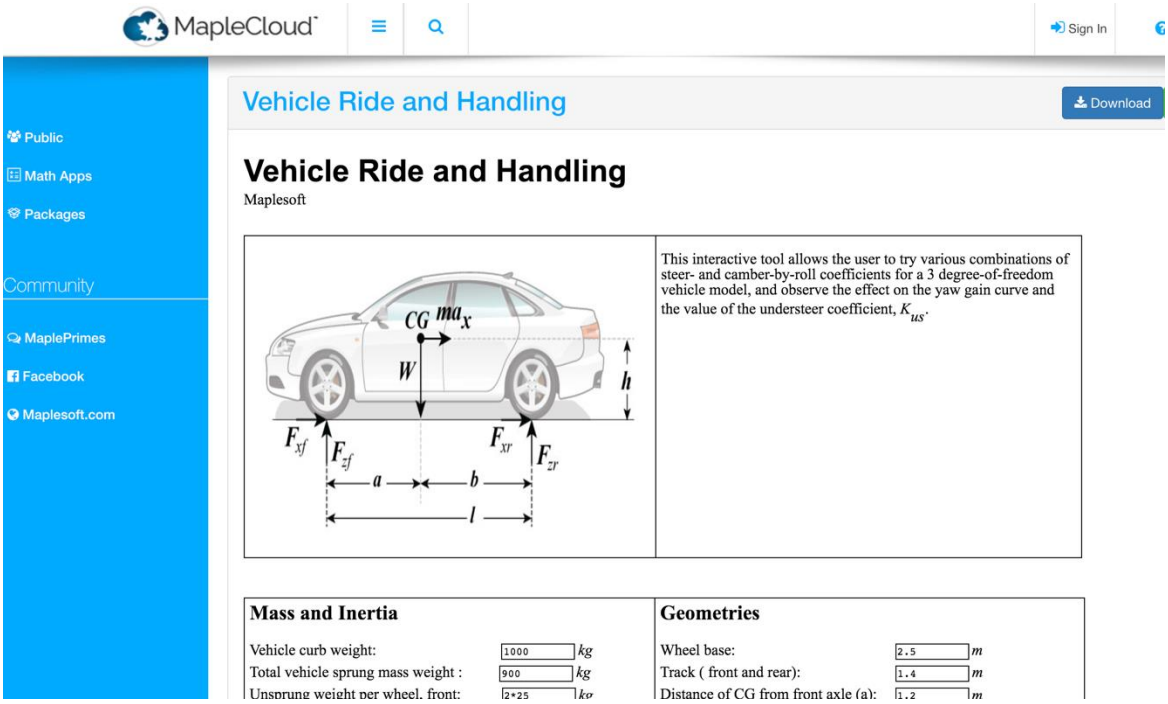


Figure 4: Web interface of Maple

Geogebra

Geogebra [19][20][21] is an educational application for teaching and learning geometry and other mathematical fields such as statistics, algebra, calculus, etc. in Mathematic field. It provides a user-friendly interface which allows users to implement mathematical functions and do visualization in the web. Users do not need any programming background to use the software.

Figure 5 illustrates the web interface of the classic Geogebra application. Users can easily toggle to other sub-applications, for example, graphing, geometry, etc. The software gives basic operation to create points, lines, and graphics by inputting function on the web. Besides, it allows users to import and export the application by other developers. On top of that, users can download or print the results. Furthermore, it supports mobile devices to use all the software which are developed by Geogebra.

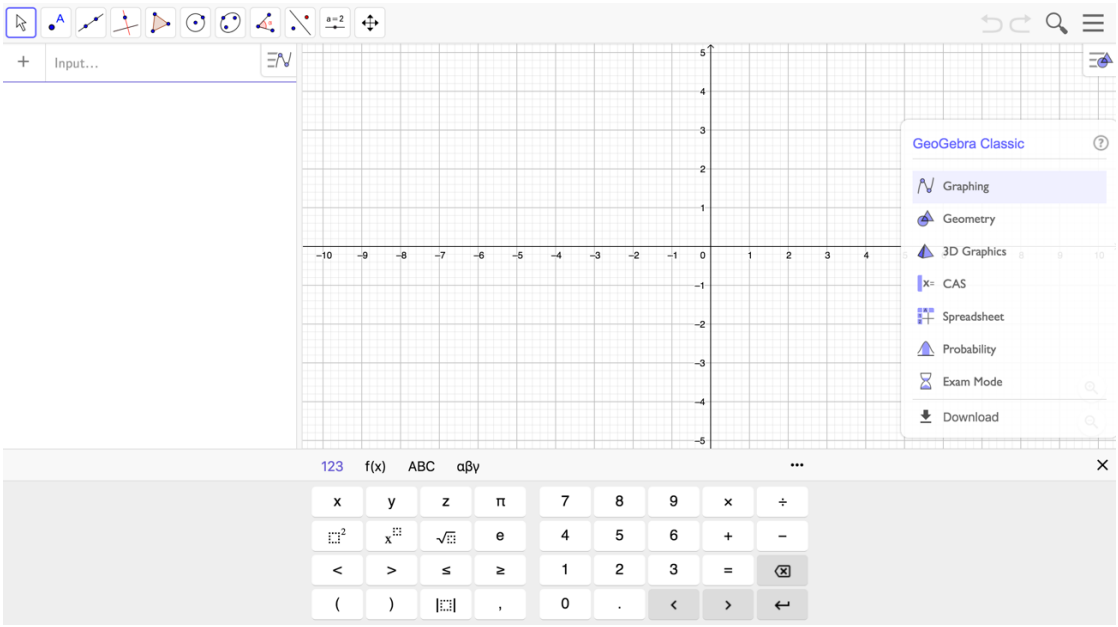


Figure 5: Web interface of Geogebra

With the help of above software, it is not necessary for people to develop difficult scientific applications by themselves, but they may simply learn how to use the software to resolve problems on the web, which brings more convenience for developers and users nowadays. On the other hand, this also represents a great achievement for educational purposes.

2.2. Web Interface Frameworks

In order to develop web programs, developers should understand the basic concepts of web technologies such as the HyperText Transfer Protocol (HTML) [22], Cascading Style Sheets (CSS) [22], JavaScript [22] and server configuration. In the past, Java programs could be executed in the web by special add-ons (plugins), but recently most of the web browsers have stopped supporting Java add-ons. To achieve the demands of web development by using Java, developers should consider other solutions. Therefore, four possible methods (RAP, JSP, OpenXava and GWT) will be demonstrated in this section.

2.2.1. Remote Application Platform (RAP)

RAP [6][26] is an open-source software which allows web developers to build Ajax-enabled rich Internet applications [6][26] by using the Eclipse Integrated Development Environment (IDE) [6][26] and a Java-only Application Programming Interface (API). In addition, it provides abundant developer documentations, server APIs, and resources integrated with other technologies.

The predecessor of RAP is Rich Client Platform (RCP) which is built on SWT and JFace libraries and operated on the top of operating system. SWT [5][6] is a graphical widget toolkit using native GUI for Java to create widgets on Java Native Interface. RAP is established on the RAP Widget Toolkit (RWT) library and operated on a servlet container. RWT [6][26] is an alternative widget library of SWT but quite similar to it. The client has to use HTTP to connect with server via web browser with RWT installed. The remaining parts preserve the same functionality as RCP. Figure 6 shows the architectures of RCP and RAP.

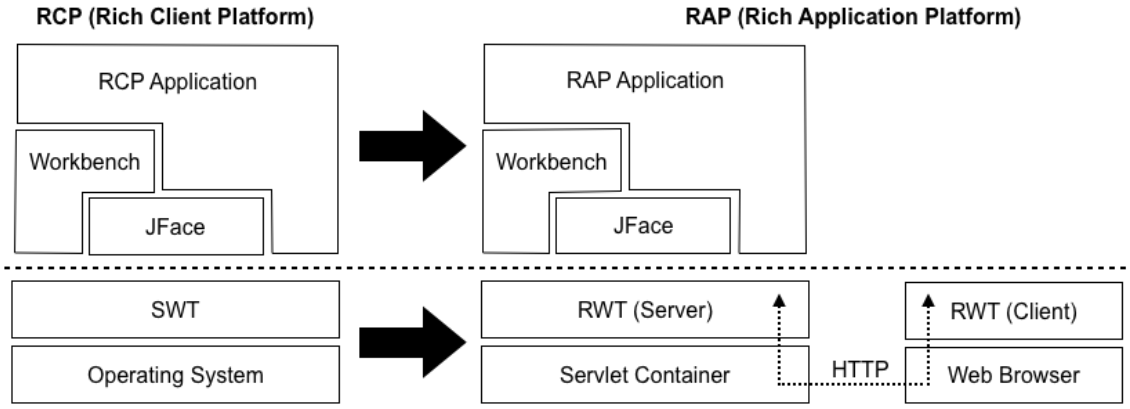


Figure 6: RCP/RAP architecture

To sum up for RCP/RAP, SWT is developed for desktop applications and RWT is used to build web applications. However, for an application which uses SWT, it can be executed on RWT with little or no modification. It seems like the migration of RCP to RAP will only have to revise the SWT to RWT and deploy the program on the servlet container.

2.2.2. Java Server Pages (JSP)

JSP [27][28] is a kind of HTML or XML pages, which is aimed at providing a rapid development to create dynamic web pages. However, JSP uses the JavaServer Pages Standard Tag Library (JSTL) as additional specific JSP syntax and Java code embedded in HTML or XML pages. A special JSP compiler can convert JSP to Java source code, but the web server needs to be equipped with the Java execution unit to execute Java source code. Web developers who are familiar with HTML can use JSP without learning the Java language.

JSP specifies a special JSP engine so that the server can recognize .jsp pages instead of .html pages. When the user sends a request to a JSP web page, the JSP engine will load the JSP pages from disk and convert these into a servlet content, which means that all the JSP content will be translated into Java code. After the JSP engine has compiled the Java code to a Java class, it will execute the Java class and respond to the client in HTML format. Figure 7 shows the JSP architecture.

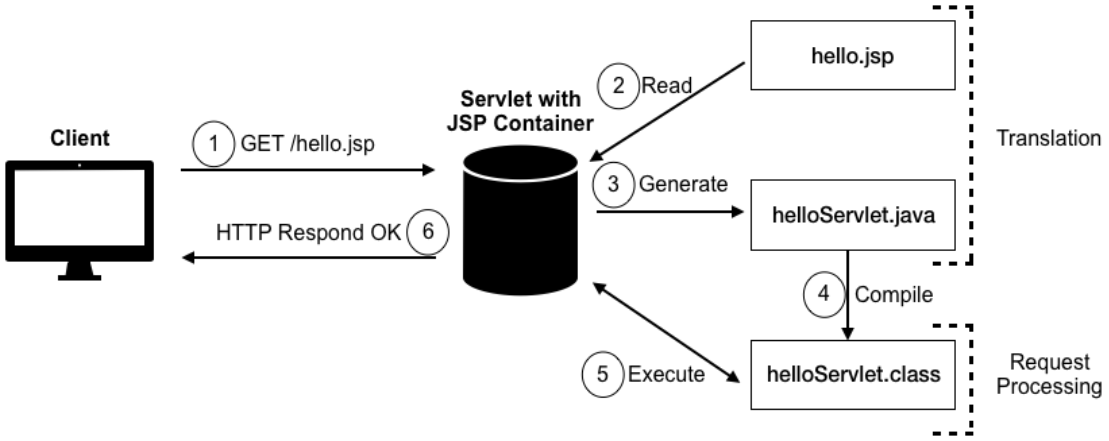


Figure 7: JSP architecture

According to the JSP architecture, for those pieces of software which are already developed in the Java language, for the migration from desktop to the web one has to figure out how to configure the connection between Java applications and web pages. However, the other possible way will be to rewrite the original program into the JSP format, which leads to more efforts to do the migration.

2.2.3. OpenXava

OpenXava [29][30] is an Ajax Java framework and an open-source software for the rapid development of web applications. Developers who are familiar with the Java language do not need to know HTML, CSS, JavaScript, etc. to implement a web program, which means the development is done by only using the Java language. Furthermore, OpenXava is integrated with thirty-party tools and based on Java standards. Thus, developers can easily migrate the Java source code to OpenXava.

Unlike RAP and JSP, OpenXava is a lightweight model-driven [31] oriented framework. Figure 8 shows the Model-Driven Framework of OpenXava. It combines the method of Model-Driven Development (MDD) and the concept of Model-View-Controller (MVC) architecture where the code is divided by data (Model), interface (View), and logic (Controller). OpenXava uses simple Java classes for the model definition. The application will be dynamically generated during runtime. Instead, it is quite different then the classic MDD which uses the Unified Modeling Language (UML) / Domain-Specific Language (DSL) and code generation. Nevertheless, the core of OpenXava is still an MVC-liked architecture.

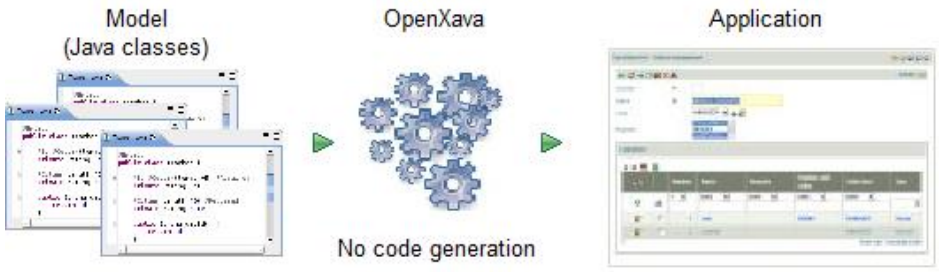


Figure 8: OpenXava model-driven framework [30]

The original design of OpenXava is aimed at the rapid development of applications by using Java. To revise a Java application into a MVC-liked framework is not only an enormous project but also violates the original design of OpenXava. Using OpenXava may allow to rapidly develop a project which starts from the beginning, but for Java software which was already been developed, it may not be suitable.

2.2.4. Google Web Toolkit (GWT)

GWT [32][33] contains Java API libraries, widgets, a compiler, and a development server, which allows users to develop Ajax applications written in Java. It compiles the source code to a JavaScript file, so that developers can directly deploy the software on the web browser. Moreover, GWT also provides a plugin for using the Eclipse IDE.

GWT is a framework which uses JavaScript for frontend and Java for backend. It offers a Java-to-JavaScript compiler which is suitable for desktop applications to migrate to the web. Figure 9 shows the GWT architecture. For the user interface, GWT uses the GWT Web UI class library to create controller and user interface. While developing a program, it applies a development mode to test the Java code on the local web browser without compiling to JavaScript. Once the development is finished, it can then be compiled to JavaScript and deployed on the servlet container. To exchange the Java objects over HTTP, developers need to set up a GWT RPC framework. Since the GWT RPC framework does not use the same protocol as web services, developers have to integrate a GWT RPC service into the application.

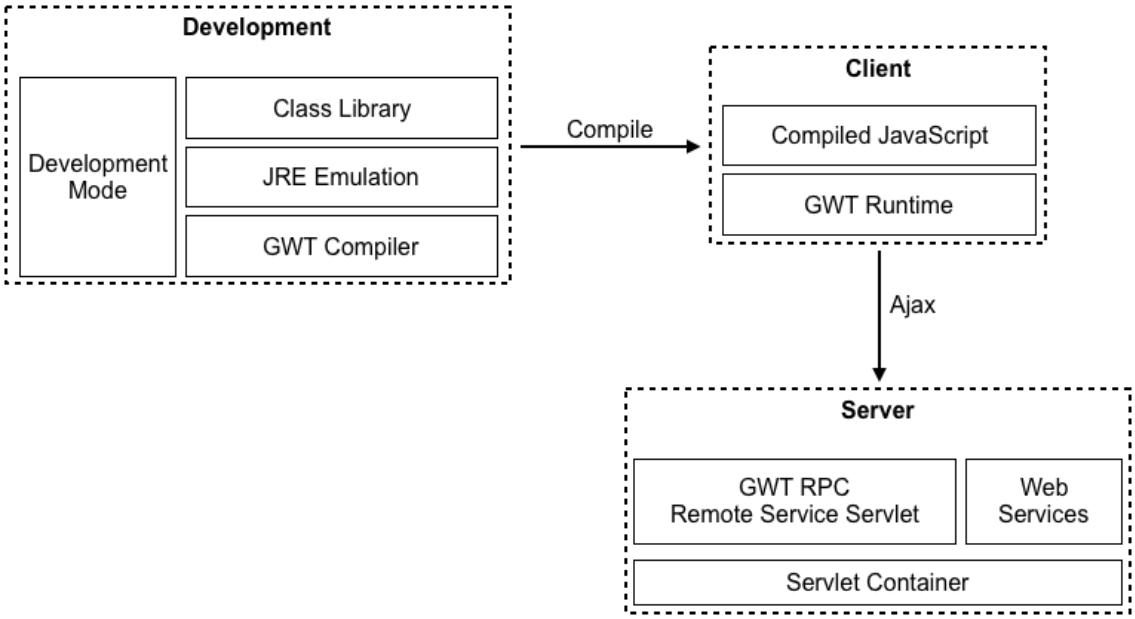


Figure 9: GWT architecture

The Java-to-JavaScript compiler provided by GWT seems to be attractive for any Java program which demands migrating to the web, but it still needs an additional GWT RPC framework to provide a complete operating environment. Furthermore, the best advantage of using the GWT framework is the scalable responsive development with all the Java development included. The process of migrating a Java program is to implement the remote service servlet to handle the Java object exchanging.

