The University of Hong Kong

Final year project interim report

Web-based integrated development environment for Java, Android and a new programming language

Poon Stirling Yeu, Shum Chi Chung and Wong Man Chun

supervised by
Dr. TW Chim

January 24, 2016
Abstract

Despite the increasing popularity of using an Integrated Development Environment (IDE) for application development, traditional IDE platforms impose software and hardware requirements on the user’s computer, setting up a barrier of entry for people unable to afford a performant computer. Furthermore, popular languages such as Java and C++ supported by these IDEs often encourage bad coding habits and limit developers’ productivity. Therefore, there is a need to develop a machine-independent cloud-based IDE which can reduce dependence on the user’s computer. Additionally, to enable higher developer productivity, a new programming language will be introduced and incorporated into the new IDE. At present, the basic front-end and back-end infrastructure of the IDE has been created, while the syntax for the new programming language has been drafted and a parser to convert the language into an abstract syntax tree (AST) has been created. The project team has encountered challenges brought by a lack of time for familiarizing with the tools and workbenches needed to create the IDE and the new programming language. It is expected that these challenges can be overcome by spending time learning how to use these tools and seeking guidance from our advisers who have experience working with these tools.
Acknowledgements

We would like to thank our teachers and advisors, Dr. T.W. Chim, Dr. C.K. Chui, Dr. Jon Hui and Prof. Bruno C. d. S. Oliveira for their advice and guidance.
This project uses materials provided by the following entities.

Copyright (c) 2010, 2012, 2013 The Go Authors. All rights reserved.
jQuery v1.11.3 (c) 2005, 2015 jQuery Foundation, Inc.
Copyright (c) 2013 Bootsnipp.com
Copyright (c) 2014 Ashley Jeffs
Copyright 2016 Microsoft Azure
Copyright (c) 2014 Cenk Alt
Copyright 2016 Gary Burd
Copyright (c) 2012 Rodrigo Moraes. All rights reserved.
Copyright 2012 The Go-MySQL-Driver Authors. All rights reserved.
Copyright (c) 2011-2013, “pq” Contributors Portions Copyright (C) 2011 Blake Mizerany
Copyright (c) 2009,2014 Google Inc. All rights reserved.
Copyright (c) 2014 Andy Smith
Copyright (c) 2010, Ajax.org B.V. All rights reserved.
Copyright Joyent, Inc. and other Node contributors. All rights reserved.
Copyright (c) 2014 Ivan Bozhanov
Copyright (c) 2011-2016 Twitter, Inc.
Copyright (c) 2012-2013, Christopher Jeffrey (https://github.com/chjj/)
Copyright (c) 2014-2015 Automattic (dev@cloudup.com)
Float Label Pattern Plugin for Bootstrap 3.1.0 by Travis Wilson
Copyright (c) 2014-2015 almasaeed2010
Copyright (c) 2015 Tony Cabaye
Copyright (c) 2013 Alex Lockwood
Copyright (c) 2014-2016 almasaeed2010
Copyright (c) 2010-2013 - Gustavo Niemeyer (gustavo@niemeyer.net)
Copyright [2015] Contributors: Jacek Marchwicki (jacek.marchwicki@gmail.com), Karol Wojtaszek (karol@appunite.com)
jacekmarchwicki/android (https://github.com/appunite/docker)
ksoichiro/android (https://github.com/ksoichiro/dockerfiles)
steinwaywhw/client.js (https://gist.github.com/steinwaywhw/9920493)
Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.
Android is a trademark of Google Inc.
# Contents

1 Introduction ........................................... 6

2 Objectives and Benefits ................................. 11

3 Existing Work ........................................... 14

4 Scope ...................................................... 15

5 Design and Procedures ................................... 16
   5.1 Platform setup ......................................... 16
   5.2 Web-based IDE front-end .............................. 17
   5.3 Web-based IDE back-end ............................... 17
   5.3.1 Use of Go language as implementation language ... 17
   5.3.2 Database ............................................ 20
   5.3.3 Compiling and executing environment .......... 21
   5.4 System work-flow design ............................. 21
   5.4.1 Sign up .............................................. 21
   5.4.2 Java applications ................................... 21
   5.4.3 Android applications ............................... 22
   5.5 Security design ........................................ 23
   5.6 User interface design ................................. 23
   5.7 New programming language ........................... 24

6 Deliverables ............................................... 30
   6.1 Web-based IDE front-end ............................. 30
   6.2 Web-based IDE back-end ............................... 30
   6.3 Agent App - BuildApp Agent .......................... 31
   6.4 New programming language ........................... 31

7 Division of Labour ........................................ 32

8 Schedule of Tasks ........................................ 33
9 Current Progress and Challenges
  9.1 Web IDE front-end ........................................ 35
  9.2 Web IDE back-end ......................................... 36
  9.3 Agent App .................................................. 36
  9.4 New programming language .............................. 37

10 Future plans .................................................. 41
  10.1 Front-end development goals ............................ 41
  10.2 Back-end development goals ............................. 42
  10.3 Agent app development goals ............................ 42
  10.4 Translation language development goals ............... 42

11 Conclusion ..................................................... 43
Chapter 1
Introduction

Following the prevalence of smartphones, more than a billion smartphone users across the globe have been downloading mobile applications onto their devices [13]. As such, many client-facing organizations desire launching their own mobile application for their customers, creating a huge demand for mobile application development. While corporate teams are used to developing software on a structured, systematic basis, popular IDE software for developing mobile applications such as Android Studio and Eclipse lack native support for team collaboration. It is also time-consuming to install and configure these software and add-ons. Application development in a local environment, in that case, discourages team coding, which becomes a bottleneck for rapid development.

In fact, a lack of productivity is one of the major issues programmers face during software development, stemming from pressure to complete more software development projects in a limited time frame. A software productivity study in 1980 by Boehm et al. [6] propose a resolution to this issue by equipping developers with an easily accessible IDE platform.

An IDE is an application that contains text editor, debugger and compiler elements for software development [35, 24]. Instead of writing code in a text editor such as Notepad, programmers can now code in a more user-friendly interface, with a command-line console and debugging tools all in one place. Traditional IDEs including Eclipse and IntelliJ IDEA can automate syntax highlighting, code completion, error checking and code refactoring [15, 23]. Although developers view these tools as essentials, they need to spend significant amount of time installing and configuring these tools [12]. Said tools also consume vast amounts of CPU and memory space, which may be taxing on old computers [12]. Furthermore, with the increasing trend of project team development, a majority of these single-environment development platforms fail to serve team collaboration, which makes version
control and build management more difficult [12, 27]. As such, the traditional form of IDE platforms limits the efficiency of building applications on these platforms.

