FIT3077

Software Architecture & Design

Composite

Team Information

Zoe Pei Ee, Low*
31989985
zlow0011@student.monash.edu

Ci Leong, Ong*
31835996
cong0017@student.monash.edu

Guangxing, Zhu*
32597517
gzhu0009@student.monash.edu

*Equal contribution. Listing order is random.

1 The Team: *Composite*

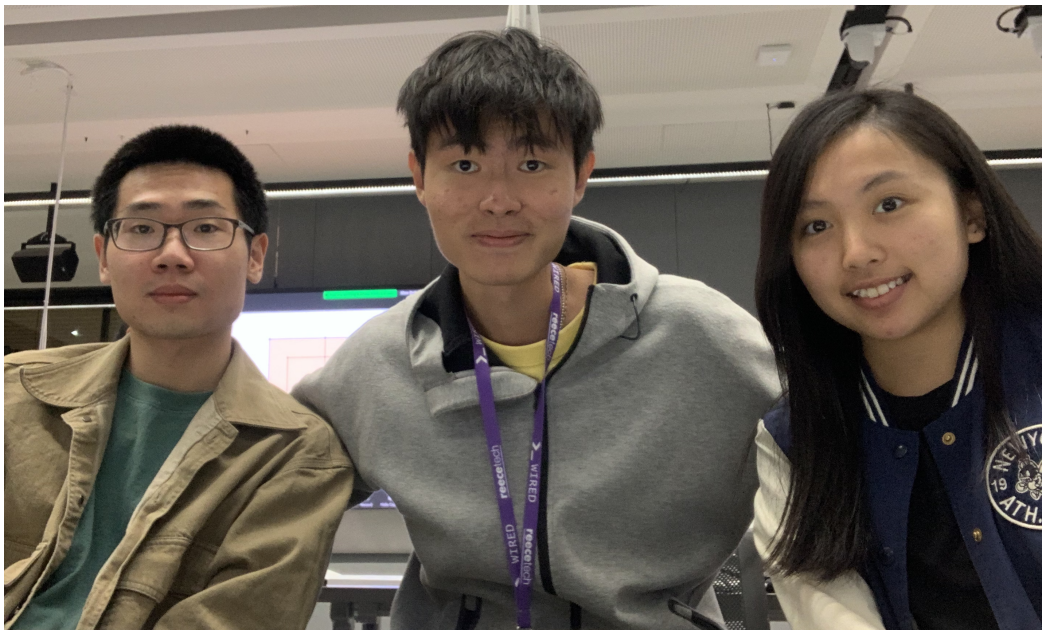

Figure 1: *Team Photo.* Pictured here is our team - *Composite*. We have *Guangxing*, *Ci Leong*, and *Zoe* (left to right).

Inspired by the idea of *composition* in domain modeling, we named the team *Composite*. At the heart of our team lies the principle of working as a collective - we are *interdependent* and *indispensable* to one another. With a strong emphasis on collaboration, we are confident that together as a team, we have the ability to develop the Nine Men's Morris game within the semester.

2 Members

This section presents the key details about each team member, their technical and professional strengths. Additionally, we have included a lesser-known fun fact about each member beyond their work.

Zoe Pei Ee, Low

Nationality: Malaysian

Mobile no.: +61 411 685 250

WhatsApp no.: +60 12 643 8637

WeChat ID: Zoe_0614

Discord Tag: Zoe#6951

Work email: zlow0011@student.monash.edu

Technical strengths:

- Backend and database development
- Object-oriented programming in Python, C# and Java
- Deep learning model research and model construction

Professional strengths:

- Multilingual proficiency (Chinese, English, Malay, Korean, Japanese, etc.)
- Time management skills
- Leadership and project management

Fun fact:

Enjoys watching movies and listening to music.

Ci Leong, Ong

Nationality: Malaysian

Mobile no.: +61 412 012 754

WhatsApp no.: +60 11 2381 9131

WeChat ID: bill_ong

Discord Tag: Bill Ong#6767

Work email: cong0017@student.monash.edu

Technical strengths:

- Deterministic algorithms & data structures
- Deep learning model research and construction, with focus in transformers
- Object-oriented programming in Python, Java and C/C++
- Technical writing in LaTeX

Professional strengths:

- Multilingual proficiency (Chinese, English, Malay)
- Tutoring and mentoring
- Quality assurance

Fun fact:

Will not get fat despite eating and sleeping a lot.