2.3. Summary

There are advantages and disadvantages for above-mentioned frameworks. While these frameworks claim that they are able to do rapid development, possess abundant functionality, are well integrated with third-party tools, etc., it is hard to find any research performing an integrated evaluation of them. Java developers are therefore easily confused when forced to choose among these frameworks. Our research will therefore discuss the evaluation of each framework after the implementation of some simple examples and provide a reference for Java developers, which is one of the original points of this paper.

Most of the foregoing mathematical software can be integrated with JSP in one's own web pages, for example, MATLAB, Mathematica, RStudio and Maple. But not all of them can be easily and successfully migrated to the web by using RAP, JSP, OpenXava, and GWT because of the different programming languages used in these developments and the problem of commercial licenses or company conditions. However, Geogebra [34] is one of the most successful mathematical software using the GWT compiler to create web-based platforms. All the web-based platforms of Geogebra are using JavaScript code, which is compiled by GWT from the original Java code. This illustrates that GWT may be one of the simplest ways to migrate a Java based program to the web. Though this is only one of the cases, we cannot ensure that GWT will be the best solution for an already existing Java software.

In summary, by understanding the aforementioned architectures, it provides us a better-known aspect for different frameworks. Since the RISCAL software is developed by using RCP framework, the simplest way to do the web migration is to use RAP for the implementation which is based on an architecture that is similar to RCP. However, the functionality, API, etc. may not be completely and successfully operating. Therefore, it will be necessary to evaluate the functionality of each framework in detail. The next section will describe in depth how each framework performs according to various criteria that we will define.

3. The RISCAL Software

The RISCAL software implements an educational language in the field of mathematics and used for formally specifying mathematical algorithms and theories. It is a system to check the correctness of mathematical algorithms, specifications, and theories. This section will explain each part of functionality in the RISCAL software. Figure 10 shows the main window after launching the RISCAL software.

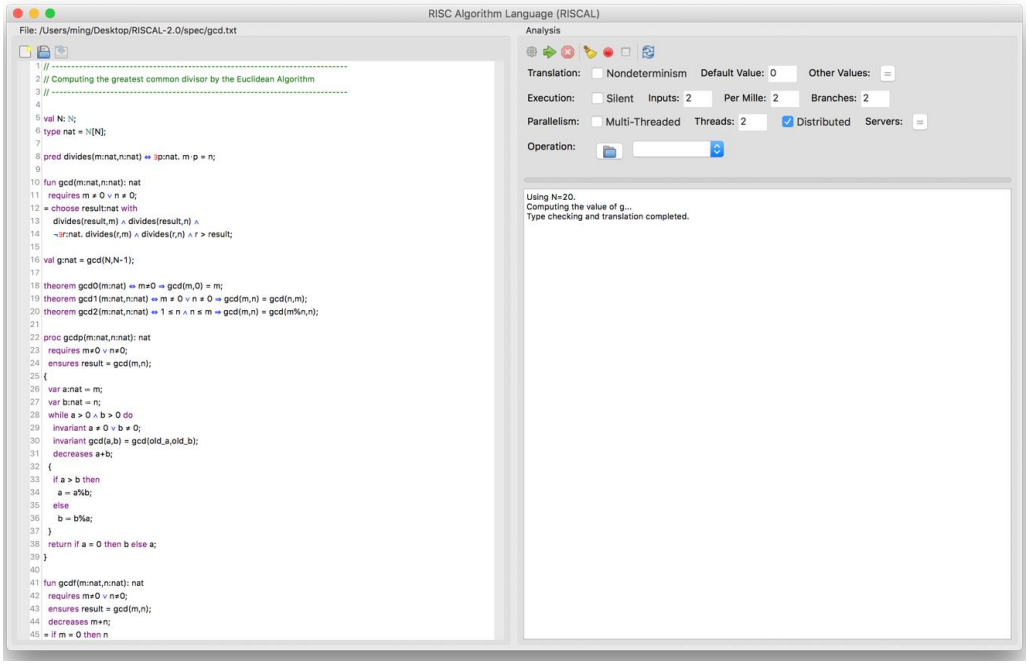


Figure 10: The main window of RISCAL software

The main screen is divided into two important sections. The left part of the main window provides a frame for users to write a specification in the RISC algorithm language and to perform basic file I/O operation, e.g. opening a new file, loading an existing file, and saving a file to the disk. Furthermore, it shows the currently used file location on the left of the main window. On the other hand, the right part of the main window outputs the analytical result on the console frame, which specifically indicates the syntax errors, semantic errors, current operation, etc. Besides, the top of the main window offers a menu bar including “process specification”, “start/stop execution”, “clear output”, “start/stop logging” and “reset system” buttons. The other optional functions which are shown under the menu bar use either checked boxes or option boxes which allow users to dynamically adjust the executed operation, including multi-threaded execution, distributed execution on a server, etc.

After users have opened a new file or loaded the file from disk and executed the analysis, the system will automatically check the code for errors. Figure 11 shows the error detection of using a sample code of greatest common divisor. Users can readily know in which line of the code and what is the error.

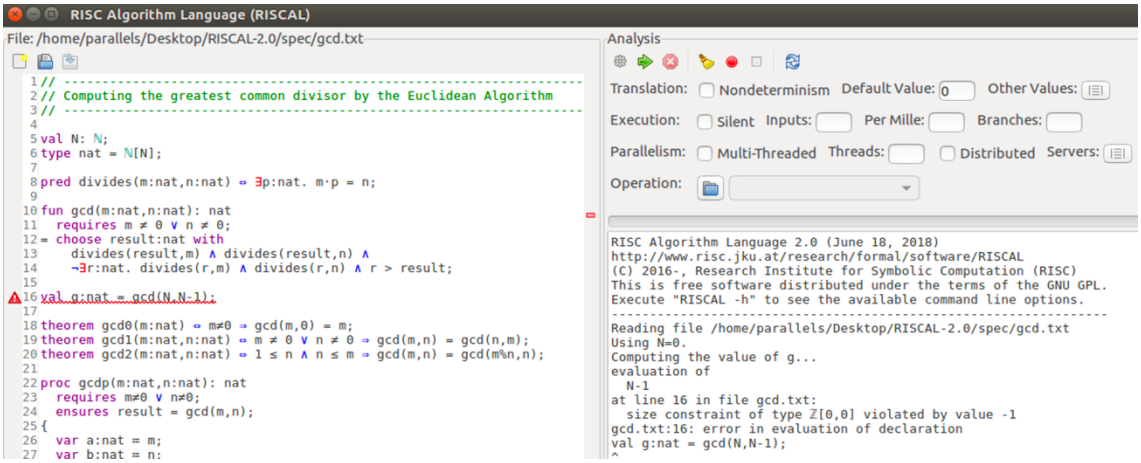


Figure 11: Error detection of RISCAL software

By using the logging function, the system will account every operation and error which users have done. Afterwards, users can find a recorded file under the root of RISCAL directory. If the system encountered a failure or fatal error while running, users are able to reset the system core manually. Furthermore, if the content of the output console is getting too long, users can use the “clear output” features to clear the panel.

The optional features which show in the Figure 12 are “Translation”, “Execution”, “Parallelism”, and “Operation.” In default, the specification is executed in determinism mode, “Default Value” is set to 0, and “Other Value” has no value being configured. Users can decide to select the “Nondeterminism” mode and define a default value in the specification in the “Default Value” field.

Figure 12: The optional features under the menu bar of RISCAL software

To give particular values of various constants, users can press the “Other Value” button to add/delete the name and the value of specific constants. Figure 13 displays the window of the “Other Value” with example values set up.

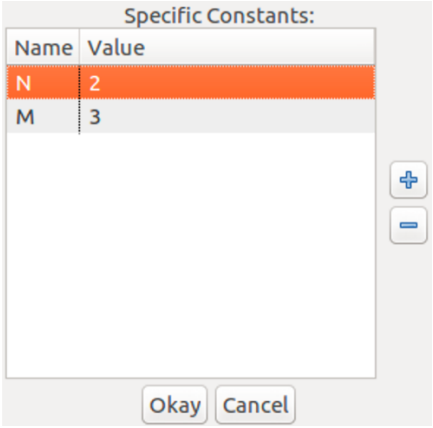


Figure 13: The “Other Value” window of RISCAL software

The “Silent” option gives a summarizing result instead of the explanation of every operation. By using the same sample code, the difference between the “Silent” option being selected is shown as in Figure 14.

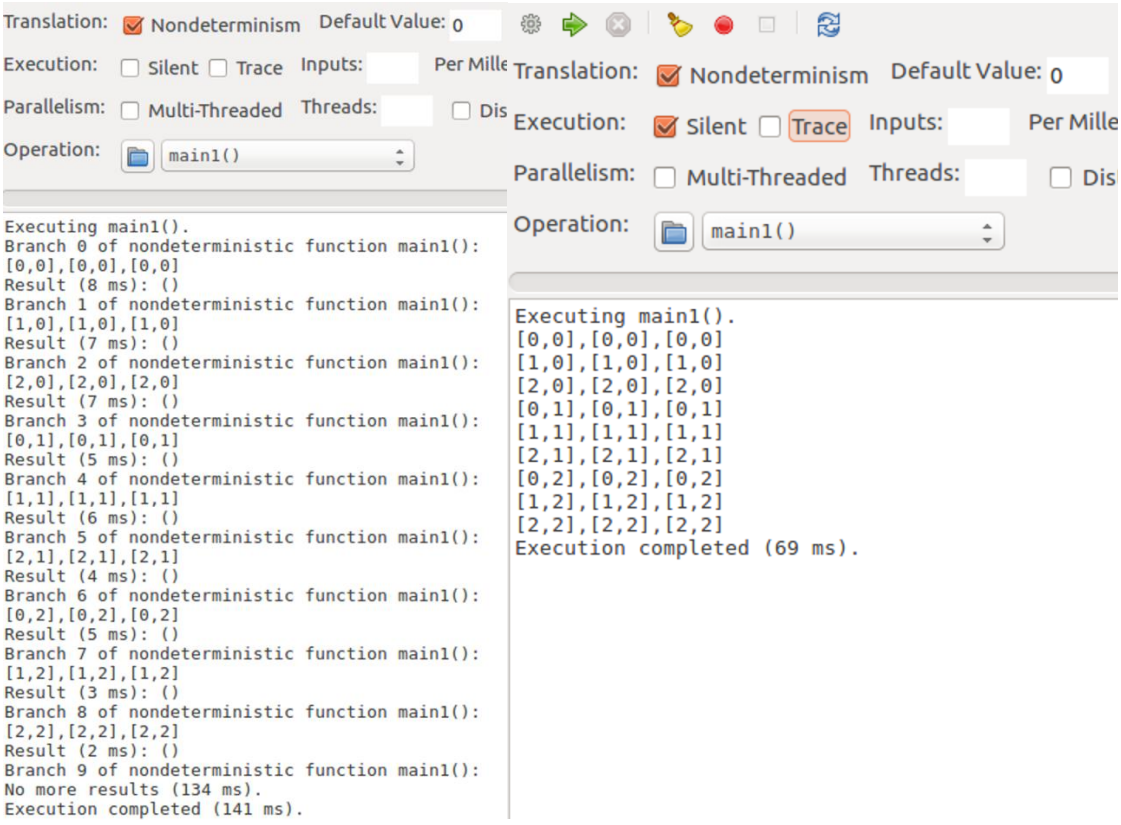


Figure 14: The difference of the “Silent” option being selected

The RISCAL software provides capabilities to verify the correctness of an operation. If users press the directory figure right beside the “Operations”, it will show/hide the task panel. The user interface will then be expended at the right side by a few tasks which are related to the currently selected operation. Figure 15 shows the user interface of the RISCAL software after extending the task panel. With the assistance of task panel, users can easily perform a validation/verification of the different operations.

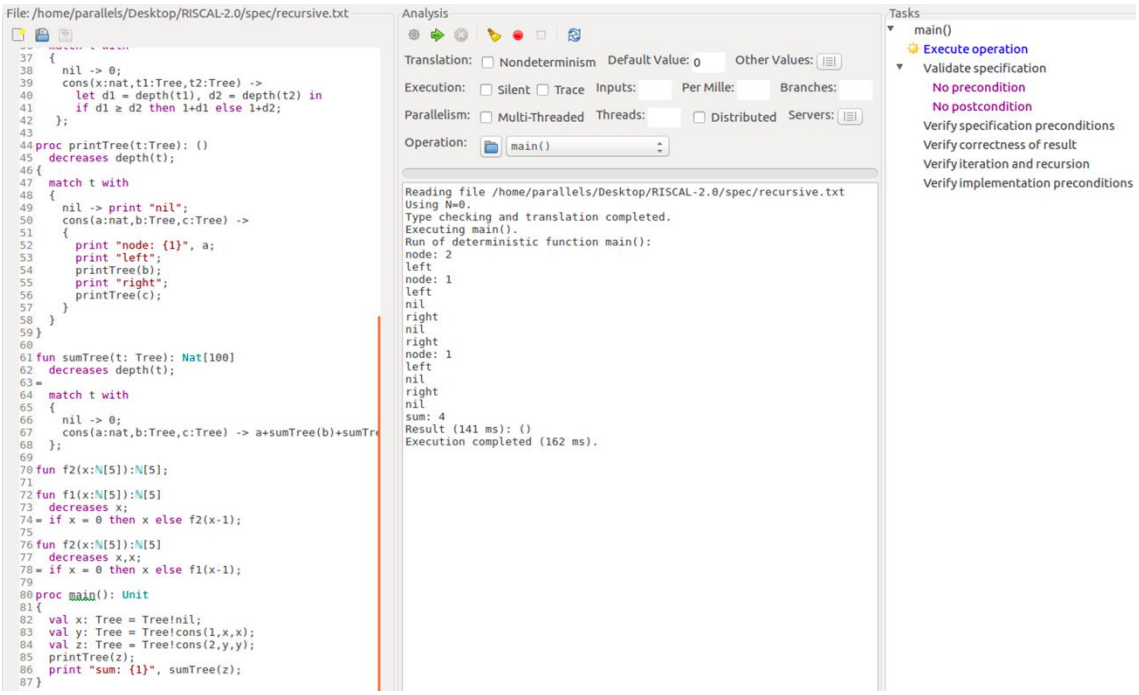


Figure 15: The user interface of RISCAL software with task panel

To activate the visualization features, the user has to click the “Trace” check box without the option “Nondeterminism”, “Multi-Threaded” and “Distributed” selected. After executing the operation, a window will pop up with the combination of nodes and arrows which represent the procedure states and the states change for each input. Figure 16 shows the visualizing execution trace by using the sample code of sample operation.