Kats et al. [27] stated that an online IDE, being a centralized platform for program development, serves as a solution to the above issues. It avoids the installation of bulky software on local machines and eliminates installation and configuration time [30]. Furthermore, every team player can access to the single copy of source code on server which can avoid version conflict and prevent duplicates. This type of Software As a Service (SaaS) platform took over the software control of the development toolkit. Programmers can code on the SaaS platform any time on any computer with internet access, without worrying about storage or computing power [27, 30]. Kats et al. [27] made three points in their research paper describing their perception of an ideal online IDE.

Firstly, a modern code editor should have a scanner and parser that processes the source code for presentation, semantic and editing purposes [27]. The research paper by Kats and Visser [26] visualizes the typical composition of an IDE parser (See Figure 1.1). In the presentation module, the outline view allows a preview of the source program structure, syntax highlighting can make syntax and operation stand out and code folding can expand and collapse text within brackets or within a document scope. In the editing module, bracket matching and automatic indentation can preserve the integrity and consistency of source code structure. Syntax completion allows developers to reduce their mental load by permitting typing the partial form of a keyword for the IDE to “complete” into the full form of the keyword, making coding faster and more efficient. In the semantic module, name analysis...
takes place to resolve references and to highlight words with no occurrences. An error or warning description should be displayed if the predicate of some error validation rules, usually implemented by regular expression, is satisfied.

Secondly, an online IDE should facilitate team collaboration processes. Inside an organization, multiple developers can edit the source code at the same time. Source code can be synchronized by web browsers uploading the source code changes to a server which then broadcasts these changes to other online clients. Operational transformation is implemented to ensure that the version of source code stays the same on all client browsers without being affected by connection delay or sudden disconnection and re-connection of computers [4]. Issue tracking is also an essential feature for project management and progress tracking. Other related features include version control and build management systems such as Git, which can ensure data consistency, facilitate branch development and allow the restoration of previous version of source code. A group chat function can also be implemented to foster group communication.

Thirdly, it is possible to implement e-learning features in the online IDE. Utilization of a cloud development platform can eliminate the difficulty of installing software in different lab machines or students computers. Furthermore, professors can, all on the online platform, view students’ assignments, run their homework and grade and comment on assignments. Compared with submitting a compressed file of source codes to professors, a cloud IDE can enhance teaching efficiency significantly.

Apart from the points mentioned by Kats et al. [27], more features can be implemented in an IDE to align with typical modern software engineering processes. Similar to the popular team collaboration platform JIRA, user stories can be assigned to different team members in the organization [2]. In order to further simulate software development, developers can add clients onto the platform and send emails to them so that both clients and developers can fill in comments during presentation meetings. These user stories are then shown on the IDE panel alongside with the text editor. Therefore, developers do not need to switch windows for stakeholders’ comments and can focus on the issue at hand.

In addition to improving the user interface of the IDE to facilitate development processes, coding efficiency is also directly affected by the complexity of the programming language itself. More specifically, programmers often spend a substantial amount of time coding to cater for vulnerabilities exhibited in popular programming languages such as Java. One major obstacle to developer productivity is committing mistakes that the compiler can detect. In one case, the so-called `NullPointerException` has been called a ‘billion dollar mistake’ by its inventor Tony Hoarse, due to its frequent occurrences
in numerous Java projects [19]. Although the compiler/interpreter can differentiate between nullable and non-nullable types, the decision to make all object types nullable delegates the task of type safety to the developer instead of the compiler. Compared with newer programming languages such as Python and Ruby, Java is verbose; it uses more lines of code to represent the same concept and often contains boilerplate code. Furthermore, Java is built from Object-Oriented Programming (OOP) concepts, leading to an unnatural integration for alternative programming paradigms. In response to Java’s shortcomings, different alternative Java Virtual Machine (JVM) languages, such as Scala and Groovy, have been developed. While they avert the problems of Java, the need to compile down to JVM bytecode causes long compilation and linking time, thereby reducing developers productivity and desire to use these languages. Part of the goal of this project is to build a new programming language that addresses these shortcomings yet does not hinder developers work.

Android programming is chosen as the mobile application programming language that can be developed in the IDE because of the higher penetration rate of Android devices, higher transparency of source code and lower technical difficulty. First, mobile application usage has experienced significant growth in recent years [28, 5]; as of the second quarter of 2015, Android has had an market share of over 80% in the mobile operating system market [22]. Second, due to the open nature of Android [20], it is easier to push applications developed by the IDE to Android devices by installing APK files [20, 17] when compared to the iOS platform [20]. Java is also chosen to be implemented because Android uses Java as the application’s development language [1] and Java is one of the most popular programming language in the world [37].

In this project a number of hypotheses are made when the new IDE was designed. It is proposed that collaborative coding is an essential feature for Rapid Application Development (RAD). Additionally, mobile applications can be tested on a physical device in a wireless manner. Finally, It is posited that a new programming language can open up the possibilities of other programming paradigms beyond the traditional OOP approach.

This project intends to make three contributions. Differing from traditional IDE software, modern software engineering practices as well as e-learning elements are incorporated into the IDE platform. The user story module on the IDE enables programmers to view user requirements at the same time and provide students a question panel when they code. This project also shortens the time for developing, testing and deploying applications by pushing applications to mobile devices, testing applications within isolated virtual machines and eliminating local installation. Last but not
least, a new programming language is introduced to foster development speed and to ease development complexity.

The remainder of the report will proceed as follows. First this report discusses the objectives and benefits of this project, followed by the scope and the deliverable of the project. Then, it discuss the current work done and challenges encountered. Finally, future work as well as goals for this project would be addressed before closing up with a conclusion.
Chapter 2

Objectives and Benefits

This solution is to build a web-based IDE with code editing, project management tools as well as compilation and debugging tools, for Android and Java development, and to facilitate development with a new translation language. The goal of this project is to offer a more efficient tool that can match modern programming technique seamlessly.

In order to achieve this goal, three objectives were introduced when the IDE was first proposed, namely to shorten the time for developing, testing and deploying applications, to make programming lectures more interactive and efficient, and to integrate modern software engineering elements into development processes.