Guangxing, Zhu

Nationality: Chinese

Mobile no.: +61 423 419 589

WhatsApp no.: N/A

WeChat ID: Oren_tripple_z

Discord Tag: N/A

Work email: gzhu0009@student.monash.edu

Technical strengths:

- Object-oriented programming in Java, Python and C/C++

Professional strengths:

- Multilingual proficiency (Chinese, English, Cantonese)
- Time management skills
- Self-learning skills

Fun fact:

Likes to sleep and daydream, but usually learns at a leisure pace.

3 Schedule

Creating a viable meeting schedule is a non-trivial task for our team, as all of us are enrolled in a full-load academic semester with hectic schedules. Additionally, we are currently working part-time with flexible schedules which further complicates the problem. We have managed to finalize the schedule for sprints 1 and 2, but do take note that the schedule for sprint 3 onwards may need to be adjusted.

The team will conduct weekly meetings on Monday 5pm - 6pm, before the unit workshop to discuss the project. This arrangement accommodates Guangxing's travel from the Melbourne CBD to the Clayton campus.

As we are utilising a distributed version control system (VCS), *git*, for the project, we are able to work asynchronously during our free time. The work schedule below is finalised during our week 4 meeting.

Team Member	**Day**	**Time**
Zoe	Tuesday	10am - 3pm
	Friday	7pm - 11pm
	Saturday	10am - 5pm
Ci Leong	Monday	10am - 5pm
	Thursday	4pm - 8pm
	Saturday	11am - 2pm
Guangxing	Wednesday	11am - 3pm
	Friday	10am - 2pm
	Sunday	6pm - 11pm

Table 1: *Work schedule.*

4 Technology Stack

In designing software, it is of importance to pick a technology stack that maps naturally to the problem domain, which for our case is the domain model, rather than rushing to a conclusion based on familiarity. However, considering our tight deadline to deliver the first MVP by week 11, we must at least opt for a stack we had some experience in. The goal is to find a balance between suitability and efficiency in the development process.

Component	**Selection**
Language	Java
IDE	IntelliJ IDEA Ultimate
GUI	JavaFX
Testing	JUnit & Mockito
Build Tool	Gradle
Dependency Management	
Documentation	Javadoc
VCS	Git (hosted on GitLab)

Table 2: *Technology stack*. This is the technology stack that we will be utilising for developing the Nine Men's Morris game.

4.1 Language

The team possess extensive programming experience in Python from taking the core units of Computer Science and some object-oriented game development experience from the prerequisite unit, FIT2099. After careful consideration, we decided on using Java as our language of choice for developing the Nine Men's Morris game.

We opted for Java over Python because, although Python allows for rapid prototyping thanks to its flexibility of the gradual typing system, it relies heavily on error handling at runtime and allows poor design choices to go unpunished. This is due to its "Easier to Ask Forgiveness than Permission" (EAFP) design philosophy. On the contrary, Java's static typing system verifies the variable types and method signature, facilitating informed decision-making during the development phase by providing type-informed code completion as a result of its "Look Before You Leap" (LBYL) design philosophy.

While some may argue that Python has robust library support, they must also recognise that Java also provides an extensive range of libraries and frameworks, which we will explore later.

Given our project takes on the object-oriented approach, Java's typing system will allow us to more effectively represent our design decisions and optimise the development workflow.

4.2 Integrated Development Environment (IDE)

There are two popular IDEs for available for developing Java applications: IntelliJ IDEA and Visual Studio Code, each backed by a strong following.

While Visual Studio Code is technically a text editor, it can be transformed into a fully-fledged IDE with the Java Extensions Pack available in the Marketplace, with features including code completion, code navigation, debugging, and testing.

However, JetBrains' IntelliJ IDEA offers a more streamlined experience specifically designed for Java application development. Unlike the former, it is designed specifically for Java development, making it easier to set up, with features such as advanced code completion, automated code refactoring, and faster code indexing. If that is not convincing enough, as students, we have complimentary access to the paid version, IntelliJ IDEA Ultimate, which offers support for enterprise frameworks and more.