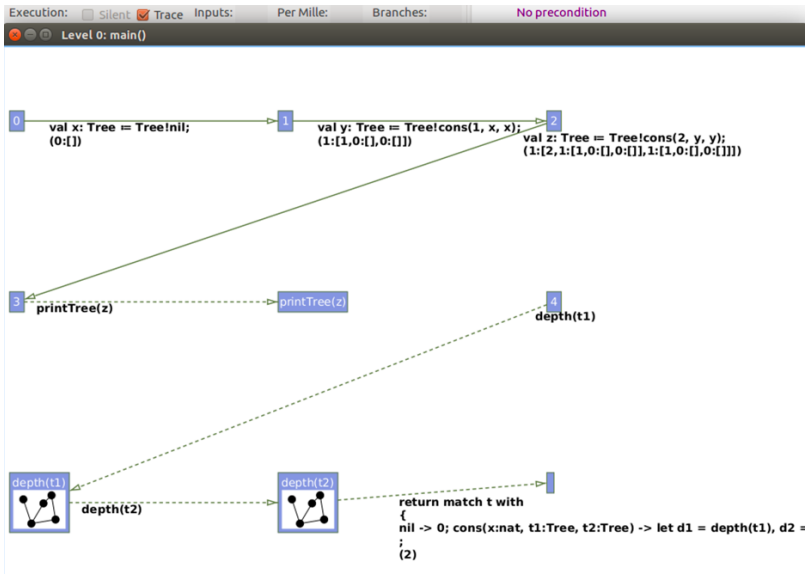


Figure 16: The visualization feature of RISCAL software

Usually, the code is executed in single thread. Nevertheless, the software offers multi-threading executions which users can assign how many threads to be used on the local machine. The parallelism is not limited on a single machine, but also can be used on distributed servers by appointing the socket address, port number, and password. Figure 17 shows the distributed execution function of the RISCAL software.

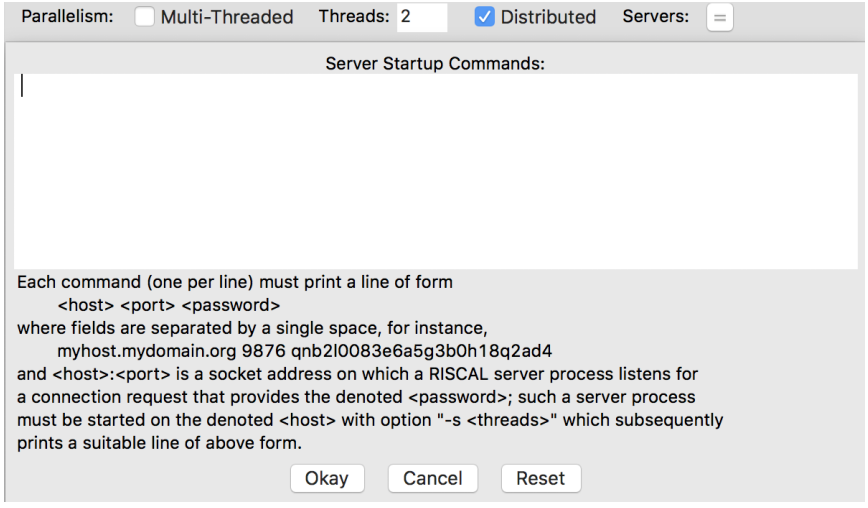


Figure 17: The parallel function of RISCAL software

This section has briefly described all the essential functionalities of the RISCAL software and presented in user interface, including all the additional pop-up windows. Moreover, it has given an overview of how the user interacts with the program by selecting the option boxes and checked boxes. Intentionally, the web version of the RISCAL software should also be provided with the features introduced above.

4. Evaluation of Web Frameworks

To evaluate aforementioned web frameworks, this section will elaborate various criteria, including how these frameworks are used for development, their integrity, their user interface, and their performance. The evaluation is experimentally validated by a simple test program, which provides the features of getting values from an input text box by pressing a button and displaying the result in an output text box.

For each criterion, we will describe and evaluate the following:

1. Installation & Portability
2. Support & Popularity
3. Development Process
4. Client-Side Features
    a. Rich Text Editor Panel
    b. Input & Output Panel
    c. Menus
    d. File Selection Dialog
    e. Selection Boxes
    f. Pop Up Panels and Windows
    g. Tree-structured Panels
    h. Built-in Browsing
5. Aesthetics & Usability
6. Mobility & Browser Support
7. Performance

There will be given a short mark for each section and a summation mark in the end for conclusion. (The marks are in the range 1-4 where 4 means the best and 1 means the worst.) The marks are used to determine which framework is most appropriate for the RISCAL software to be adopted.

4.1. Installation & Portability

For each framework, users have to download and install the Java SE Development Kit (JDK) from the Oracle website. RAP, JSP and GWT support JDK 8 or later. However, OpenXava only supports JDK 6-8. The different frameworks can be briefly installed by downloading the Eclipse IDE with specific plugins installed.

RAP requires the “Eclipse IDE for RCP and RAP Developers.” JSP, OpenXava, and GWT need the “Eclipse IDE for Java EE Developers.” RAP and GWT need additional plugins to operate properly. These plugins can be simply installed by the Eclipse IDE. JSP does not need any other plugins. Nevertheless, OpenXava cannot be installed by built-in plugins in Eclipse IDE. It has to download and import the plugin from the OpenXava website. Since Java is a cross-platform software, all the framework above can be installed, developed, and run on different operating systems. Table 1 gives a short summary of this criterion.

Table 1: Marks for Installation & Portability

<table>
<tr><th></th><th>IDE</th><th>SDK</th><th>Plugins</th><th>Marks (1-4)</th></tr>
<tr><td>RAP</td><td>Eclipse IDE for RCP and RAP Developers</td><td rowspan="2">8 or later</td><td>Built-in</td><td>3</td></tr>
<tr><td>JSP</td><td rowspan="3">Eclipse IDE for Java EE Developers</td><td>No</td><td>4</td></tr>
<tr><td>OpenXava</td><td>6-8</td><td>Additional</td><td>1</td></tr>
<tr><td>GWT</td><td>8 or later</td><td>Built-in</td><td>2</td></tr>
</table>

4.2. Support & Popularity

For a software developer, it is very important to have a good documentation in order to speed up the development process. The user can easily understand how to use the API to do the development. Therefore, it is necessary for the software provider to provide an official instruction/tutorial. Furthermore, the more materials the user can find for a software, the more popular the software is. It is more stable, if the regular updates of the software give more functionalities and fix bugs.

RAP, JSP, OpenXava, and GWT provide official online documentations and official forums. However, there are also books, videos and other third-party platforms that give instruction or guide for developers about above mentioned frameworks. All the documentations of each framework are constructed by either lessons or functionalities which is good for a developer to understand. The regular update time for each framework is quite similar; it is every 6-12 months for a big update and every few months for a small patch. The popularity can be easily distinguished by using search engine: the more results are found, the more popular the software is. The summary of this criteria is organized as Table 2.

Table 2: Marks for Support & Popularity

	Official Forums	Official Documentations	Regular Updates	Popularity	Marks (1-4)
RAP	V	V	Every 6-12 months	3	3
JSP	V	V		4	4
OpenXava	V	V		1	1
GWT	V	V		2	2

4.3. Development Process

To evaluate the development process of different frameworks, we have designed a Java testing program by using the SWT library. Figure 18 shows the user interface of the testing program. The source code of the testing program is shown in Figure 19. The development process for each framework will be introduced in the following subsections. We will tell about how to use the framework to write a simple program.

Figure 18: The testing program with Java SWT

```
public class Helloswt {
        Display display = new Display();
        Shell shell = new Shell(display);
        Label inputLabel, outputLabel;
        Text input1, output1;
        Button button;
        public Helloswt() {
                shell.setText("Hello SWT");
                shell.setLayout(new GridLayout(2, false));
                inputLabel = new Label(shell, SWT.NULL);
                inputLabel.setText("Input: ");
                input1 = new Text(shell, SWT.SINGLE | SWT.BORDER);
                outputLabel = new Label(shell, SWT.NULL);
                outputLabel.setText("Output: ");
                output1 = new Text(shell, SWT.SINGLE | SWT.BORDER);
                button = new Button(shell, SWT.PUSH);
                button.setText("Get Input !");
                Listener listener = new Listener() {
                        public void handleEvent(Event event) {
                                String data = input1.getText();
                                output1.setText(data);
                        }
                };
                button.addListener(SWT.Selection, listener);
                shell.pack();
                shell.open();
                while (!shell.isDisposed()) {
                        if (!display.readAndDispatch()) {
                                display.sleep();
                        }
                }
                display.dispose();
        }
        public static void main(String[] args) {
                new Helloswt();
        }
}
```

Figure 19: Source code of Java testing program with SWT

4.3.1. RAP Development

In RAP, the UI is designed by using the Java language with the SWT library. Most of the functions can be implemented with minor revisious. Figure 20 shows the source code of the UI design on RAP.

```
public class BasicEntryPoint extends AbstractEntryPoint {
        @Override
        protected void createContents(Composite parent) {
        parent.setLayout(new GridLayout(2, false));
        Button checkbox = new Button(parent, SWT.CHECK);
        checkbox.setText("Hello");
        Button button = new Button(parent, SWT.PUSH);
        button.setText("World");
        }
}
```

Figure 20: Source code of the UI design on RAP

While migrating the testing program from Java to RAP, users does not need to revise many lines of code. It is very simple to do the migration. Figure 21 shows the result after migrating. The source code of the test program after migration is shown in Figure 22.

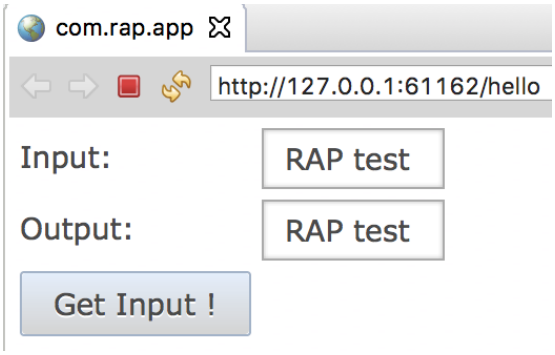


Figure 21: The testing program on RAP

```
public class BasicEntryPoint extends AbstractEntryPoint {
  private static final long serialVersionUID = 1L;
  @Override
  protected void createContents(Composite parent) {
            Label inputLabel, outputLabel;
            Text input1, output1;
            Button button;
            parent.setLayout(new GridLayout(2, false));
            inputLabel = new Label(parent, SWT.NULL);
            inputLabel.setText("Input: ");
            input1 = new Text(parent, SWT.SINGLE | SWT.BORDER);
            outputLabel = new Label(parent, SWT.NULL);
            outputLabel.setText("Output: ");
            output1 = new Text(parent, SWT.SINGLE | SWT.BORDER);
            button = new Button(parent, SWT.PUSH);
            button.setText("Get Input !");
            Listener listener = new Listener() {
                  private static final long serialVersionUID = 1L;
                  public void handleEvent(Event event) {
                  String data = input1.getText();
                  output1.setText(data);
               }
            };
            button.addListener(SWT.Selection, listener);
  }
```

Figure 22: Source code of the test program on RAP after migration

4.3.2. JSP Development

In JSP, the UI is designed by using Java, HTML, JavaScript and CSS. Users have to revise all the code about the UI design from Java to HTML. Figure 23 shows the source code of the UI design on JSP.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Hello World JSP</title>
        <input type="checkbox" /> Hello
        <button>World</button>
    </head>
    <body>
    </body>
</html>
```

Figure 23: Source code of the UI design on JSP

Since the SWT library is used for designing desktop software, JSP is hard to implement. Therefore, the simplest way to migrate from Java to JSP is to redesign the user interface in HTML and to use the existed Java program to handle the main function. Figure 24 shows the testing program based on the HTML function. The source code of the test program after rewriting in JSP is shown on the next page.

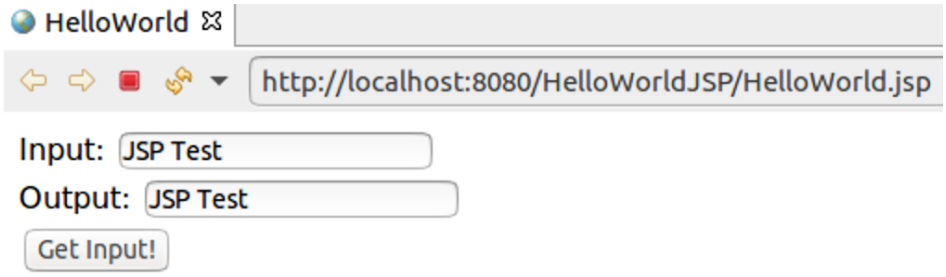


Figure 24: The testing program on JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-
8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HelloWorld</title>
  </head>
  <body>
    <form action = "#" onsubmit="getInput();return false">
      Input: <input type = "text" name = "input" id = "input">
      <br />
      Output: <input type = "text" name = "output" id = "output">
      <br />
      <input type="submit" value="Get Input!"/>
    </form>
    <script>
      function getInput(){
        document.getElementById("output").value
        = document.getElementById("input").value;
      }
    </script>
  </body>
</html>
```

Figure 25: Source code of the test program after rewriting in JSP

4.3.3. OpenXava Development

The design of a UI on OpenXava is based on JSP, but OpenXava is more like a modularization tool for business purposes. Users have to understand many modules inside the OpenXava project to get used to it. Furthermore, as mentioned in section 4.3.2, JSP is hard to implement SWT. To migrate from a Java program to a modulize program will be a huge progress. Users have to rewrite all the function into MVC-archtecture when using OpenXava. Therefore, we retain from presenting a corresponding test program in this subsection.

4.3.4. GWT Development

The UI design of GWT is based on Java, HTML and CSS. Figure 26 shows the source code of a UI design on GWT.

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Web Application Starter Project</title>
    <script type="text/javascript" language="javascript"
src="helloworld/helloworld.nocache.js"></script>
  </head>
  <body>
    <h1>Web Application Starter Project</h1>
    <table align="center">
      <tr>
        <td id="nameFieldContainer"></td>
        <td id="sendButtonContainer"></td>
      </tr>
    </table>
  </body>
</html>
```

Figure 26: Source code of a UI design on GWT

As mentioned in section 4.3.2, the SWT library can not be used here. A simple way to migrate from Java to GWT is to rewrite the code for the UI design and to use the original Java function. Figure 27 shows the testing program based on the GWT function. The source code of the test program after rewriting in SWT is shown in Figure 28.

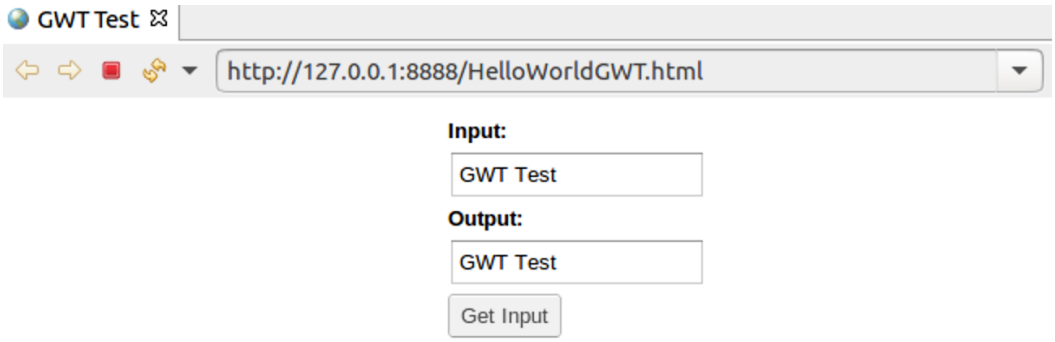


Figure 27: The testing program on GWT