Compared with traditional IDE software, such as Eclipse and IntelliJ IDEA, a cloud-based IDE platform enjoys substantially more benefits. Programmers can code anywhere and anytime with a computer connected to the Internet. They can start their project instantly without spending time installing software. Users do not have to install dependency files or compilation packages when they build their project. Also, software development can be carried out without hardware constraints. Code can be taught and shared more easily such that more people can benefit from learning code in a systematic and efficient way.

Instead of testing mobile apps on devices one by one with a bunch of USB cables, the IDE can test multiple devices cable-free seamlessly with a single click. The IDE also indicates whether the test is carried out on each of the devices successfully. The screen output of these testing devices are then recorded and stored in the development platform so that the testing process can be more measurable and quantifiable.

Moreover, a new programming language (tentatively named Java+) will be introduced into this cloud IDE platform. The new programming language not only aims to reduce verbosity and boilerplate code, but also encourages
the use of different programming paradigms. It is also designed to prevent developers from making mistakes the compiler/interpreter can detect, maintaining a higher level of performance and efficiency. If even some developers are interested in learning a new language, Java+ can still benefit these users with the “smart preview” function. In a typical IDE, developers can only toggle the visibility of code blocks (i.e. hide/unhide them), which allows them to focus attention on the important sections but does not allow them to refer back to the hidden blocks. In a smart preview, Java+ code logically equivalent to the underlying Java code will be displayed to the developer, giving they now have a bird-eye’s view of different components. Since Java+ code is often many times shorter than the equivalent Java code, this avoids the issue of cluttering the screen when the developer previews the code. We envision that the gentle introduction to Java+ will lead developers to adopt it over time.

BuildApp also offers an original and innovative “WISE” editor for development teams, WISE standing for:

- W: Wireless mobile application testing. Mobile application is compiled online and its executable is then tested on multiple testing smartphones with the agent app installed. The application can be run on these testing devices by clicking a Run button on the interface.

- I: Integrated project management. Clients can play an active role in this application development platform for the first time. They can provide feedback for developers to review their code when they develop the application on the same user interface.

- S: Syntax-highlighted collaborative editor. Multiple developers may code on the same file simultaneously. Position tag can visualize the location of the cursor of other developers.

- E: Easier application development. The platform supports a new translation language so that Java code can be automatically generated. This can facilitate advanced and more experienced developers to build complicated programs more efficiently.

The IDE platform also inherits multiple form of Everything as a Service (XaaS) [3]. By definition, XaaS is a type of cloud computing under which a single form of or, more often, a combination of services are provided online. [32, 36, 9, 18, 14] These services are easily accessible, independent of device and location, often with flexible scalability, customizability and reusability.
A normalized form of service types provided by BuildApp can be summarized as five major categories.

Firstly, BuildApp provides a Platform as a Service (PaaS) to software development teams. PaaS is a type of online applications or services that customers can create and deploy their solution on a cloud platform with the aid of a pool of resources supplied by the software provider. In this case, BuildApp provides a development environment for customers to create their own projects and develop their applications. By using the Docker Containers hosted on BuildApp server, users can also compile and deploy their software projects. Therefore, a software development platform is built for software developers to build their own applications.

Moreover, BuildApp provides a SaaS to software project managers who are desperate in finding a collaborative application development service. SaaS is a type of online applications or services that customers can use or subscribe instead of setting up their own self-host application. In practice, customers can purchase BuildApp services on a subscription basis. Since the system architecture is centrally hosted, customers do not have to worry about security, storage and network problems. Therefore, users can utilize software development services provided by BuildApp to empower their software development processes.

In addition to software development processes, one of the missions of BuildApp is to offer Education as a Service (EaaS) for IT training coaches and students who want to learn code. Students can code their program assignments and teachers can process their assignment using an online compiler. Marks can be produced online and teachers can also monitor their progress in completing those assignments.

Apart from education service, BuildApp offers Communication as a Service (CaaS). The integrated project management function facilitates direct communication between developers and client. Developers can make their changes on the program files according to the users feedback on the same user interface.

Last but not least, BuildApp provides Logging as a Service (LaaS) for analysts and project managers. When mobile applications are compiled and run on testing devices, these logging information will be stored in BuildApp database which can be further analyzed and visualized in a distribution of devices. The agent app can also capture the screens of every testing device and send it back to the BuildApp database for business analysts to produce a higher quality of documentation. It can also make software projects more measurable, quantifiable and traceable throughout the testing processes.
Chapter 3

Existing Work

Current competing cloud-based IDEs in the market are Cloud9, Nitrious.io and Codeanywhere. However, these IDEs lack several key features of our IDE. The process of deployment is clunky and inefficient, with no innovation in this area as compared with traditional IDEs. Since developers spend considerable time building different versions of their apps and then deploying them to devices for testing, developers’ productivity is not significantly increased. Furthermore, they do not address the problem of the limitations on developer productivity due to constraints of existing programming languages. New programming languages that aim at increasing productivity such as Scala and Groovy also exist, but they are usually not supported in these cloud IDEs. They also have other issues discouraging developers from using them, such as having slow compilation or interpretation time, or taking up significant amount of space in an application package. The IDE and new programming language integration tries to avoid the existing problems present in current solutions.
Chapter 4

Scope

In this project, three components that make up the whole software system will be developed. First, a web-based IDE with coding, compilation and debugging functionalities for Android and Java software development, which can support a large number of users, will be implemented. A runtime debugging and testing tool that can be installed into Android devices, allowing users to run their Java applications on our server and to preview the compiled application in their Android smartphones, will also be created. Last but not least, a new programming language and its interpreter or compiler will be developed.
Chapter 5

Design and Procedures

The overall process model this project will follow is the Agile procedure. In this procedure, components will be divided into tasks and assigned priorities. During every sprint, tasks with the highest priority will be assigned to each member for completion. At the end of each sprint, the team should be able to develop a usable product and will gather feedback. According to the feedback received, the team can re-assign priorities to existing and new tasks and start a new sprint. Adopting this methodology allows the team to move on quickly. As there are many components in this project, the following sections will describe the design and procedures for each components in detail.

5.1 Platform setup

The development server fyp-pc06 has been assigned to our team by the Department of Computer Science in late September. After setting up configurations of the server, virtual machines were created and tested. The login and catalog pages have also been created and they are now running to test the web server component, including Nginx, the web server program and the web server component of the Go programming language. Screenshots of these pages are shown in figure 9.1 and 9.2 respectively.