Therefore, we have decided to use IntelliJ IDEA Ultimate for developing the Nine Men's Morris game.

4.3 Graphical User Interface (GUI) Framework

The domain model we developed maps naturally to the MVC architecture, and the GUI framework that best fits this architecture and our needs is JavaFX. There is really no competition.

While Swing is more lightweight and has been widely adopted for developing GUIs and sports a much larger and stronger following, the framework tends to encourage the procedural paradigm, rather than leveraging the modularity of the MVC architecture.

Conversely, JavaFX adopts a modern approach that emulates the MVC architecture. It enables developers to develop visually appealing GUIs with the Scene Builder available in IntelliJ IDEA using an extensive set of UI components and CSS-styling, of which Zoe has prior experience in. Additionally, JavaFX offers resource management optimisations that are important to us in ensuring a smooth gameplay experience.

4.4 Testing Framework

In Test-Driven Development (TDD), we create unit tests before writing the code and verify the correctness of the written code with these tests. This is a crucial aspect of the software development lifecycle as it ensures the *units* in the application produces the expected behaviour.

To steamline the testing, we will be using the JUnit and Mockito frameworks together. JUnit will be used for creating and executing the test cases with annotations and assertions; Mockito will be used to generate mock objects for unit testing purposes.

4.5 Build Tool / Dependency Management Tool

In today's software development landscape, it is pretty much mandatory to employ tools for building applications and managing dependencies. This is because dependency management can be extremely time-consuming if handled poorly. Resolving dependency version conflicts manually could easily take up a huge portion of the little time we have for developing the Nine Men's Morris game. Moreover, they also streamline the creation of the distributable application - the Java archive (*.jar) files executable on any machine equipped with the Java Runtime Environment (JRE). To quote Java's catchphrase, "Write once, run anywhere"!

There are 2 viable options to consider: the classic Maven and the more recent Gradle, which is recently gaining traction, both of which will accomplish the same goal.

Maven relies on an XML file for configuration and customisation and has a much steeper learning curve. Gradle, on the other hand, utilizes a Groovy-based DSL that provides a readable modern, concise syntax.

On the contrary, Maven has a more comprehensive plugin ecosystem due to its longer presence in the industry. However, as we will not be leveraging Maven's advanced features, this benefit is irrelevant to us.

Building on the above, Ci Leong and Zoe have prior experience using Gradle in Android application development. Therefore, we will be using Gradle as our preferred tool for developing our Nine Men's Morris game.

4.6 Documentation

Documentation is a necessary part of software engineering, as it helps in understanding the code for both the developers themselves and others reading their code. A well-written documentation could save programmers invaluable time when navigating complex logic.

For documenting our project, we are considering 2 options. The first option is Sphinx, a cross-language documentation framework used mostly in Python projects. Sphinx outputs documentation in multiple formats, including but not limited to HTML, ePub, and PDF. The second option is Javadoc, a built-in Java documentation tool that generates documentation from formatted comments in the code, and outputs documentation in HTML.

It is undeniable that Sphinx offers a more comprehensive set of features. However, it does introduce an additional dependency overhead. In our context, documentation is to support asynchronous development, of which we need to understand each other's code and avoid meaningless meetings. The feature set that Javadoc provides already serves our purposes and IntelliJ IDEA has built-in support for it.

Therefore, we will be using Javadoc to create documentation for our Nine Men's Morris game.

4.7 Version Control System (VCS)

We will be using Git for tracking changes across code versions, and the Git repository will be hosted on GitLab.

Though there are *still* multiple VCS options in the market, including Git, Mercurial, and Subversion, Git has totally dominated the field despite its leaky abstraction design. This is due to its speed, reliability, flexibility and the developers' conviction on keeping it open-source. Teams using Subversion are also migrating to Git. Therefore, we will be utilizing Git for version control.

To achieve asynchronous development, we need a remote repository to host our code. Monash University hosts an on-premise GitLab server. We will be pushing our Git repository to this server to match the teaching team's request.