```
<!doctype html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>GWT Test</title>
    <script type="text/javascript" language="javascript"
src="helloworldgwt/helloworldgwt.nocache.js"></script>
  </head>
  <body>
    <table align="center">
      <tr>
        <td colspan="2" style="font-weight:bold;">Input:</td>
      </tr>
      <tr>
        <td id="inputFieldContainer"></td>
      </tr>
      <tr>
        <td colspan="2" style="font-weight:bold;">Output:</td>
      </tr>
      <tr>
        <td id="outputFieldContainer"></td>
      </tr>
      <tr>
        <td id="sendButtonContainer"></td>
      </tr>
    </table>
  </body>
</html>
```

Figure 28: Source code of the test program after rewriting in SWT

4.3.5. Summary

The summary of all criteria is shown in Table 3.

Table 3: Marks for Development Process

	Marks (1-4)
RAP	4
JSP	3
OpenXava	1
GWT	2

Since OpenXava is a modularization product, it is difficult to implement a relevant test program. Therefore, we will not consider it in the following evaluation but only the other remaining frameworks.

4.4. Client-Side Features

To get to know if a framework satisfies the various elements of the user interface of RISCAL, this section will describe all the client-side features which the frameworks support. The client-side features of RISCAL software can be mainly divided into 8 classes.

4.4.1. Rich Text Editor Panel

RAP does not support rich text editor panel since it has not been implemented in the current version. However, RAP supports Incubator add-ons. The rich text widget on RAP can be installed via a third-party custom widget. JSP and GWT can easily be implemented by using HTML and JavaScript, or Java. The marks of this section are described in the Table 4.

Table 4: Marks for Rich Text Editor Panel

	Marks (1-4)
RAP	3
JSP	4
GWT	4

4.4.2. Input & Output Panel

The input and output panel is available in RAP, JSP and GWT. RAP uses SWT/RWT to present the panel. JSP and GWT are using HTML and JavaScript, or Java classes to construct the panel. The panel can be built by using the default function in each framework. The marks of this section are described in the Table 5.

Table 5: Marks for Output Panel

	Marks (1-4)
RAP	4
JSP	4
GWT	4

4.4.3. Menus

RAP, JSP and GWT support the menu function. RAP uses SWT/RWT to build the menus. JSP uses HTML and JavaScript, but it is not included in the default functionality. The developers have to develop it by themselves. GWT uses Java classes to display menus. The marks of this section are described in Table 6.

Table 6: Marks for Menus

	Marks (1-4)
RAP	4
JSP	2
GWT	4

4.4.4. File Selection Dialog

Unlike a client-side application, it is impossible for a remote program to access a user file on the local system. To implement a similar function for file selection is to upload the file to the server for a web program. RAP and GWT provide their own tool kits to create the feature. JSP is using HTML and JSP script to do the same stuff. The marks of this section are summarized in Table 7.

Table 7: Marks for File Selection Dialog

	Marks (1-4)
RAP	1
JSP	1
GWT	1

4.4.5. Selection Boxes

RAP, JSP and GWT support the selection boxes feature. RAP uses SWT/RWT to display the selection boxes. JSP uses HTML and JavaScript to implement this feature. GWT uses Java classes to create the functionality. The selection boxes can be built by using the default function in each framework. The marks of this section are described in Table 8.

Table 8: Marks for Selection Boxes

	Marks (1-4)
RAP	4
JSP	4
GWT	4

4.4.6. Pop Up Panels and Windows

RAP, JSP and GWT support pop-up panels and the windows feature. RAP uses SWT/RWT to create the pop-up panels and windows. JSP uses HTML and JavaScript to display them. GWT uses Java classes to implement this functionality. The marks of this section are described in Table 9.

Table 9: Marks for Pop Up Panels and Windows

	Marks (1-4)
RAP	4
JSP	4
GWT	4

4.4.7. Tree-structured Panels

RAP, JSP and GWT support the tree function. RAP uses SWT/RWT to build hierarchical tree panels. JSP uses HTML and JavaScript, but it is not included in the default functionality. This functionality needs to be developed by the developers or by using third-party plugins. GWT uses Java classes to display panels. The marks of this section are given in Table 10.

Table 10: Marks for Tree-structured Panels

	Marks (1-4)
RAP	4
JSP	2
GWT	4

4.4.8. Built-in Browsing

Only RAP supports built-in browsing feature. RAP uses SWT/RWT to create a built-in browser. JSP and GWT have to either open in a new tab or in a same tab to browse the URL. The marks of this section are described in Table 11.

Table 11: Marks for Built-in Browsing

	Marks (1-4)
RAP	4
JSP	1
GWT	1

4.5. Aesthetic & Usability

For a web developer, it is necessary to provide an attractive user interface to the users. Usability is also a key point to interact with the users. It can be considered by setting up the UI and defining the events handler. However, the aesthetics is subjective for each individual. It is very difficult to satisfy the demands for all people simultaneously.

RAP, JSP, and GWT provide basic widgets which can be associated with CSS. The developers can easily modify each element to define how the appearance is. There are some guidelines to follow in order to generate a positive impression, e.g. try to keep everything organized and easy to read, avoid the use of too many colors, minimize the scrolling and graphics should be meaningful. With these simple guidelines, we can be sure to create a concise and usable site.

Aesthetics and usability are differently judged by different people. Technically speaking, the web domain name for RAP does not have any file extension which is more clear and beautiful compared with other frameworks. An example of the web domain of RAP is http://[IP_ADDRESS:PORT/example]; however, other frameworks might ended with example.jsp or example.html. In personal judgment, the default widgets without any CSS on RAP are more impressive. GWT comes next, and the last one will be JSP. The summary of this criteria is shown as Table 12.

Table 12: Marks for Aesthetics & Usability

	Marks (1-4)
RAP	4
JSP	2
GWT	3

4.6. Mobility & Browser Support

With the increasing of tablets and smartphones, it is important for these frameworks to support either iOS or Android devices. However, above mentioned frameworks are all web-based architectures. The frameworks can be used on most of the browsers nowadays, such as Safari, Google Chrome, Firefox and Microsoft Edge (with slight bugs). Additional add-ons are necessary to target the mobile platforms.

In RAP, it is possible to offer solutions for modern mobile devices without additional plugins, but there are some limitations or issues for different platforms, for instance, the touching-scrolling function may not work significantly. GWT needs the MGWT plugin to develop mobile platforms. JSP is based on HTML, CSS, and JavaScript, which means it does not need any add-ons to be operated on mobile devices.

To sum up, these frameworks need either additional add-ons or UI pages to target the mobile platforms. As a matter of fact, all of them can be applied perfectly from the major browsers on PC. The marks of this criteria for each framework are described as Table 13.

Table 13: Marks for Mobility & Browser Support

	Marks (1-4)
RAP	3
JSP	4
GWT	2

4.7. Performance

This section aims to evaluate the performance of the simple testing program on different frameworks. The environment of the experiments is executed in the local area network. There are three important points to be evaluated, page size, page request and page speed. When the user opens a website, the browser will get all the necessary files from the server, including HTML, CSS, and JavaScript files. The page request is about how many documents the browser download. The page size is the summation of all the request files. The page speed is how long does it take to get the files. Moreover, the testing tool is using the built-in web developer tools on Firefox. The value of performance for each framework is organized in Table 14.

Table 14: Marks for Performance

	Page Size	Page Request	Page Speed	Marks (1-4)
RAP	2205.82 KB	6	55 ms	3
JSP	0.825 KB	1	17 ms	4
GWT	2240 KB	9	113 ms	2

From the table above, there is no doubt that JSP has the best performance. Although RAP and GWT have a similar page size, the page request and page speed for GWT are more than for AP. The reason that RAP and GWT have a large page size and a large number of page request is because both of them are using additional web toolkits. However, JSP is using simple HTML and JavaScript.

4.8. Summary

To conclude, the marks for each subsection are summarized in Table 15. The table directly extracts the values from the table in the previous sections. Except for Section 4.4, it makes an average mark from all the subsections. Both RAP and JSP have made a remarkable performance. Although the summation of both frameworks is quite similar, the marks on each criterion for RAP is higher than 3. JSP seems to be not working pretty well on the client-side features and aesthetics & usability for the RISCAL software. Therefore, RAP will be the best candidate compared with other web frameworks to be selected for migrating the current software to the web.

Table 15: Marks Summary

	4.1	4.2	4.3	4.4	4.5	4.6	4.7	Sum
RAP	3	3	4	3.5	4	3	3	23.5
JSP	4	4	3	2.75	2	4	4	23.75
GWT	2	2	2	3.25	3	2	2	16.25

5. Porting RISCAL to the Web

According to the results from Chapter 4., RAP is the best candidate to port the RISCAL software to the web. While doing the migration, some functions or packages which the original software is using may not be operating successfully. In this situation, we have to find other substitute packages or reduce some optional functions in order to make the core functions work properly. In this chapter, we will present what is the difference after porting the RISCAL software to the web, what are the fundamental changes, which function is working or not and the system architecture compared to the original one.

5.1. System Architecture

The RISCAL software is originally designed to be used on a local machine. Users must download the software from the Internet and install Java and all the necessary packages by themselves. If one of the requirements is not fulfilled, the software may lead to non-operating. On a user's perspective, this may bring down the desire of using a software. To reduce the conditions of installation, it is required to construct a web platform. In other words, users can access the software by launching an URL on a web browser. Figure 29 shows the system architecture of the original RISCAL software. Figure 30 shows the system architecture of the RISCAL software after porting to the web.

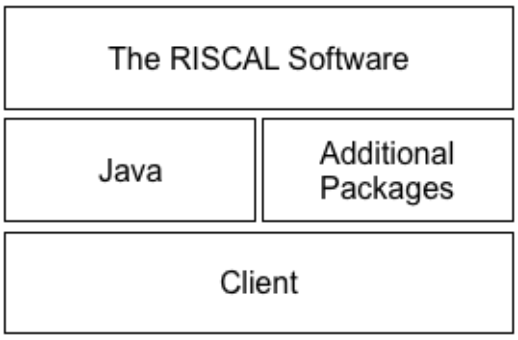


Figure 29: The system architecture of the original RISCAL software

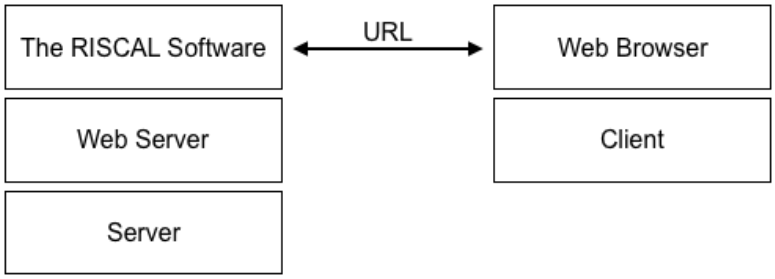


Figure 30: The system architecture of the RISCAL software after porting to the web

5.2. File I/O

From RCP to RAP, most of the SWT libraries are compatible to be used in the RAP environment. While porting the software to the web, there are some features that have to be replaced by the RWT libraries. One of the most fundamental feature has to be changed is the basic I/O function. The reason is that any web services is restricted to access the user's file system and read data directly. If this becomes possible, which means that the web services are able to access the user's file system simply while the user is browsing any web sites, it might become an enormously disaster.

The RISCAL software provides an input field for users to key in the mathematical theories/algorithms. This feature is known as the basic I/O function. It is also allowed to open or save a document file on the local machine. As mentioned above, this feature is not able to be implemented on the web service. Therefore, the only option is to replace this function with file upload and download service. With this service, users can upload the specification file from the client side to the server side, revise the document on the web and download the file back to the client. The execution/validation on the correctness of the theories/algorithms are all done on the server side. Figure 31 shows the scenario of the I/O features after porting the RISCAL software to the web.

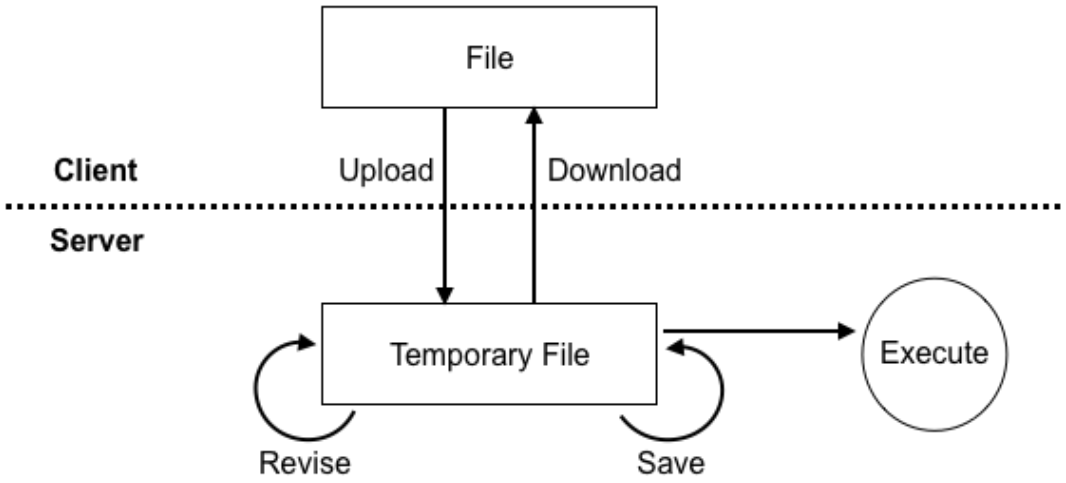


Figure 31: The scenario of the I/O features

The source code of the file upload and download service shows in Figure 32. The function “startUploadReceiver()” handles the file upload event. Once the upload progress is successful, it will open the file and present the document in the input field. The “sendDownload” function will register a download service to create a URL link and send the link to the user by using URL launcher.

```
public boolean sendDownload(byte[] data, String filename) {
        DownloadService service = new DownloadService(data, filename);
        service.register();
        UrlLauncher launcher = RWT.getClient().getService(UrlLauncher.class);
        launcher.openURL(service.getURL());
        return true;
}

private String startUploadReceiver() {
        DiskFileUploadReceiver receiver = new DiskFileUploadReceiver();
        FileUploadHandler uploadHandler = new FileUploadHandler( receiver );
        uploadHandler.addUploadListener( new FileUploadListener() {
        @Override
        public void uploadProgress( FileUploadEvent event ) {
                // handle upload progress
        }
        @Override
        public void uploadFailed( FileUploadEvent event ) {
                Main.getOutput().println( "upload failed: " + event.getException() );
        }
        @Override
        public void uploadFinished( FileUploadEvent event ) {
                for( FileDetails file : event.getFileDetails() ) {
                        Main.getOutput().println( "received: " + file.getFileName() + "\n");
                        Main.file = receiver.getTargetFiles()[ fileSequence ];
                        openFile(Main.file);
                        fileSequence += 1;
                }
        }
        } );
        return uploadHandler.getUploadUrl();
}
```

Figure 32: Source code of the file upload and download service

5.3. User Interface

The basic I/O features might be the most important changes when migrating from RCP to RAP. Furthermore, we did not revise anything in the core function which does the mathematical theories/algorithms execution and validation but focus more on revising the UI parts of the code. While doing the migration, some of the functions do not occur any errors in the Eclipse IDE, but the errors then come at the runtimes. Other errors illustrate that the required packages are missing or not supported. This will be the next problem to be resolved.

Other Value

In “Other Value” function, the original RISCAL software was implemented by using the “TableCursor” package. This package allows to create a table element for users to add/remove dynamic values within the table. There are two columns in one row inside the table. The users can add/remove one row in one operation. However, this package has not been implemented in RAP yet. To make this function be operational, the substitute package “TableEditor” has been adopted. This package works no difference with the previous one, but it occurs one problem which is that to change from one column in the table to another column by using mouse click is not available. To fix this problem, the substitute solution is to use “Tab” to replace the mouse click. The revised code of this part is shown in Figure 33. The revised code allows users to switch between two columns by using “Tab”.