In the meantime, the IDE website environment has been set and is ready for construction of the IDE platform. Fundamental programming packages have been installed onto the web server. Basic features such as logging in, database querying and port listening in the back-end were already implemented.
5.2 Web-based IDE front-end

One of the major features delivered in our front-end is collaborative editing, through which multiple developers may edit the same file simultaneously on the platform. This can be achieved by implementing the terminology of operational transformation (OT) in our system. OT eases the concurrency of online groupware system, which can be explained by an example below.

Assume two developers, John and Mary, are editing the same file simultaneously, as shown in figure 5.1. After they type a character on our coding editor, the browser client then sends an `insert` instruction to the server, together with the character input and the position the character is inserted to the document. Since messages are received by the server at a different time instant (even within an interval of milliseconds), the server can handle the messages in chronological order and correspondingly alter the file copy on the server. Then the messages received by the server are broadcasted back to the clients to notify changes made by other browser clients and to verify that the server has received the `insert` message and made the changes. This can resolve the version control issue involving multiple clients in the network.

However, the system is likely to cause a positioning issue when John and Mary type on the same line, as figure 5.2 illustrates. Suppose both developers insert a word on the same line and their browser send a command to the server, similar to the workflow above. Due to network latency, when John sends an `insert` message to the server, there is a time period when Mary sends an `insert` message without realizing the changes made by John. Hence, the position to insert marked by Mary may not be valid when server processes her request. In this case, server first inserted the words “hello world” instructed by John. Then it inserted the word “smart” at a wrong position because `hello world` is inserted at the beginning of the sentence, changing the position of the original phrase.

To cater for this problem, the terminology of OT was put forward in 1989. In relation to concurrency control, the mechanism of groupware event handling has become the fundamental principle and design of office automation and the basis of online collaboration.

5.3 Web-based IDE back-end

5.3.1 Use of Go language as implementation language

In order to discover a programming language suitable for our server design, four different back-end programming languages, namely PHP, Go, Ruby and
Figure 5.1: Two developers may edit the same file simultaneously by sending an insert command to the server by providing the keyword and position number.

Python, were evaluated to determine the programming language our business logic would be implemented on the web server. They were evaluated on maturity, nature of variable types, nature of language, whether the language provides native concurrency support and whether the stack size is dynamic Table 5.1 summarizes results of the evaluation.

Developing PHP pages might be relatively easier because there might be more online resources. However, its inconsistent naming format and dynamically typed variables might occupy more processing memory, which in turn might slow down processing speed. Among the three strictly typed programming languages, Go is better in terms of runtime speed. While Ruby and Python are interpreted programming language, Go is a compiled programming language. In other words, an executable generated from Go files may get better code optimization. Go, as a statically typed language, has better performance in concurrency because of its dynamic stack size for threads [8]. Where in Go each thread only requires 4KB of stack, Python needs 32KiB and Ruby (implemented in C) needs 1MB. Also, channels and locks between Goroutines allow thread-to-thread communication. In spite of the fact that Go is a relatively new language which might be less mature than PHP, Ruby and Python, its features such as pointer reference, package support, zero initialization and garbage collection is similar to those features in Java, shortening our learning time.
The performance of concurrency control in Go Language is better than that in other programming language. Messages can be transmitted between different Go routines with simplistic implementation, which can be demonstrated by the code snippet 5.1.

Listing 5.1: Sample code of Go concurrency and messaging

```go
going main
import (  
  "fmt"
  "time"
)
func pinger(c chan string) {  
  for i := 0; ; i++ {  
    c <- "ping"
  }
}
```

Function pinger here accepts c as a channel variable and passes the message “ping” to the channel c and wait for retrieval. Channel is a special transfer pipe provided by Go Language that allows information exchange between two concurrent threads. The message is then received by another Go routine called printer and printed out on the console in program 5.2 [8].

Listing 5.2: Code to receive messages from program 5.1
<table>
<thead>
<tr>
<th>Language</th>
<th>PHP</th>
<th>Go</th>
<th>Ruby</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature of Variable Types</td>
<td>Dynamically typed</td>
<td>Statically typed</td>
<td>Dynamically typed</td>
<td>Dynamically typed</td>
</tr>
<tr>
<td>Nature of Language</td>
<td>Interpreted</td>
<td>Compiled</td>
<td>Interpreted</td>
<td>Interpreted</td>
</tr>
<tr>
<td>Native Concurrency Support</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Dynamic Stack Size</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of PHP, Go, Python and Ruby for the implementation of back-end processing language

```go
func printer(c chan string) {
    for {
        msg := <- c
        fmt.Println(msg)
        time.Sleep(time.Second * 1)
    }
}

func main() {
    var c chan string = make(chan string)

    go pinger(c)
    go printer(c)

    var input string
    fmt.Scanln(&input)
}
```

The above example illustrates how channels can be used to implement concurrent procedures in Go, which may provide a better multi-tasking environment with advanced memory optimization [8]. Such efficient inter-thread communication may help implement our design more effectively. Therefore, the Go language is selected as the back-end server processing language during the design phase.

### 5.3.2 Database

MongoDB has been chosen as the database system. The following are the reason for this choice. From Seguin, a fixed schema need not to be defined for the collection in the database which provide more flexibility[25]. Besides,
with this flexibility on the schema, it works better with OOP, which objects could be saved directly to database after serialization[25].

5.3.3 Compiling and executing environment

Docker[10] is used to provide an compiling and executing environment for users. Each organization is provided with a preconfigured Docker container which contains necessary developing tools and compilers for Android and Java development includes the Android Software Development Kit and Java Development Kit. The Docker Container also provides a fully functional terminal, allowing access to the underlying Linux OS of the Docker container. As a result, the docker container not only permits users to compile their Java/Android project manually, but also provides the flexibility of installing any additional tools inside the container that can assist their development work. Thus, developers can code in an all-in-one interface and a highly customizable development environment.

5.4 System work-flow design

The designed work flow of the usage of the IDE will be described in below subsections

5.4.1 Sign up

In BuildApp, the administrator can create the organization with its first user account. After creating an organization, the administrator can create new account for its users. He can then send the activation instructions to another user. The user can complete account registration by entering a new password and the given invitation code. Behind the scenes, one Docker container is assigned to each organization and boot up. Users can now start their software development by creating projects.