```
final TableEditor editor = new TableEditor(table);
final int EDITABLECOLUMN = 0;
final int EDITABLECOLUMN2 = 1;
table.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e) {
        Control oldEditor = editor.getEditor();
        if (oldEditor != null) oldEditor.dispose();
        TableItem item = (TableItem)e.item;
        if (item == null) return;
        Text newEditor = new Text(table, SWT.NONE);
        newEditor.setText(item.getText(EDITABLECOLUMN));
        newEditor.addModifyListener(new ModifyListener() {
                public void modifyText(ModifyEvent e) {
                        Text text = (Text)editor.getEditor();
                        editor.getItem().setText(EDITABLECOLUMN, text.getText());}});
        newEditor.addFocusListener(new FocusAdapter() {
                public void focusLost(FocusEvent e){
                        if (shell.isDisposed()) return;
                        newEditor.setText(newEditor.getText());
                        newEditor.dispose();}});
        newEditor.selectAll(); newEditor.setFocus();
        editor.setEditor(newEditor, item, EDITABLECOLUMN);
        newEditor.addTraverseListener(new TraverseListener(){
                public void keyTraversed(TraverseEvent e) {
                        if (e.detail == SWT.TRAVERSE_TAB_NEXT) {
                        newEditor.selectAll(); newEditor.setFocus();
                editor.setEditor(newEditor, item, EDITABLECOLUMN);
                        Text newEditor2 = new Text(table, SWT.NONE);
                newEditor2.setText(item.getText(EDITABLECOLUMN2));
                newEditor2.addModifyListener(new ModifyListener() {
                        public void modifyText(ModifyEvent e) {
                                Text text = (Text)editor.getEditor();
                                editor.getItem().setText(EDITABLECOLUMN2, text.getText());}});
                newEditor2.addFocusListener(new FocusAdapter() {
                        public void focusLost(FocusEvent e){
                                if (shell.isDisposed()) return;
                                newEditor2.setText(newEditor2.getText());
                                newEditor2.dispose();}});
            newEditor2.selectAll();  newEditor2.setFocus();
            editor.setEditor(newEditor2, item, EDITABLECOLUMN2);}}});}});
```

Figure 33: Source code of “Other Value” features after revising

Input Fields

In addition, to ensure all the control options, e.g. the checkbox of “Nondeterminism” is checked or not and the value changed in the “Inputs” field, are always synchronizing with the user’s behavior. The original RISCAL software is using background threads to implement this feature. However, there are limitations when using background threads on RAP. Most notably, when a UI state is changed by a background thread, the UI of the client would not be updated normally until the next time the user does something with the UI.

To fix the problem of synchronization, one of the solutions is to use the UI sessions to update the user interface. Nevertheless, it is not necessary to use this method to fix the problem due to the original design of the software is using a function “writePreferences” to record the states of the UI. Therefore, a simple method to fix the synchronization problem is to remove the background thread and call the function “writePreferences” whenever the UI state has been changed.

Confirmation Dialogs

While users want to close the program, the original RISCAL software will pop up a dialog shell and ask for an exit confirmation. Figure 34 shows the exit confirmation feature on the original RISCAL software. This function does not contain any errors while porting the software to the web, but this feature is not working. The reason is that the function is designed for closing a shell instead of a web page. To ask for an exit confirmation on a web site, RAP uses an exit confirmation service to implement this feature. By simply adding the code in Figure 35 in the main function of the UI, it will present the same effect as the original one.

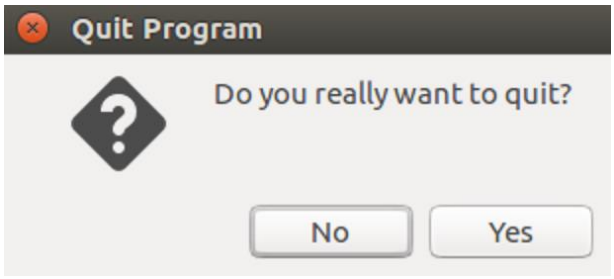


Figure 34: The exit confirmation dialog on the original RISCAL software

```
ExitConfirmation exitConfirmService = RWT.getClient().getService( ExitConfirmation.class );
exitConfirmService.setMessage( "Do you really wanna leave RISCAL?" );
```

Figure 35: Source code of exit confirmation dialog on RAP

5.4. Known Bugs and Unimplemented Features

According to the official document from the RAP website, there are some unimplemented widgets, e.g. StyledText, Tracker, Taskbar, and Tray. These unimplemented widgets are available in RCP, but not in RAP. Furthermore, the RSICAL software used “StyledText” widget to implement the error detection and undo manager functions for validating the correctness of the mathematical theories/algorithms. Thence, these functions are currently not available.

The editor panel is using the basic “Text” widgets without any additional features now as a substitute solution. The trace and visualization features are also not supported due to the shortage of required Java libraries. Additionally, the distribute and parallel features are removed because these two functions are not necessary while the program is already running on the server side.

The accelerator is known as the keyboard shortcut for interacting actions. This feature usually helps the user to shorten the time of finding the demanded items and increases the user experience. Although the accelerators have been set up in the program, there are some accelerators which are not working properly. This is often happened when setting up an accelerator which is already used by the browser. In our program, “Ctrl + n” has been set up for opening a new file. However, the browser opens a new blank page when testing this feature in the web. This is a known bug which is not able to be fixed.

Besides, there is one more known bug which is the output console will extrude the other panels when the length of the output is too long. The original design of using dynamic shell is to fit every screen size so that there will be no problem with different machines. This bug can be fixed by using a fixed shell size, but this might affect the using experience on different devices. In addition to the above-mentioned bugs and the features which are not able to be implemented, all the other functions work as good as the original RISCAL software.

5.5. Deployment on the Server

When all of the bugs and functions are fixed in the Eclipse IDE, then the program is ready to be ported to the web. While there are some configurations have to be set up before deploying the software on the web server. To be able to run a Java program on a web server, a web server which supports Java is required, such as Apache Tomcat or Eclipse Jetty. The component of a RAP program is not only the Java program itself, some files of configuration have to be defined. The configuration files describe the entry point of the web sites and which classes are loaded during the runtime. When creating a RAP project in Eclipse IDE, there are three folders, META-INF, OSGI-INF, and WEB-ING are generated by default including four configuration files, MANIFEST.MF, contribution.xml, launch.ini and web.xml within the corresponding folders. One property files, build.properties which are related to the MANIFEST.MF file is also automatically created in the root path of the project. Figure 36 shows the layout of the components inside a RAP program.

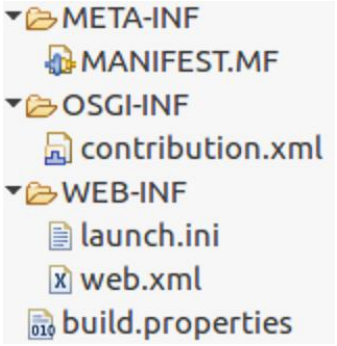


Figure 36: The components of a RAP program

The functionality of each component in a RAP program is described as follows:

- META-INF/MANIFEST.MF: defines the required plugins, packages to import, runtime classes path, execution environments, and extensions.
- OSGI-INF/contribution.xml: specifies the provided services and the implementation classes in the servlet.
- WEB-INF/launch.ini: describes the properties which are loaded to start the framework.
- WEB-INF/web.xml: indicates the servlet description and the configuration of the parameters.
- build.properties: defines the libraries to be built and the source folders which should be compiled. This file is connected to the MANIFEST.MF file.

During the development, the contribution.xml, launch.ini and web.xml files preserve the default configuration since these files are related to the components of the servlet framework. Therefore, MANIFEST.MF will be the core file to focus on. The additional Jar packages which the developer is using have to be specified in the bundle class path; otherwise, the program will occur errors during the runtime. The version of require bundle and import package should be conscious. The official document also mentioned that the “javax.servlet” and “javax.servlet.http” should be included in the imported package rather than in the required bundle . The detail of the MANIFEST.MF file which the program is using is shown in Figure 37.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Riscal
Bundle-SymbolicName: com.riscal
Bundle-Version: 1.0.0.qualifier
Automatic-Module-Name: com.riscal
Bundle-RequiredExecutionEnvironment: JavaSE-1.8
Require-Bundle: org.eclipse.rap.rwt;bundle-version="[3.0.0,4.0.0)"
Service-Component: OSGI-INF/contribution.xml
Import-Package: javax.servlet;version="[2.3.0,4.0.0)",
 javax.servlet.http;version="[2.3.0,4.0.0)",
 org.eclipse.rap.fileupload
Bundle-ActivationPolicy: lazy
Bundle-ClassPath: lib/antlr4.jar,
 lib/org.eclipse.rap.filedialog-3.6.0.jar,
 lib/icons.jar,
 .
```

Figure 37: The configuration of the MANIFEST.MF file

After all the configuration have been set up, the program is ready to be exported as a WAR file, which is the specific format for a Java web server to recognize. This action can be done by using the war tool plugin in the Eclipse IDE. For deploying the program on the web server, the developer should place the exported WAR file in the corresponding path of the web server. The detailed guides for developers and users can be found in Appendix.

6. Use of the System and Evaluation

To evaluate the program after porting the RISCAL software to the web, this chapter is aimed to demonstrate all the functionalities of the system. The new UI of the system which is shown in Figure 38 is roughly as same as the original software, only some of the features which are not able to implement have been removed. The detailed features will be described in the subsection.

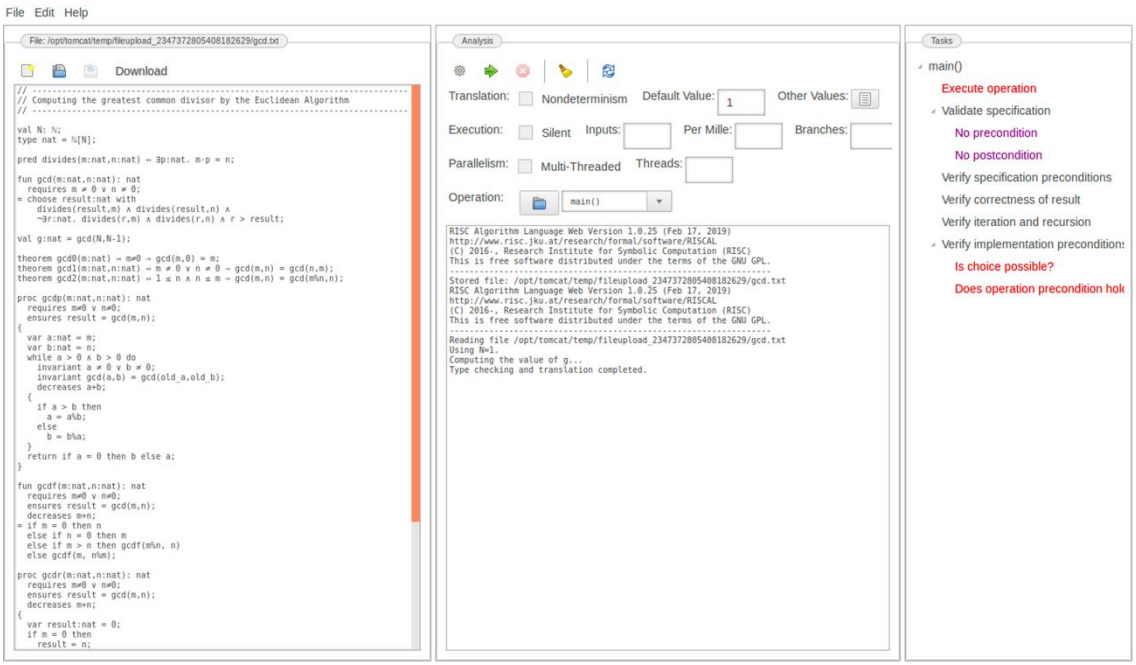


Figure 38: The new UI of the RISCAL software

6.1. Menu Bar

The menu bar is located on the top of the UI, including three menu items, which are File, Edit, and Help. Figure 39, Figure 40 and Figure 41 show the contents inside the corresponding menu items.

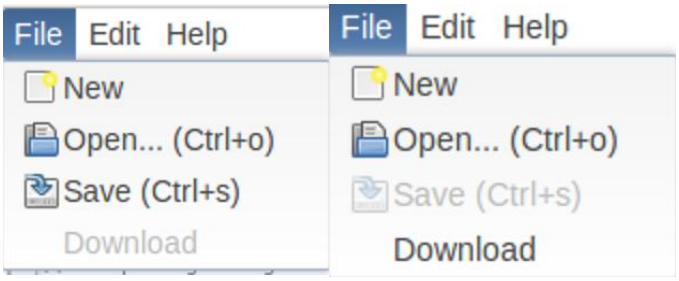


Figure 39: The contents inside the File menu

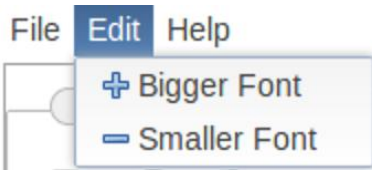


Figure 40: The contents inside the Edit menu

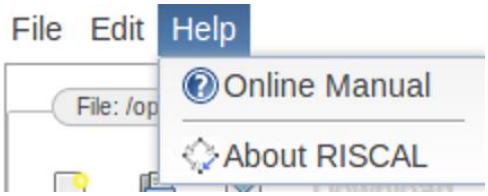


Figure 41: The contents inside the Help menu

The functions inside the File menu is used for handling the I/O features. The user is able to open a new file by pressing the “New” item. However, the shortcut “Ctrl + n” has been removed due to it is duplicated with the default setting of the web browser. A file upload dialog which is shown in Figure 42 will pop up while pressing the “Open” item. The user can select the file in their local machine and upload to the server as a temp file by using this feature. The “Open” function can also be accessed by entering the shortcut “Ctrl + o”. For each file which is successful uploaded, the program will execute the main function to check if the mathematical specification is legal or not.

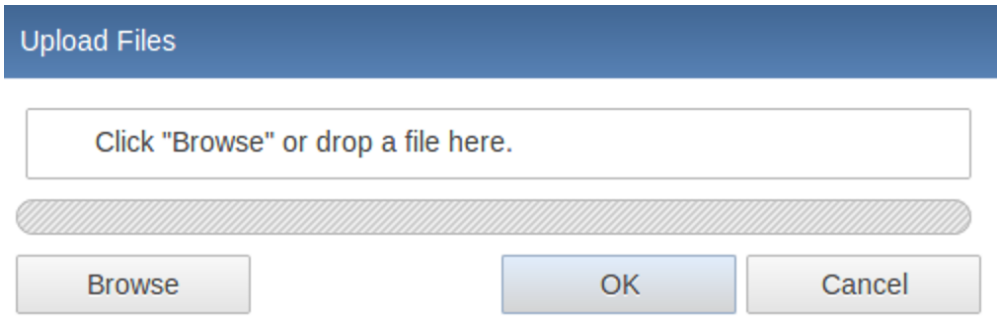


Figure 42: The file upload dialog

The “Save” item is used to save any changes of the file after uploading to the server. Moreover, the program is able to detect if the user does any change on the uploaded file. Once the user does not revise anything on the file, this function will remain unselected and vice versa. The “Download” item allows users to get the

current document from the server. This function is selectable only when the file has been saved and vice versa. Figure 43 shows an example of the download function.

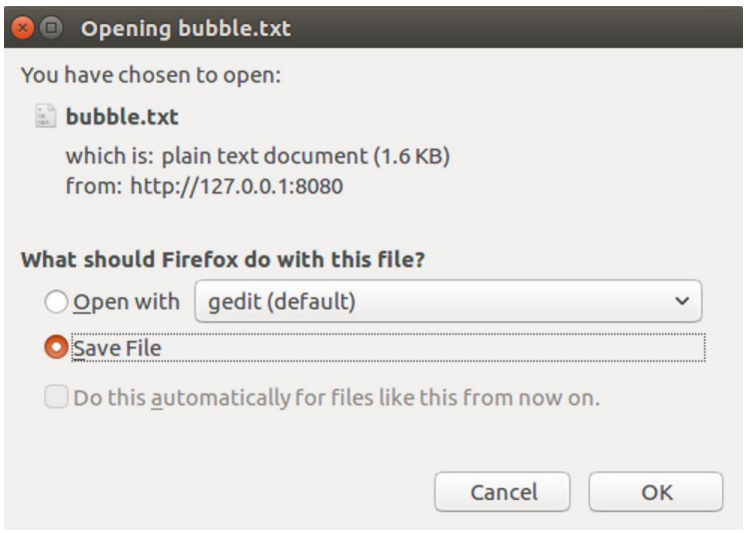


Figure 43: An example of the download function

Besides, if the current state of the file has not been saved when the user presses the “New” or the “Open” function, the program will pop up a dialog which will ask the user to save the file. This feature will force the user to save the modified file before doing any next action. Figure 44 shows the dialog of file not saved.

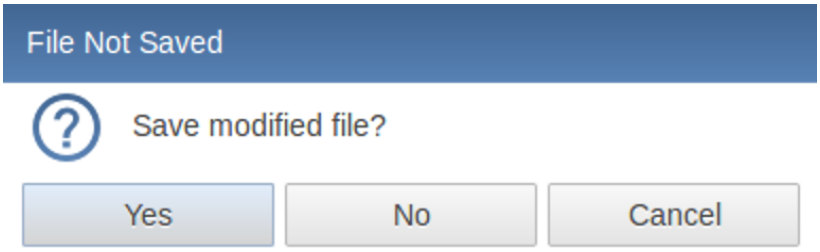


Figure 44: The dialog of file not saved

As for the functionalities inside the “Edit” menu, the Undo and Redo features have been removed because it is currently not able to be implemented. However, the default shortcut “Ctrl + z” from web browser is still available for one-time undo feature. “Bigger Font” and “Smaller Font” will influence the font size of the editor panel and the output console. The accelerator shortcuts for both of them are not available due to it is duplicated with the default setting of web browser.

By pressing the “Online Manual” inside the “Help” menu, the program will present the PDF document of the RISCAL software within a pop-up window. This feature is shown in Figure 45. Furthermore, the “About RISCAL” item will also show up a pop-up window which described the copyright of the software and the author information in HTML format. This function is shown in Figure 46.

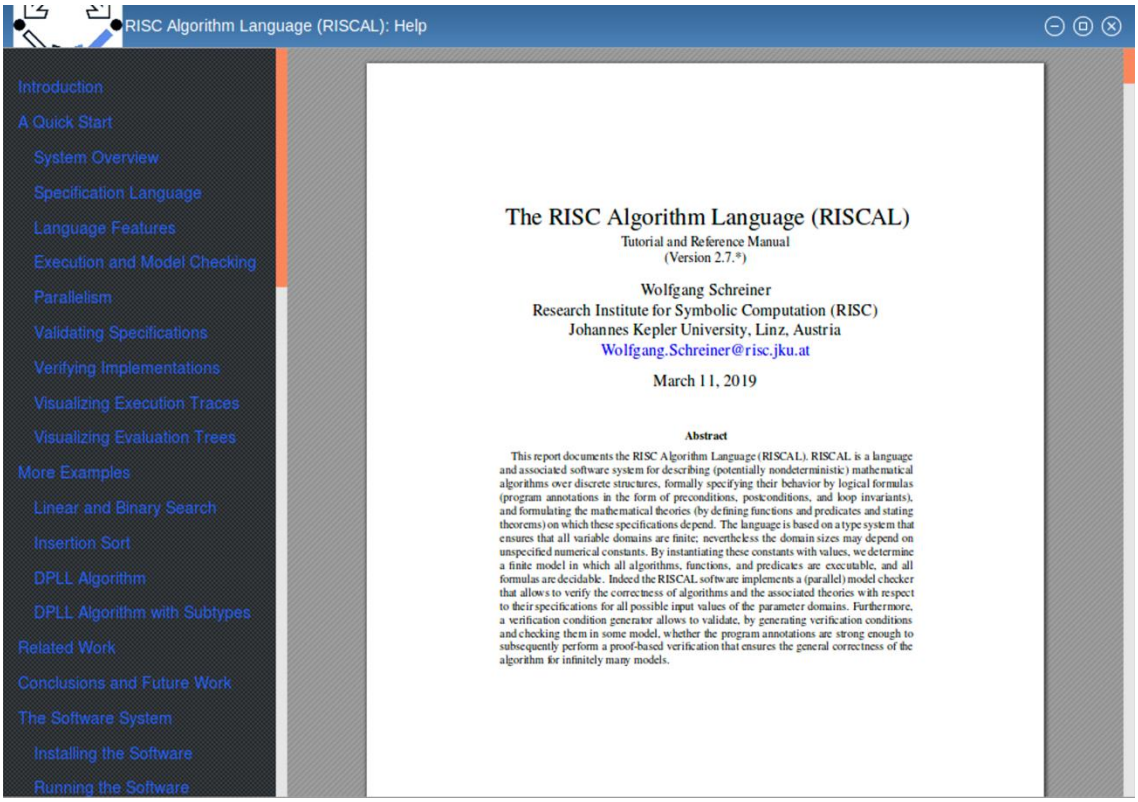


Figure 45: The feature of the “Online Manual”

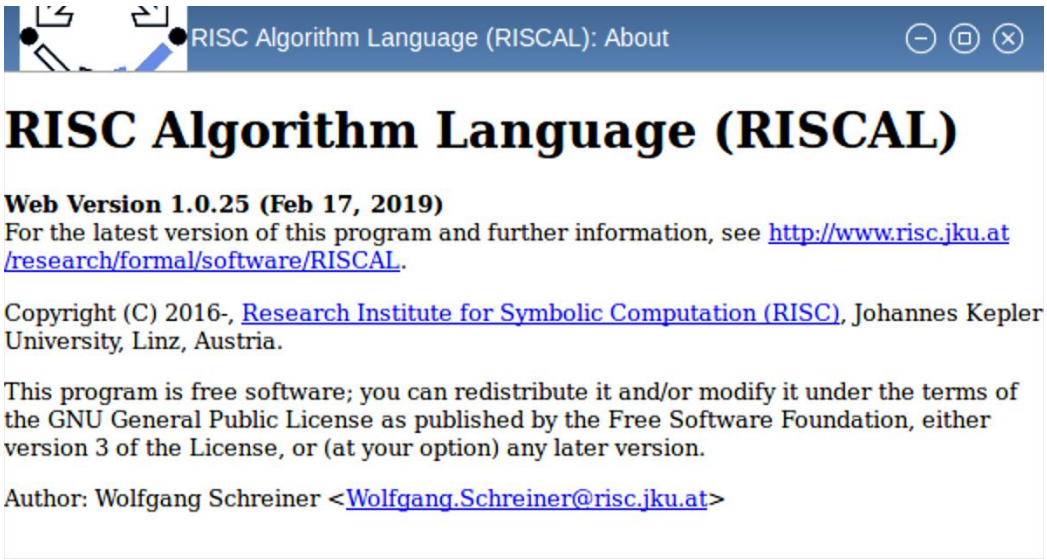


Figure 46: The feature of the “About RISCAL”

6.2. Editor Panel

The editor panel is located on the left-hand side of the main UI. Figure 47 shows the detail of the editor panel. From top to down are the current file path, the buttons of I/O functions and the input field. When the current file is a new file, it will show “File: (Untitled)” in the file path. Once the file has been successful uploaded to the server, it will show the absolute location of the file on the server in the file path.

The icons of the I/O functions from the left to the right represent “New”, “Open”, “Save” and “Download” features. The functions of the I/O buttons inside the editor panel are as same as those inside the menu bar. Since the “Download” function is a new feature in the program, there is currently no icon for it.

The last element inside the editor panel is the input field which allows the user to do the mathematical programming. The functionalities of the error detections and undo/redo managers are not available in the current version due to the widget of “StyledText” is unimplemented in the current RAP API. However, the program will indicate the semantic and grammatical errors after the execution and present these errors on the console field. Therefore, these features have been removed, only left with the basic input feature.

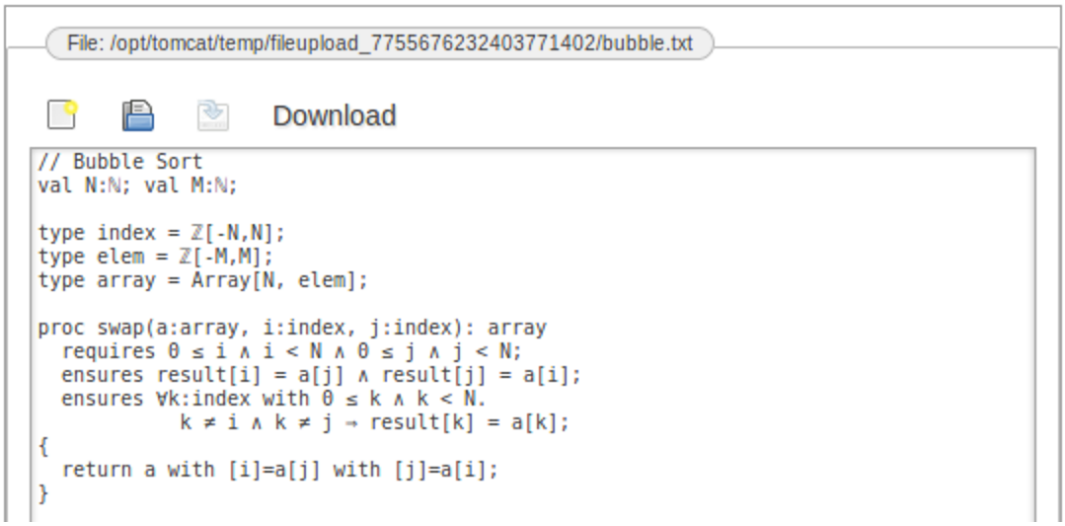


Figure 47: The editor panel

6.3. Analysis Panel

On the middle of the main UI is the analysis panel. The icons represent the core functions, including “Process Specification”, “Start Execution”, “Stop Execution”, “Clear Output” and “Reset System”. Other executing options and the output console are sequentially ordered from top to down. In this aspect, all the features work as good as the original one except the “Logging”, “Trace”, and “Distributed” functions which have been removed by the reason of not able to be implemented or unnecessary.

By clicking any checkbox options or giving values for any input boxes, this action will be synchronized to the server in real time. This is guaranteed that any changes on the UI will always be updated. In addition, the default setting of the executing options is that any checkbox is unselected, the value for “Default Value” is 0. Moreover, the default values of “Inputs”, “Per Mille”, “Branches” and “Threads” are null. Figure 48 shows the detail of the analysis panel.

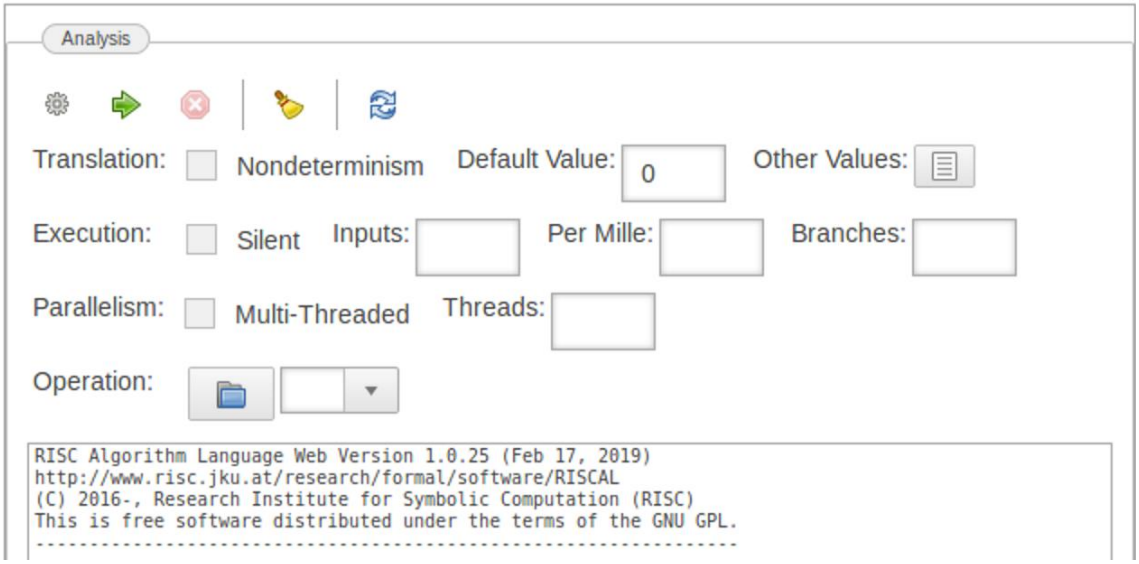


Figure 48: The analysis panel

A pop-up window will show up after pressing the button of “Other Values”. This allows users to add/remove a row of specific constants by pressing “+” or “-” button. The users must use the “TAB” on the keyboard to switch from different columns between “Name” and “Value”. The program will automatically check if the given input in each column is valid or not after pressing the “Okay” button. If the given value is legal, the specific constants will be saved. Furthermore, any change on this dialog will remain unchanged if the user presses the “Cancel” button. Figure 49 shows the dialog of “Other Value”.

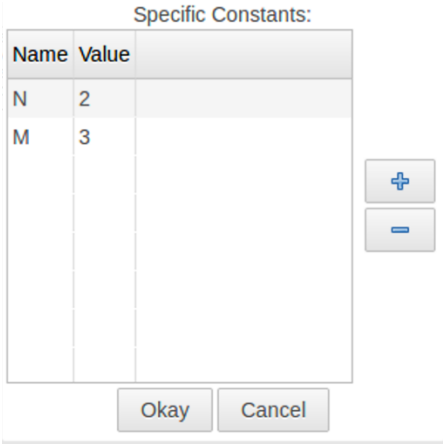


Figure 49: The other value dialog

The tasks panel can be displayed/hided by clicking the “Operation” button. The drop-down option next to the “Operation” button will load the defined processes from the specification file, which allows the user to select different operations to be executed or verified. Figure 50 shows the tasks panel after being displayed.

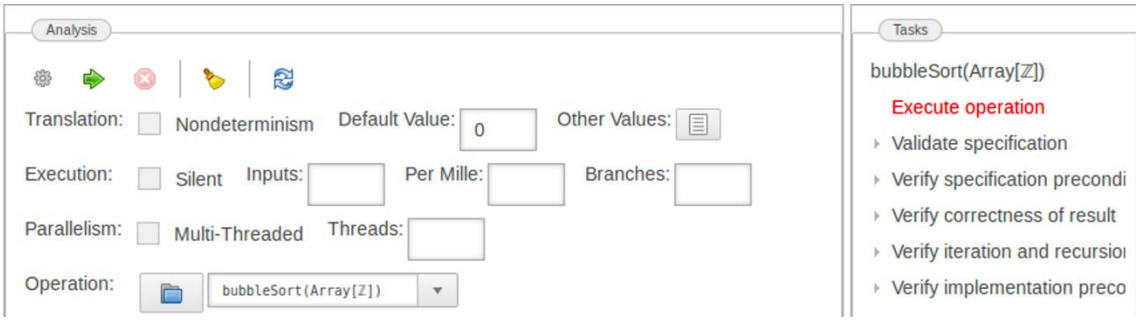


Figure 50: The tasks panel

6.4. Other Services

Once the user leaves the web page, the data will not be saved. To prevent the user from accidently closing the web page, an exit confirmation dialog will show up to remind the user. If the user chooses to leave, the program will be reset and reverted to the initial state. The new exit confirmation dialog is shown in Figure 51.

Figure 51: The new exit confirmation dialog on RAP

For user's interaction control, the default of a single session is 30 minutes. The server will respond a session timeout dialog if the user has been idled for more than the specific time. The duration of session can be defined in the web application manager. Figure 52 shows the session expired dialog and Figure 53 shows the web application manager on the Tomcat server.

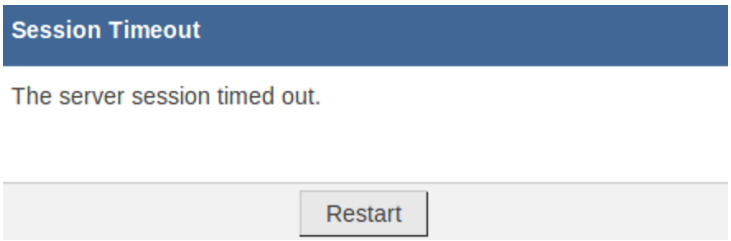


Figure 52: The session expired dialog

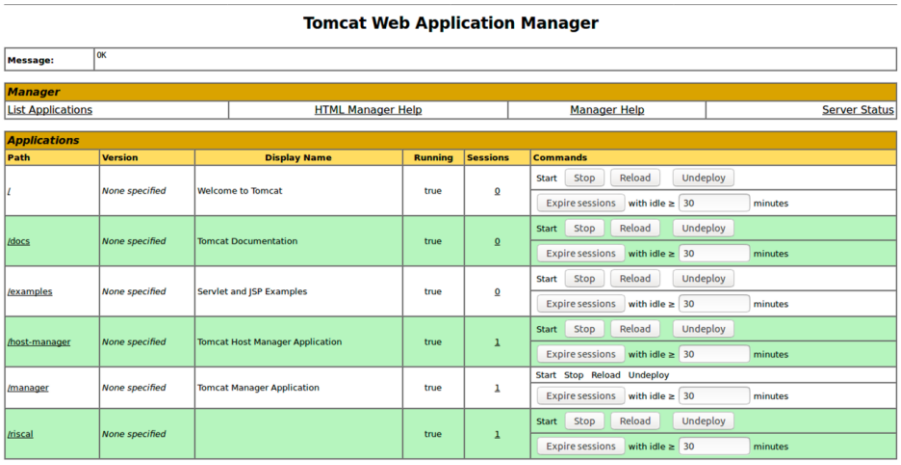


Figure 53: The web application manger on Tomcat server

7. Conclusion

The necessarily of using a program on the web is currently the main requirement to fit the demands of people. A software as a service not only brings the convenience of using the technology, but also solves problems which are common in life. Our research has identified four successful use cases of mathematical software which have been originally developed for the desktop but have successfully been ported to the web. Although these programs are not using the same developing tools or programming languages, these still bring us inspiration.