5.4.2 Java applications

To develop a Java application, users need to create a Java project. The flow of developing a Java application is shown in figure 5.3. Users can write their code on the front-end of IDE collaboratively. After users finish writing the code, they can start the compilation of the source code files under the project directory, which is carried out inside a Docker container assigned to the organization. After the compilation has finished, users can execute the
compilation result. The execution output is sent back from back-end server to front-end interface and displayed to users.

5.4.3 Android applications

For Android application development as shown in figure 5.5, after users create an Android project, they can write code collaboratively. After clicking the ‘Compile’ button on the interface, the Android project is then built inside the Docker container assigned to the organization. The application package would be generated.

For deploying Android projects for testing and debugging, BuildApp supports testing Android application on ‘real’ devices remotely. Beforehand, users have to register the testing devices to BuildApp using the BuildApp Agent - the Android application of BuildApp. Then, when the developer wants to test the compiled app, he/she can click the Run button. Then they will be prompted to select, from the list of registered devices, which device(s) they would like the app to be tested on. Upon selection, BuildApp Agent on the selected device(s) will automatically download the app. For security reasons, users need to manually accept the permissions before the app is installed and launched. Then, functionalities of the application can be tested. During testing, BuildApp Agent collects system debug messages in the background and sends back to the IDE. In the deploy panel of the IDE, users have a bird’s-eye view of the status of all devices and identify and resolve issues by analysing the messages collected.

In order to enhance software development productivity, BuildApp also provides features designed to streamline software development workflows.
For Android projects, BuildApp can deliver the testing app not only to developers’ device but also to clients' devices during presentation. Clients can test the application after developers deployed the app. BuildApp also allows clients to provide feedback directly from the device to the developers using BuildApp agent. To deploy an application to client device, the client needs to install and sign in the agent app with a client account or with an invitation code from developers. During app presentation, the development team can push the finished app to the client immediately and the client can test the latest version of the application. When he/she wants to give feedback or software bug reports, the client can take screenshots or even screen recordings and send them together with comments back to the developers via the agent app. The developers can then read the information provided by the client and fix any issues identified.

5.5 Security design

Due to the project scale, an individual server is used to implement the BuildApp platform. A public IP address is assigned to the IDE server which facilitates deployment, testing and demonstration processes. Also, a SSL certificate is applied to the BuildApp platform. Any unused ports that can be pointed by the public IP are blocked by the middleware SSL proxy server before data reaches the server.

Each compilation job and compiled executable of an organization is run in the pre-assigned Docker Container which can only be accessed by the users of that organization. As stated in Docker’s official website[11], applications running in a Docker Container is isolated from those in other containers. This can prevent users from affecting each other and provide a better user experience. Files, including source code files, are stored in folders that only the owner or authorized users have the right to access. This makes a private source code repository for users and prevents damages by unauthorized users, as well as protects intellectual property.

5.6 User interface design

When a user is logged in, he/she can enter the catalog page. Here, the user can create a project for the organization. Administrators in the organization can change the visibility and locks of code in the project files. A code editor has been set up together with the directory list on the left of the interface. When a user chooses a project, he can open a specific file and edit
the file content. Any changes in file content is synchronized to our back-end server immediately and the changes are saved automatically. More than one developer can edit a file simultaneously.

The user interface of BuildApp IDE editor is influenced by that of popular programming IDE in the market, so that developers can get familiar with BuildApp more easily and hence start their work immediately on the cloud platform. The default dark-coloured IDE theme aims at protecting developers eyes. [21] There will also be an alternative in selecting the appropriate theme for the editor.

The left column of the IDE editor is the directory list. In the list, the directory loads all the files and folders of the project directory. When a folder is clicked, a list of files inside the folder can be seen. When a file is clicked, the current editor will terminate the current file and replace it with the content of the newly clicked file.

Inside the editor, syntax highlighting and hinting as well as automatic indentation should be implemented for Java code used in Android and Java programs. A position tag can also be shown, visualizing the location of cursors of other online users who are also editing the same file.

The right column of the editor is the project management module, where issues and the corresponding client’s feedback are displayed. Furthermore, the terminal of the docker assigned to the organization is displayed underneath the editor so that developers may take control over the docker on the same user interface.

When the Internet is disconnected from the computer, the text inside the editor cannot be changed until the computer is re-connected to the Internet.

5.7 New programming language

One approach to creating a new programming language is to perform modifications to an existing programming language such as Python or Ruby, and then integrate this adopted language with Java. There are numerous advantages with this language design; the large amount of learning resources supporting these established languages renders a manageable learning curve for the majority of developers, thereby encouraging adaptation of this new language. Furthermore, since these languages already have a mature implementation, time needed to create a working adaptation is likely reduced compared to implementing a completely new programming language.

Despite these advantages, there are major shortcomings that prevent this approach from achieving the stated goals for the new programming language. Since the syntax of Java and the modified programming language greatly
differ, programmers will need to mentally switch contexts between the two languages as they alternatively edit source files encoded in Java and the new programming language, reducing their productivity.

A more serious issue is the penalty brought by runtime interoperability and translation. As shown in figure 5.6, Java classes exist in Java runtime (the green box) and classes of the new programming language exists in a different runtime (the yellow box). When data or functions flow from classes of one runtime to classes of another runtime, a procedure called translation happens, allowing the classes to communicate across the runtime boundaries. This procedure will need to run every time the boundaries are crossed. Furthermore, this procedure grows in size as more classes need to be translated into a different runtime. The cumulative time and space penalties in a medium to large application would bring a visible impact to the performance of the application and discourage developers from using the new programming language.

Therefore, another language design will be used, called the thin-layer approach as illustrated in figure 5.7. In this design, the new programming language is a thin layer on top of Java, meaning that the syntax diverges only slightly from Java. Differences in syntax are mainly for the purpose of avoiding well-known design pitfalls of Java. Due to the similarity, the translation from the new programming language back to Java or Java bytecode require less time and effort. This avoids the issues associated with integrating an adopted language.

Based on the decision to use a thin layer approach, there are two more technical aspects to consider before implementing the new programming language. Table 5.2 shows the advantages and disadvantages of the two types of output that can be generated from interpreting/compiling the new programming language, namely Java source code and Java bytecode, while table 5.3 evaluates the 5 different interpreter or compiler frameworks, namely LLVM, PyPy, MPS, Graal/Truffle and XText, on key criteria such as maturity, scale of Java support and Java interoperability support.

Comparing the advantages and disadvantages of outputting Java source code versus outputting Java bytecode, both have advantages that can contribute to the stated goals of the new programming language. The plug-and-play nature of Java source code output means that developers do not need to create special configurations for the new programming language and do not need to worry that incorporation of the new code into existing applications will create additional errors. Overall, this allows the developer to be more efficient. On the other hand, Java bytecode is more performant in nature as a second round of compilation is not needed. Balancing the benefits and disadvantages of the two approaches, outputting Java is a better solution for
Java source code
- Plug & play
- Can utilize existing Java libraries and runtime
- Can integrate with any environment that runs Java with further configuration

Java bytecode
- Performant
- Easier to create than Java output in some cases

<table>
<thead>
<tr>
<th>Output</th>
<th>Java source code</th>
<th>Java bytecode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantages</td>
<td>- Plug &amp; play</td>
<td>- Performant</td>
</tr>
<tr>
<td></td>
<td>- Can utilize existing Java libraries and runtime</td>
<td>- Easier to create than Java output in some cases</td>
</tr>
<tr>
<td></td>
<td>- Can integrate with any environment that runs Java with further configuration</td>
<td></td>
</tr>
<tr>
<td>Disadvantages</td>
<td>- Encourages manual editing of output</td>
<td>- Cannot integrate with existing environment without configuration</td>
</tr>
<tr>
<td></td>
<td>- Requires more time to create</td>
<td>- Reduces portability of output</td>
</tr>
</tbody>
</table>

Table 5.2: Comparison of Java source code output and Java bytecode output from the new programming language

the project for two main reasons. Outputting Java allows a plug-and-play approach where the output can simply be dropped into wherever the old code existed. This can ensure that future maintainers unable to read Java+ can still maintain the project. In addition, for Java+ to be used in the smart preview function, we must build a Java to Java+ translator. If a one-to-one mapping from Java+ to Java is established, it can ease the development of the Java to Java+ translator. Since LLVM, PyPy and Graal/Truffle outputs Java bytecode, it is removed from consideration and only MPS and XText are considered.

Although MPS and XText are similar in the criteria applied, the approaches taken by MPS and Xtext in creating the language are different. XText uses a more traditional approach that uses a parser to parse textual files. On the other hand, MPS uses a projection-based editor [38] that appears like text but actually allows editing of the underlying AST structure. While the additional layer of abstraction in MPS provides certain advantages, such as easier embedding of languages in existing languages, this does not provide much benefit since a new language is created. However, the underlying format is specific to MPS and is harder to reuse outside of the workbench. XText, on the other hand, allows the language to be reused outside the workbench as other tools can manipulate the textual files. Therefore, XText will be the language workbench used to develop the language.
<table>
<thead>
<tr>
<th>Framework</th>
<th>LLVM</th>
<th>PyPy</th>
<th>MPS</th>
<th>Graal/Truffle</th>
<th>XText</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outputs</td>
<td>Bytecode</td>
<td>Bytecode</td>
<td>Source code</td>
<td>Bytecode</td>
<td>Source code</td>
</tr>
<tr>
<td>Maturity</td>
<td>Stable</td>
<td>Stable</td>
<td>Stable</td>
<td>Under R&amp;D</td>
<td>Stable</td>
</tr>
<tr>
<td>Scale of Java back-end support</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Just in time optimization in JVM</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Java interoperability</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5.3: Comparison of LLVM, PyPy, MPS, Graal/Truffle and XText for the implementation of a new programming language
(a) BuildApp Agent app login screen

(b) BuildApp Agent app successful login screen

(c) BuildApp Agent app successful login screen
Figure 5.5: Android application development work-flow

Figure 5.6: Translation happens when classes transfer data across the runtime boundaries

Figure 5.7: The thin-layer design creates a new programming language which has a syntax that only slightly differs from that of Java
Chapter 6

Deliverables

Deliverables are divided into 4 major components, namely the IDE front-end, the IDE back-end, the helper Android application and the new programming language.

6.1 Web-based IDE front-end

The web-based IDE front-end should include the following features:

- Coding editor with syntax and application program interface (API) hinting and searching
- Drag-and-drop graphical user interface (GUI) editor
- Seamless collaborative coding
- Git version control integration
- Code and application templates
- Remote delivery of executables to user devices
- Displaying execution result of Java terminal-based programs
- User account management for teams

6.2 Web-based IDE back-end

The back-end infrastructure will provide a separated environment for each user. It will also compile code, store files as well as host the database system. The back-end infrastructure will also include a communication API for
the front-end infrastructure to obtain user data and files as well as execute commands in the separated environment of each user. Besides, the back-end server will also communicate with the helper Android application, which will be explained in the next section.

6.3 Agent App - BuildApp Agent

The Agent App - BuildApp Agent helps developers to deploy Android applications to mobile devices for testing and debugging, by both developers and clients. Developers first sign in BuildApp Agent with their account in the IDE. For clients, signing in using an invitation code from developers. After signing in, the device will be bound to the account of the user or bound to a project. To deploy the application, the developers can select a device from a list of devices they have added to their account or bound to the project. The application package file will be pushed to the selected device and a prompt will appear for the device users to confirm the installation. After the installation is completed, the device user can start and test that application. BuildApp Agent would also collect debugging messages from the device and send them back to the server in order to display the messages to developers for debugging purposes. Moreover, BuildApp Agent also provides functions includes screen capturing and recording. During testing, the screen of the device could be recorded and send to developers. In this way, whenever there is a bug, developers can investigate into the videos and fix the issues accordingly.

6.4 New programming language

The new programming language will consist of three to four parts: the specification, the Java to Java+ translator, the Java+ to Java translator and integration with the cloud IDE, including the smart preview function.
Chapter 7

Division of Labour

Wong Man Chun Kelvin and Shum Chi Chung Charlie will be responsible for the development of the cloud IDE. Back-end infrastructure development will be shared between them. Kelvin is in charge of the creation of the server logic and database structure. front-end development will be Kelvins responsibility including designing the UI and creating the coding editor, syntax hinting tool, API hinting and searching tool, drag-and-drop GUI editor, collaborative coding tool, e-learning features, version control integration and code/application templates. Charlie will take charge of all other tasks including setting up the Android build system [7] and Java compiler on the Docker containers [10], creating the connection between the IDE and the build systems, implementing the Agent App on Android to receive Android application packages with debugging tools and feedback collection features, and logic on the IDE website to receive and display messages and feedback sent from the Android devices.