To be able to migrate an existed Java program to the web, our research has demonstrated four possible solutions. Understanding how these solutions are working and what are their advantages and disadvantages, gives developers a suggestion for their own developments. The evaluation of different frameworks has given individual marks for various criteria. Although the total marks of JSP seems to be very close to RAP, RAP has been chosen as the candidate because of its transferability and compatibility.

While doing the porting process, our research has documented in detail the difficulties which developers may encounter. Some packages are not available in the current developing environment which will result in reduced functionality or using alternative plugins instead. However, the core features of the RISCAL program have not been significantly influenced by these disadvantages. The differences between the original RISCAL software and the one after porting to the web have been well demonstrated.

To conclude, the RISCAL software has been successful migrated to the web, which allows users to access the program via any web browser. Research related to RCP/RAP development seems to be rare in the current situation. Therefore, contribution of our research gives developers a reference when porting Java applications to the web.

Bibliography

[1] Malik, M. I., Wani, S. H., & Rashid, A. (2018). CLOUD COMPUTING-TECHNOLOGIES. *International Journal of Advanced Research in Computer Science, 9*(2), 379.

[2] Schreiner, W. (2017). The RISC Algorithm Language (RISCAL). Retrieved December 16, 2017, from https://www.risc.jku.at/research/formal/software/RISCAL/

[3] Schreiner W. (2018) Validating Mathematical Theorems and Algorithms with RISCAL. *In: Rabe F., Farmer W., Passmore G., Youssef A. (eds) Intelligent Computer Mathematics*. CICM 2018. Lecture Notes in Computer Science, vol 11006, 248-254. Springer, Cham.

[4] Schreiner, W. (2017). *The RISC Algorithm Language (RISCAL)—Tutorial and Reference Manual (Version 1.0).* Technical report, RISC, Johannes Kepler University, Linz, Austria.

[5] Eclipse. SWT: The Standard Widget Toolkit. Retrieved March 29, 2019, from https://www.eclipse.org/swt/

[6] Blewitt, D. A. (2016). *Eclipse plug-in development beginners guide* - (2nd ed.). Packt Publishing Ltd.

[7] Wolfram Research. Mathematica. Retrieved March 21, 2018, from https://www.wolfram.com/mathematica/

[8] Wolfram Research. Wolfram Programming Lab (Open Cloud). Retrieved March 29, 2019, from https://lab.open.wolframcloud.com/app/

[9] Abell, M. L., & Braselton, J. P. (2017). *Mathematica by example* (5th ed.). London: Academic Press, an imprint of Elsevier.

[10] MathWorks. MATLAB. Retrieved March 21, 2018, from https://www.mathworks.com/products/matlab.html

[11] MathWorks. MATLAB Online. Retrieved March 29, 2019, from https://www.mathworks.com/products/matlab-online.html

[12] Higham, D. J., & Higham, N. J. (2016). *MATLAB guide* (3rd ed., Vol. 150). Philadelphia: Siam.

[13] RStudio. RStudio. Retrieved March 21, 2018, from https://www.rstudio.com/

[14] RStudio. RStudio Cloud. Retrieved March 29, 2019, from https://rstudio.cloud/

[15] Team, R. (2016). RStudio: integrated development for R. (RStudio, Inc., Boston, MA, USA).

[16] Maplesoft. Maple. Retrieved March 21, 2018, from https://www.maplesoft.com/

[17] Maplesoft. MapleCloud. Retrieved March 29, 2019, from https://maple.cloud/#group=MathApps

[18] Bernardin, L., Chin, P., DeMarco, P., Geddes, K. O., Hare, D. E. G., Heal, K. M., et al. (2011). *Maple programming guide*. Waterloo: Maplesoft.

[19] Hohenwarter, M. GeoGebra. Retrieved March 21, 2018, from https://www.geogebra.org/

[20] Chaudhari, L., Dr. Virtual Physics lab. Retrieved March 29, 2019, from https://www.geogebra.org/m/MsPMXgCs

[21] Hall, J., & Lingefjärd, T. (2016). *Mathematical modeling: Applications with GeoGebra*. Hoboken, NJ: John Wiley & Sons.

[22] Dean, J. (2018). *Web programming with HTML5, CSS, and JavaScript*. Burlington, MA: Jones & Bartlett Learning.

[23] Esposito, D. (2018). *Programming ASP. NET Core, Programming ASP. NET Core*. Microsoft Press.

[24] Django. Django. Retrieved March 29, 2019, from https://www.djangoproject.com/

[25] Otwell, T. Laravel. Retrieved March 29, 2019, from https://laravel.com/

[26] Eclipse. Remote Application Platform (RAP). Retrieved December 16, 2017, from http://www.eclipse.org/rap/

[27] Oracle. JavaServer Pages Technology. Retrieved January 10, 2018, from http://www.oracle.com/technetwork/java/javaee/jsp/index.html

[28] Ubelhor, L., & Hur, C. (2017). *Developing business applications for the web: With HTML, CSS, JSP, PHP, ASP.NET, and JavaScript*. Boise, ID: MC Press LLC.

[29] OpenXava. OpenXava. Retrieved January 10, 2018, from http://www.openxava.org/

[30] OpenXava. OpenXava philosophy_en. Retrieved May 24, 2018 from https://openxava.wikispaces.com/philosophy_

[31] Cabot, J. (2014, November 09). OpenXava: Lightweight Model-Driven Framework. Retrieved from https://modeling-languages.com/openxava-lightweight-model-driven-framework/

[32] Google. Google Web Toolkit. Retrieved January 10, 2018, from http://www.gwtproject.org/overview.html

[33] Tacy, A., Hanson, R., Essington, J., & Tökke, A. (2013). *GWT in action*. Shelter Island: Manning.

[34] GeoGebra. GeoGebra Platforms. Retrieved May 24, 2018 from https://dev.geogebra.org/trac/wiki/Platforms

Appendix

There are two parts in this appendix, one guides the developer to modify the program, another guides the user to set up and use the program.

A. Developer Guide

To develop or modify the RISCAL program, the developer will need to install the following program on Ubuntu 16.04 LTS.

- Java 8
- Eclipse IDE
- The web version of the RISCAL software and the Libra WAR tools

a. Java Installation

i. Open Terminal in Ubuntu.
ii. Enter command “sudo apt update”.
iii. Enter command “sudo apt –y install default-jre”.
iv. Check the currently Java version by entering command “java -version”
v. Setting the JAVA_HOME environment variable.

1. Check the Java installation path by entering command “sudo update-alternatives --config java”.

```
parallels@parallels-vm:~$ sudo update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr
/lib/jvm/java-8-openjdk-amd64/jre/bin/java
Nothing to configure.
```

2. Copy the installation paths. Then open /etc/environment file using vim or other text editor by entering command “sudo vim /etc/environment”.
3. Add “JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java” to the /etc/environment file and save it.

```
JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/u
sr/local/games"
```

4. Reload /etc/environment file to apply changes to the current session by entering command “source /etc/environment”.
5. Now Java has been installed.

b. Eclipse IDE Installation

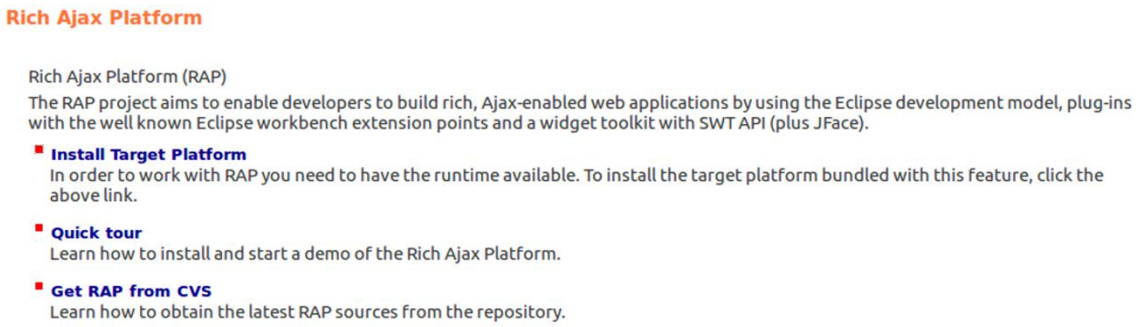
i. Download “Eclipse IDE for RCP and RAP Developers” from https://www.eclipse.org/downloads/packages/.

ii. Extract the compressed file.

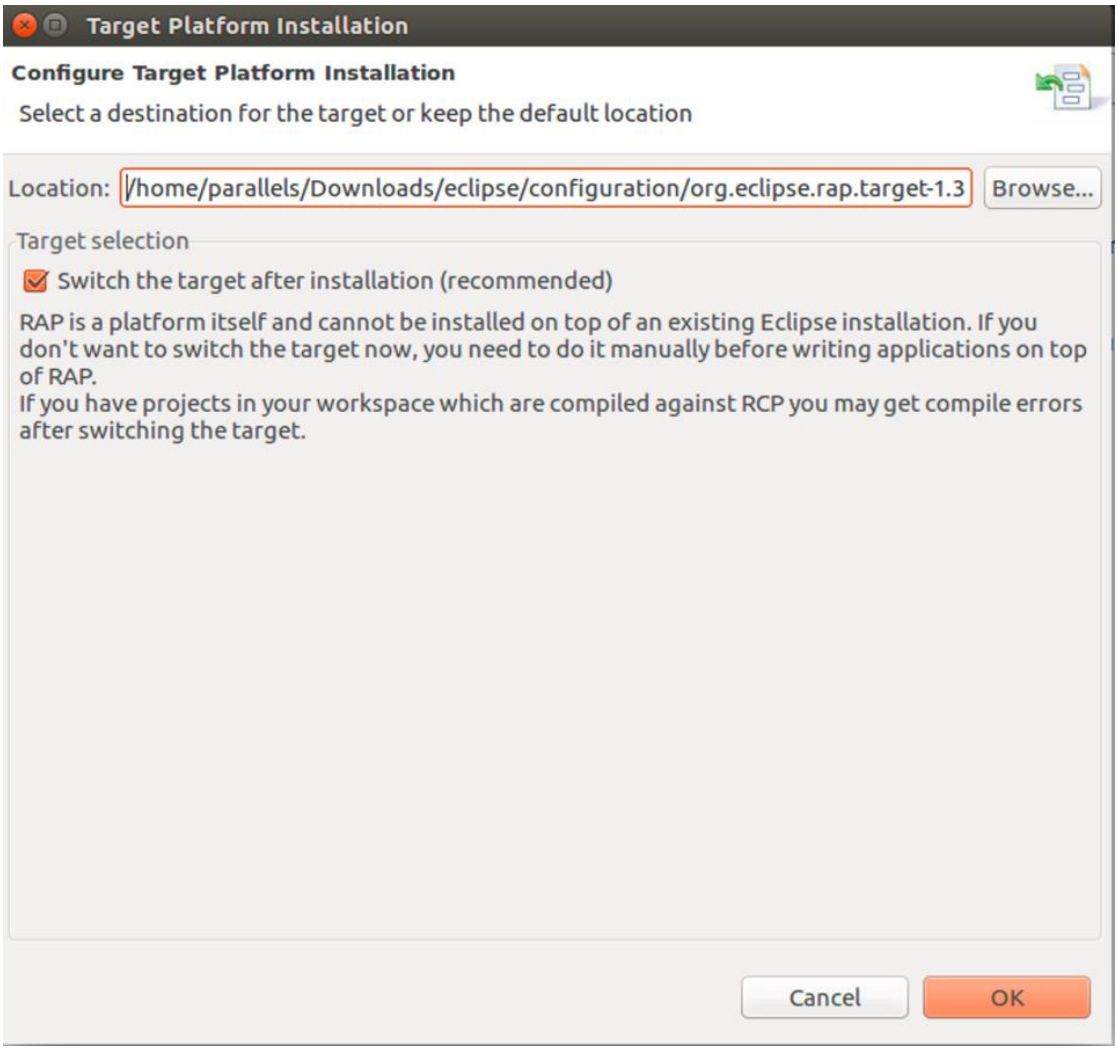
iii. Execute “eclipse” in the folder by double click it.

iv. After the program is launched, press “Overview” and find “Rich Ajax Platform (RAP)”.

v. Press “Rich Ajax Platform (RAP)” and then press “Install Target Platform”.



vi. Follow the installation and press “Ok”.



vii. Wait for the installation to be completed and restart the Eclipse IDE.

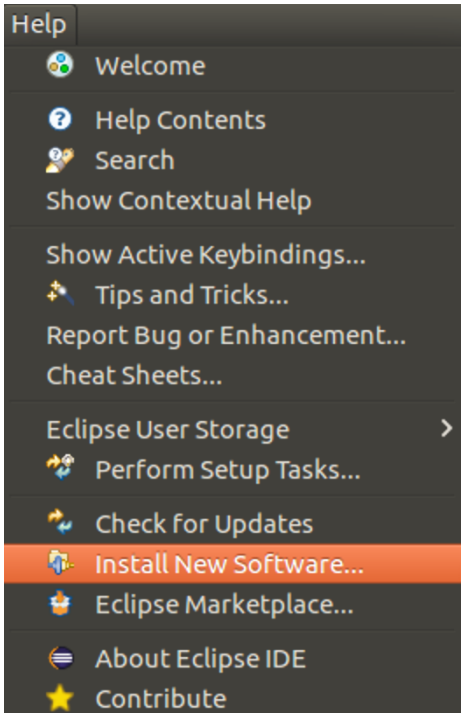
c. RISCAL Program and Libra WAR Tools Installation

i. Download the RISCAL program from GitHub https://github.com/chming1016/RISCAL_Web.

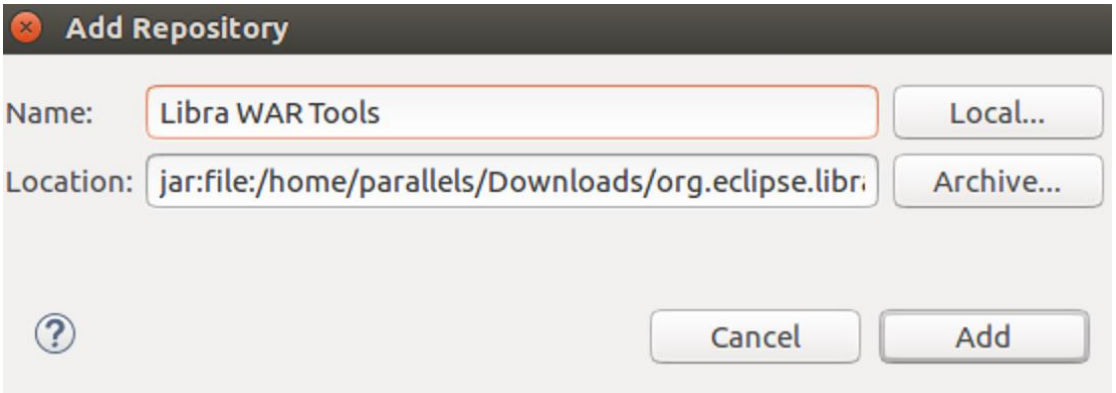
ii. Import the program in the Eclipse IDE.

iii. Download the latest Libra WAR tools from https://jenkins.eclipse.org/libra/job/Libra%20WAR/.