Poon Stirling Yeu and Wong Man Chun Kelvin will take responsibility for the development of the new programming language. Stirling is in charge of feature design and implementation, support for alternative paradigms and the implementation of compilation back-end targeting Java/JVM bytecode/assembly code/ Javascript or interpreter. Kelvin will be assisting in language design and implementation of the aforementioned compiler/interpreter.
# Chapter 8

## Schedule of Tasks

Table 8.1 shows the planned schedule, with milestones after Jan 2016 subject to change based on comments from our mid-term review.

<table>
<thead>
<tr>
<th>Date</th>
<th>Tasks</th>
<th>Completed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 2015</td>
<td>Conduct research of literature and existing solutions</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Collect user stories</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Learn required technologies and skills</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create project plan and website</td>
<td></td>
</tr>
<tr>
<td>4 Oct 2015</td>
<td>Deliver project plan and website</td>
<td>Yes</td>
</tr>
<tr>
<td>Early Oct 2015</td>
<td>Initialize server infrastructure and system architecture</td>
<td>Yes</td>
</tr>
<tr>
<td>Early Nov 2015</td>
<td>First prototype of web IDE:</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Login and catalog screen</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Back-end support for user registration and database linkage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>First prototype of new programming language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Draft syntax of language</td>
<td></td>
</tr>
<tr>
<td>Mid Dec 2015</td>
<td>Second prototype of web IDE:</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>- Back-end support for user login and user session</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Android and Java text-based code editing and compilation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Second prototype of new programming language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- AST-producing parser</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Basic interpreter for AST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Docker container and virtual machine deployment</td>
<td></td>
</tr>
<tr>
<td>Mid Jan 2016</td>
<td>First roll out of web IDE</td>
<td>Yes</td>
</tr>
<tr>
<td>Date</td>
<td>Version Description</td>
<td>Status</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Late Jan 2016</td>
<td>Third prototype of web IDE&lt;br&gt;- Syntax highlighting and debugging support&lt;br&gt;Initial new programming language integration</td>
<td>In progress</td>
</tr>
<tr>
<td>Mid Feb 2016</td>
<td>Fourth prototype of web IDE&lt;br&gt;- Display of execution results&lt;br&gt;- Syntax hinting and simple code error detection&lt;br&gt;Third prototype of new programming language&lt;br&gt;- Initial new programming language integration</td>
<td>No</td>
</tr>
<tr>
<td>Early Mar 2016</td>
<td>Fifth prototype of web IDE&lt;br&gt;- e-learning features</td>
<td>No</td>
</tr>
<tr>
<td>Late Mar 2016</td>
<td>Sixth prototype of web IDE&lt;br&gt;- Project management, issue tracking&lt;br&gt;Final prototype of new programming language&lt;br&gt;- Final new programming language integration</td>
<td>No</td>
</tr>
<tr>
<td>Apr 2016</td>
<td>Second rollout of web IDE</td>
<td>No</td>
</tr>
</tbody>
</table>
Chapter 9

Current Progress and Challenges

The current progress of various components of the project are as follows:

9.1 Web IDE front-end

For the front-end of the IDE, a login page (screenshot shown in figure 9.1) and a user creation page has been created. The user creation page contains Javascript for checking validity of user input, and a user creation request will be made to the server after the user has filled in the user creation form. Additionally, a catalog page (screenshot shown in figure 9.2) has been created which list the source code files that a user has been created.

The catalog page has been created to list all the projects owned by the organization and enable user to create a project in specific programming language for the organization. A code page has also been created to implement the IDE editor. The directory list loads all the folders and files inside the project directory.

On the code page, the editor can synchronize the text changes with remote server using the terminology of operational transformation. Syntax highlighting and automatic indentation is also in place. The editor can switch to a new file content when the file name is clicked in the directory list on the IDE. File operations can also be done in the context menu of the directory list. If there are multiple developers editing the same file, a position tag is shown on the editor to indicate the cursor position of another user in the file.

However, fellow developers who have tested the user interface have given us feedback and pointed out possible improvements to the user friendliness of the user interface. Future plans will include such improvements.
9.2 Web IDE back-end

For the back-end server program, the following tasks have been completed:

- Organization and users creation
- Java Project Creation
- Creation of designated Docker Container for each organization
- Compiling and executing Java Programs
- Building of Android Projects
- Pushing Android application installation package to real device

9.3 Agent App

The Agent App now allows user login and device registration on server. It can also receives request from server to download and install and Android Application
9.4 New programming language

After deciding to proceed with XText, the syntax of the new programming language was developed, deriving heavily from Java and Javascript as a starting point. Listing 9.1 shows a sample of the syntax.

Listing 9.1: Sample of new programming language syntax

```java
package app {

    entity MainActivity extends Activity {
        def onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.main)
        }

        def onCreateOptionsMenu(Menu menu) {
            // Inflate the menu
            getMenuInflater().inflate(R.menu.main, menu)
            val button = findViewById(R.id.button1) as Button
            button.setOnClickListener(
                {view |
                    Toast.makeText(MainActivity.this, "Button Clicked", Toast.LENGTH_SHORT).show()
                }
            )
            return true
        }
    }

    declare}
```

After creation of the syntax, a grammar defining the language has been
produced. XText can then automatically use the grammar to produce a parser that converts the input code into an AST. Then, a translator program which has also been developed converts the AST into the AST of Java. With this component, XText can take the outputted Java AST and turn it back into the ultimate Java output. Therefore, with the two components we have developed, the full pipeline has been completed where the user can type code in Java+ and get Java output. Listings 9.2 and 9.3 show a sample piece of Java+ code and its outputted Java code respectively.

Listing 9.2: Java+ code sample

```java
import java.util.ArrayList
import java.util.Arrays
import java.util.Collections

package app {
    entity Person extends Comparable<Person> {
        var age : int
        val friends = new ArrayList<Person>()

        def youngestFriendByDistance(int distance): Person {
            Collections.min(
                if (distance == 0) friends else
                friends.map([Person p |
                    Collections.max(Arrays.asList(p, p.
                        youngestFriendByDistance
                )
            )
        }
    }
}
```

Listing 9.3: Java output

```java
import java.util.ArrayList
import java.util.Arrays
import java.util.Collections

package app {
    entity Person extends Comparable<Person> {
        var age : int
        val friends = new ArrayList<Person>()

        def youngestFriendByDistance(int distance): Person {
            Collections.min(
                if (distance == 0) friends else
                friends.map([Person p |
                    Collections.max(Arrays.asList(p, p.
                        youngestFriendByDistance
                )
            )
        }
    }
}
```
Listing 9.3: Java output for Java+ code in listing 9.2

```java
package app;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import org.eclipse.xtext.xbase.lib.Functions.Function1;
import org.eclipse.xtext.xbase.lib.ListExtensions;

@SuppressWarnings("all")
public class Person implements Comparable<Person> {
    private int age;

    public int getAge() {
        return this.age;
    }

    public void setAge(final int age) {
        this.age = age;
    }

    private final ArrayList<Person> friends = new ArrayList<Person>();

    public ArrayList<Person> getFriends() {
        return this.friends;
    }

    public void setFriends(final ArrayList<Person> friends) {
        this.friends = friends;
    }
}
```
public Person youngestFriendByDistance(final int distance) {
    List<Person> _xifexpression = null;
    if ((distance == 0)) {
        _xifexpression = this.friends;
    } else {
        final Function1<Person, Person> _function = new Function1<
            Person, Person>() {
            @Override
            public Person apply(final Person p) {
                Person _youngestFriendByDistance = p.
                    youngestFriendByDistance((distance - 1));
                List<Person> _asList = Arrays.<Person>asList(p,
                    _youngestFriendByDistance);
                return Collections.<Person>max(_asList);
            }
        };
        _xifexpression = ListExtensions.<Person, Person>map(this.
            friends, _function);
    }
    return Collections.<Person>min(_xifexpression);
}

public int compareTo(final Person p) {
    return (this.age - p.age);
}
Chapter 10

Future plans

According to our schedule of tasks listed earlier in table 8.1, the following are our front-end and back-end development goals:

10.1 Front-end development goals

The project management module and education module will be built in the IDE. Syntax hinting features will be introduced into the IDE editor. Mechanism will be introduced to calculate the score of a related syntax or variable when the user types the first few character of a word into the editor. Chat module will also be embedded into the IDE to facilitate communication between team members. The Android compilation module will be further developed, with remote logging and video recording functionality. Device information and deployment status can also be seen on the IDE. Also, the Android GUI component visualizer will be implemented if schedule is not tight.

The following minor changes will also be made:

• Theme of editor can be set.
• Indicator of user responsible for the code snippet is shown.
• Loading progress is shown when the application is compiled and deployed.
• Project can be imported into the system.
• Template can be used in the project.
• Devices can be selected when the application is deployed.
• Multiple device information and statuses can be shown on the IDE.
• Displaying testing log and testing record from devices.

10.2 Back-end development goals

For the online IDE components, these goals are envisioned:

• Integration of new Translation language to Docker Containers
• Receiving and storing feedback message, screen captures/videos from client
• Live chat server implementation

10.3 Agent app development goals

The following feature would be added to the Agent App:

• Screen capturing with comments editing
• Screen recording
• Sending comments message and screen capture/recordings to developers
• Live chat function between device and IDE web interface users

10.4 Translation language development goals

Since the full pipeline of Java+ to Java compilation has been completed, the first goal is to ensure that the pipeline is correct. Therefore, testing of the pipeline will continue to ensure no corner cases are missed. Meanwhile, the pipeline will be integrated into the IDE such that users can start using Java+ modules in their Java or Android projects.
Chapter 11

Conclusion

In this report, the need for a machine-independent cloud-based IDE and a new programming language has been shown. The IDE and new programming language aim to reduce the barrier of entry to using the IDE and increase developer productivity.

The proposed technology stack and implementation of these components have been described in this report, as well as the current progress of creating fundamental elements of each component. Although difficulties have been encountered when learning how to use the tools to produce the IDE and new programming language, it is expected that these challenges will be overcome by learning how to use these tools and seeking guidance from our advisers who have experience working with these tools.
List of Figures

1.1 The structure of the code parser in a typical IDE comprises of four modules, namely the scanner and parser module, the presentation module, the semantic module and the editing module. The code parser architecture describes how code syntax and errors can be generated and visualized in the GUI code editor. (Extracted from [27]) 7

5.1 Two developers may edit the same file simultaneously by sending an insert command to the server by providing the keyword and position number. 18

5.2 When the second developer sends an insert command without realizing the changes made by the first developer, positioning issue 19

5.3 Java application development work-flow 22
5.5 Android application development work-flow 29
5.6 Translation happens when classes transfer data across the runtime boundaries 29
5.7 The thin-layer design creates a new programming language which has a syntax that only slightly differs from that of Java 29

9.1 Screenshot of login screen page 36
9.2 Screenshot of catalog screen page 37
9.3 Screenshot of code screen page 38
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Comparison of PHP, Go, Python and Ruby for the implementation of back-end processing language</td>
<td>20</td>
</tr>
<tr>
<td>5.2</td>
<td>Comparison of Java source code output and Java bytecode output from the new programming language</td>
<td>26</td>
</tr>
<tr>
<td>5.3</td>
<td>Comparison of LLVM, PyPy, MPS, Graal/Truffle and XText for the implementation of a new programming language</td>
<td>27</td>
</tr>
<tr>
<td>8.1</td>
<td>Schedule of Tasks</td>
<td>33</td>
</tr>
</tbody>
</table>
Listings

5.1 Sample code of Go concurrency and messaging  . . . . . . . . . . 19
5.2 Code to receive messages from program 5.1 . . . . . . . . . . . 19
9.1 Sample of new programming language syntax  . . . . . . . . . . 37
9.2 Java+ code sample  . . . . . . . . . . . . . . . . . . . . . . . 38
9.3 Java output for Java+ code in listing 9.2  . . . . . . . . . . . 39
Acronyms

API  application program interface.  30
AST  abstract syntax tree.  1, 38
CaaS Communication as a Service.  13
EaaS  Education as a Service.  13
GUI  graphical user interface.  30
IDE  Integrated Development Environment.  1, 4–9, 11, 12, 14–17, 21, 23, 24, 30–36, 41–44
JVM  Java Virtual Machine.  9
LaaS  Logging as a Service.  13
OOP  Object-Oriented Programming.  9
OT  operational transformation.  17
PaaS  Platform as a Service.  13
RAD  Rapid Application Development.  9
SaaS  Software As a Service.  7, 13
XaaS  Everything as a Service.  12
Bibliography


[38] JetBrainsTV on Youtube. MPS Projectional Editor. Feb. 26, 2015. URL: https://www.youtube.com/watch?v=iN2Pf1vXUqQ (visited on 01/20/2016).