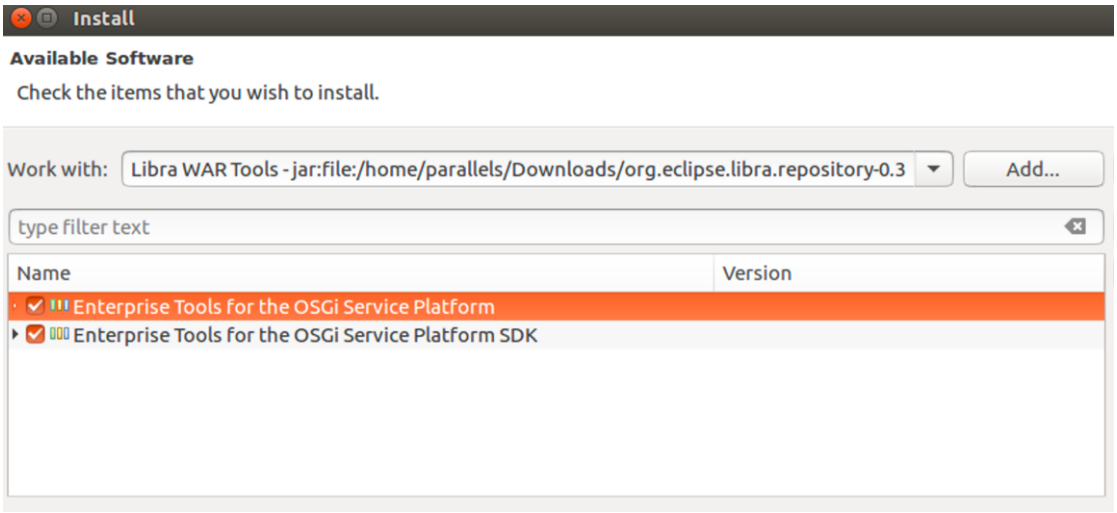
iv. Install the Libra WAR tools by clicking “Help” -> “Install New Software”.



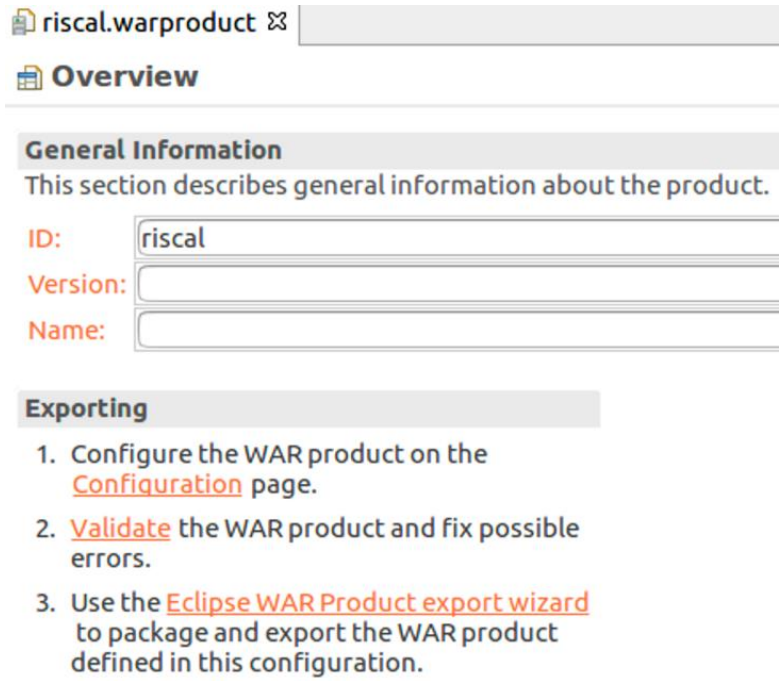
v. Click “Archive” and select the downloaded Libra WAR tools file, then click “Add”.



vi. Select all the Enterprise Tools, then click “Finish”.



vii. Now all the requirement is fulfilled. To use the Libra WAR tools, simply click “rascal.warproduct”. Follow the exporting processes to create a war file.



d. Virtual Box and Parallels Desktop Images

In case the Eclipse version changes or the API have been removed, the pre-installed program can be used with virtual machine. The developers can either use Virtual Box or Parallels Desktop to develop. Following is the detail information.

Download Link:
https://drive.google.com/drive/folders/1WzSdHXXO2VADPldBuIxwdxBqYF1o2qSk?usp=sharing

User Account: parallels
User Password: riscal2019

Eclipse IDE is located in /Home/parallels/Download/eclipse
The web version of the RISCAL software is located in /Home/parallels/Desktop/com.riscal

B. User Guide

To user the web version of the RISCAL program, users can either deploy the program on their web server or launch a pre-installed virtual machine image. The instruction uses Tomcat on Ubuntu 16.04 as an example, users can also use other web servers which support Java, e.g. Eclipse Jetty.

a. Web Server Installation

i. Install Java. See Appendix A.

ii. Open Terminal and create a Tomcat group and a user by entering commands “sudo groupadd tomcat” and “sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat”.

```
parallels@parallels-vm:~$ sudo groupadd tomcat
parallels@parallels-vm:~$ sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

iii. Find the latest version on the official Apache Tomcat http://tomcat.apache.org/download-80.cgi. Download the core binary distribution with tar.gz file type.

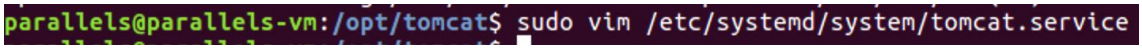
iv. Create the /opt/tomcat directory and extract the downloaded archive to it by entering commands “sudo mkdir /opt/tomcat” and “sudo tar xzvf apache-tomcat-8*tar.gz -C /opt/tomcat --strip-components=1”.

```
parallels@parallels-vm:/tmp$ sudo mkdir /opt/tomcat
parallels@parallels-vm:/tmp$ sudo tar xzvf apache-tomcat-8*tar.gz
-C /opt/tomcat --strip-components=1
```

v. Update permissions for the user tomcat by entering the following commands in sequence. “cd /opt/tomcat”, “sudo chgrp -R tomcat /opt/tomcat”, “sudo chmod -R g+r conf”, “sudo chmod g+x conf” and “sudo chown -R tomcat webapps/ work/ temp/ logs/”.

```
parallels@parallels-vm:/tmp$ cd /opt/tomcat/
parallels@parallels-vm:/opt/tomcat$ sudo chgrp -R tomcat /opt/tomcat
parallels@parallels-vm:/opt/tomcat$ sudo chmod -R g+r conf
parallels@parallels-vm:/opt/tomcat$ sudo chmod g+x conf
parallels@parallels-vm:/opt/tomcat$ sudo chown -R tomcat webapps/ work/ temp/ logs/
```

vi. Create a systemd service file to run Tomcat as a service by enter the command “sudo vim /etc/systemd/system/tomcat.service”. Copy and paste the following configuration to the file. Make sure the JAVA_HOME path is the same as the one which you installed.



/etc/systemd/system/tomcat.service

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64/jre
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

vii. Reload the systemd dameon by entering the command “sudo systemctl daemon-reload” and start the Tomcat server by entering the command “sudo systemctl start tomcat”.

```
parallels@parallels-vm:/opt/tomcat$ sudo systemctl daemon-reload
parallels@parallels-vm:/opt/tomcat$ sudo systemctl start tomcat
```

viii. Adjust the firewall and enable Tomcat to start at boot by entering the following commands, “sudo ufw allow 8080” and “sudo systemctl enable tomcat”.

```
parallels@parallels-vm:/opt/tomcat$ sudo ufw allow 8080
Rules updated
Rules updated (v6)
parallels@parallels-vm:/opt/tomcat$ sudo systemctl enable tomcat
Created symlink from /etc/systemd/system/multi-user.target.wants/tomcat.service
to /etc/systemd/system/tomcat.service.
```

ix. Configure Tomcat web management interface by editing the /opt/tomcat/conf/tomcat-users.xml file. The admin account is defined in this file by putting “<user username="admin" password="password" roles="manager-gui,admin-gui"/>” between the <tomcat-users…> and </tomcat-users>.

```
parallels@parallels-vm:/opt/tomcat$ sudo vim /opt/tomcat/conf/tomcat-users.xml
```

```
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary. It is
  strongly recommended that you do NOT use one of the users in the commented out
  section below since they are intended for use with the examples web
  application.
-->
<!--
  NOTE:  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<user username="admin" password="riscal2019" roles="manager-gui,admin-gui"/>
</tomcat-users>
```

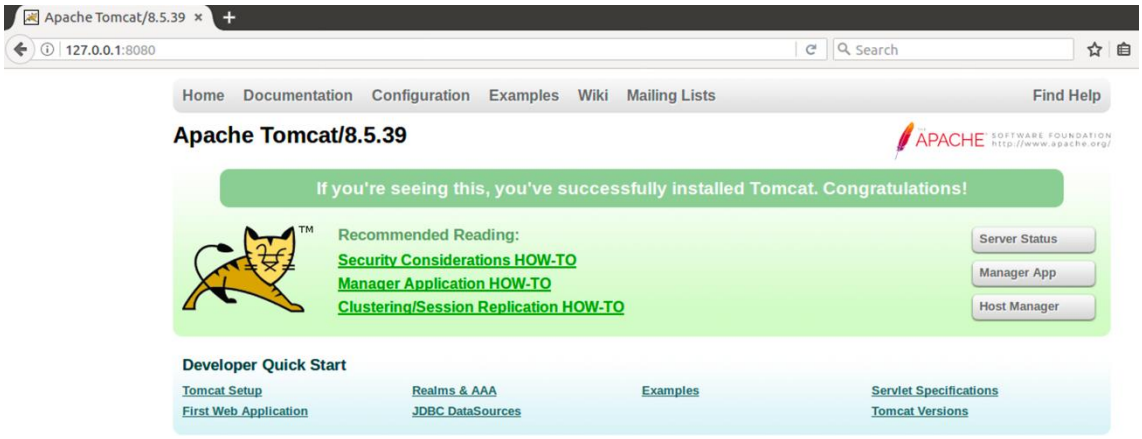
x. To enable the remote connection to access the Tomcat web manager application, users should comment out the IP restriction in the “/opt/tomcat/webapps/manager/META-INF/context.xml” and “/opt/tomcat/webapps/host-manager/META-INF/context.xml”. Following is the example of the context.xml file.

```
<Context antiResourceLocking="false" privileged="true" >
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|
Long|Number|String)|org\.apache\.catalina\.filters\.CsrfPreventionFilter\$LruCac
he(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>
</Context>
```

xi. To make sure the changes into effect, restart the Tomcat service by entering commands “sudo systemctl restart tomcat”.

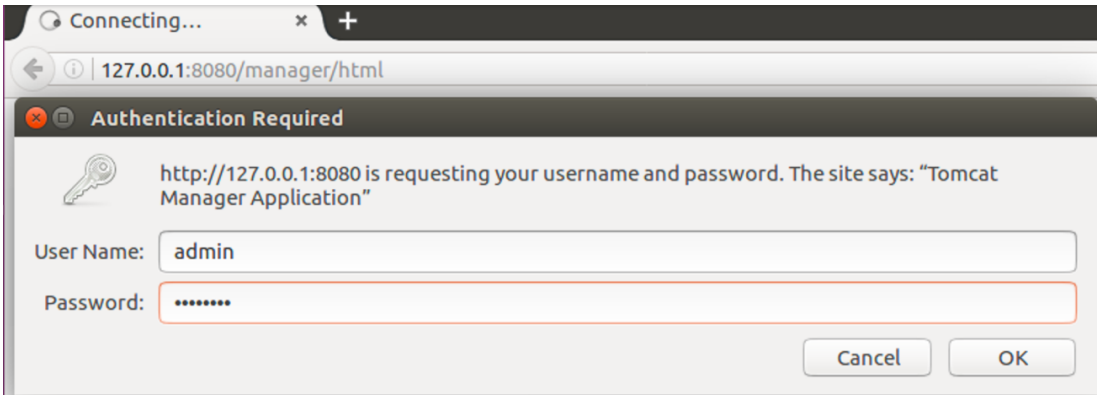
```
parallels@parallels-vm:/opt/tomcat$ sudo systemctl restart tomcat
```

xii. Now, the user can access to the web interface by entering http://[IP_ADDRESS]:8080.

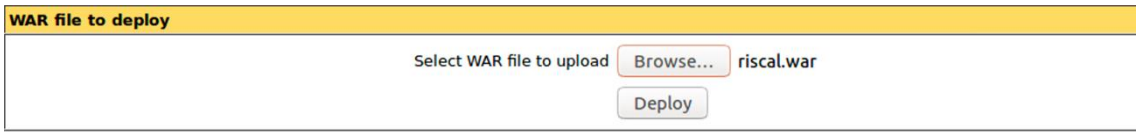


xiii. To deploy the web version of the RISCAL program, download the pre-built WAR file from https://github.com/chming1016/RISCAL_Web.

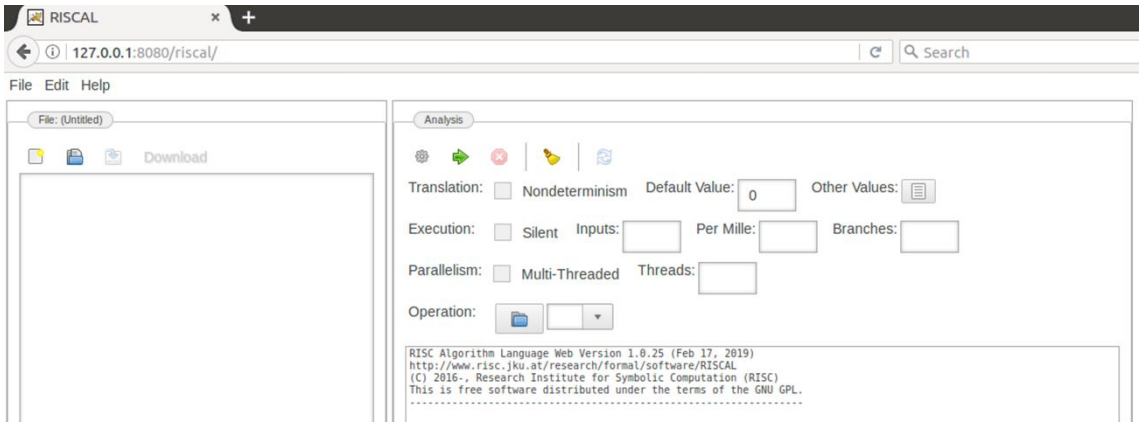
xiv. Open http://[IP_ADDRESS]:8080/manager/html in the web browser and key in the account/password which is set in Step ix.



xv. After login, select the WAR file to upload and press “deploy”.



xvi. Once the deployment is successful, open http://[IP_ADDRESS]:8080/riscal/ in the web browser. The program is ready to be used.



b. Virtual Box and Parallels Desktop Images

A simple way to use the program is to use the virtual machine to deploy. All the necessary file is installed and well configured, the only thing to do is download and start the virtual images. Following is the detail information.

Download Link:
https://drive.google.com/drive/folders/1WzSdHXXO2VADPldBuIxwdxBqYF1o2qSk?usp=sharing

User Account: parallels
User Password: riscal2019

JOHANNES KEPLER UNIVERSITY LINZ

CURRICULUM VITAE

Personal Data

Name	Hsuan-Ming Chen
Address	No. 20-6, Jiujia Ln., Beitun Dist., Taichung City 406, Taiwan (R.O.C.)
Phone	+886422416950; +886987817857
E-Mail	chming1016@gmail.com
Nationality	Taiwan
Date of birth	October 16, 1992

Education

Sep. 2015 - Jun. 2019	**Master of Business Administration (Information Management)** National Sun Yat-sen University (NSYSU, Kaohsiung), Taiwan Master Thesis - Big Data Intrusion Detection Using Cluster Computing
Sep. 2011 - Jun. 2015	**Bachelor of Business Administration (Information Management)** National Chi Nan University (NCNU, Nantou), Taiwan Bachelor Thesis - Design and Research of Linux Kernel Level Intrusion Detection System Reaction Mechanism

Skills

Developments	Docker, Hadoop, Spark, Kafka, Elasticsearch
Databases	MySQL, MongoDB, Cassandra
Systems	Ubuntu Server, Kali Linux, CentOS
Languages	Python, R , Scala, Pig Latin, Java, C, PHP, JavaScript
Hardware	Raspberry Pi

JOHANNES KEPLER
UNIVERSITY LINZ

STATUTORY DECLARATION

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
This printed thesis is identical with the electronic version submitted.

Linz, 04.04.2019

Signature