Group 10 - Deliverable #2

SFWRENG 3A04: Software Design III - Large System Design

Andrew Hum 400138826
Arkin Modi 400142497
Hongzhao Tan 400136957
Christopher Vishnu 400129743
Shengchen Zhou 400050783

March 5, 2020

1 Introduction

1.1 Purpose

The purpose of the document is to focus on the architecture of the HackerSim system. The system's architecture is based upon business events developed in Deliverable 1 to outline the components of the HackerSim software for both the client and the developer. It covers the architectural decisions that have been made regarding the system and its components. This document is intended for the project manager, the current project team and any future development teams for the HackerSim Project.

1.2 System Description

The HackerSim system is an interactive game that will allow the user to raise a Software Engineer in their room. The main component of our software would be the General Room which is the link that interacts with the rest of the sub-components. The main sub-components that the General Room interacts with which would be the Shop, Friends and Chat, Project and the Time-step. The Shop component focuses on interacting with the inventory for purchasing and browsing items. The Friends and Chat component focuses on providing message functionality between the user and the friends. The Project component focuses on the project and future projects the Software Engineer has to do to gain in-game currency. Finally, the Time-step component focuses on the passage of time and which affects the Software Engineer's attributes.

1.3 Overview

This document is organized by the following sections: Analysis Class Diagram, Architectural Design, Class Responsibility Collaboration Cards. Analysis Class Diagram focuses on providing details about the structure of the classes and their relationships. Architectural Design focuses on the overall architectural design of the HackerSim application, showing the division of the system into subsystems. Finally, Class Responsibility Collaboration (CRC) Cards focus on each individual class and its responsibilities and relations in which they collaborate with other classes.

1.4 Definitions, Acronyms, Abbreviations

SE - Software Engineer

2 Analysis Class Diagram

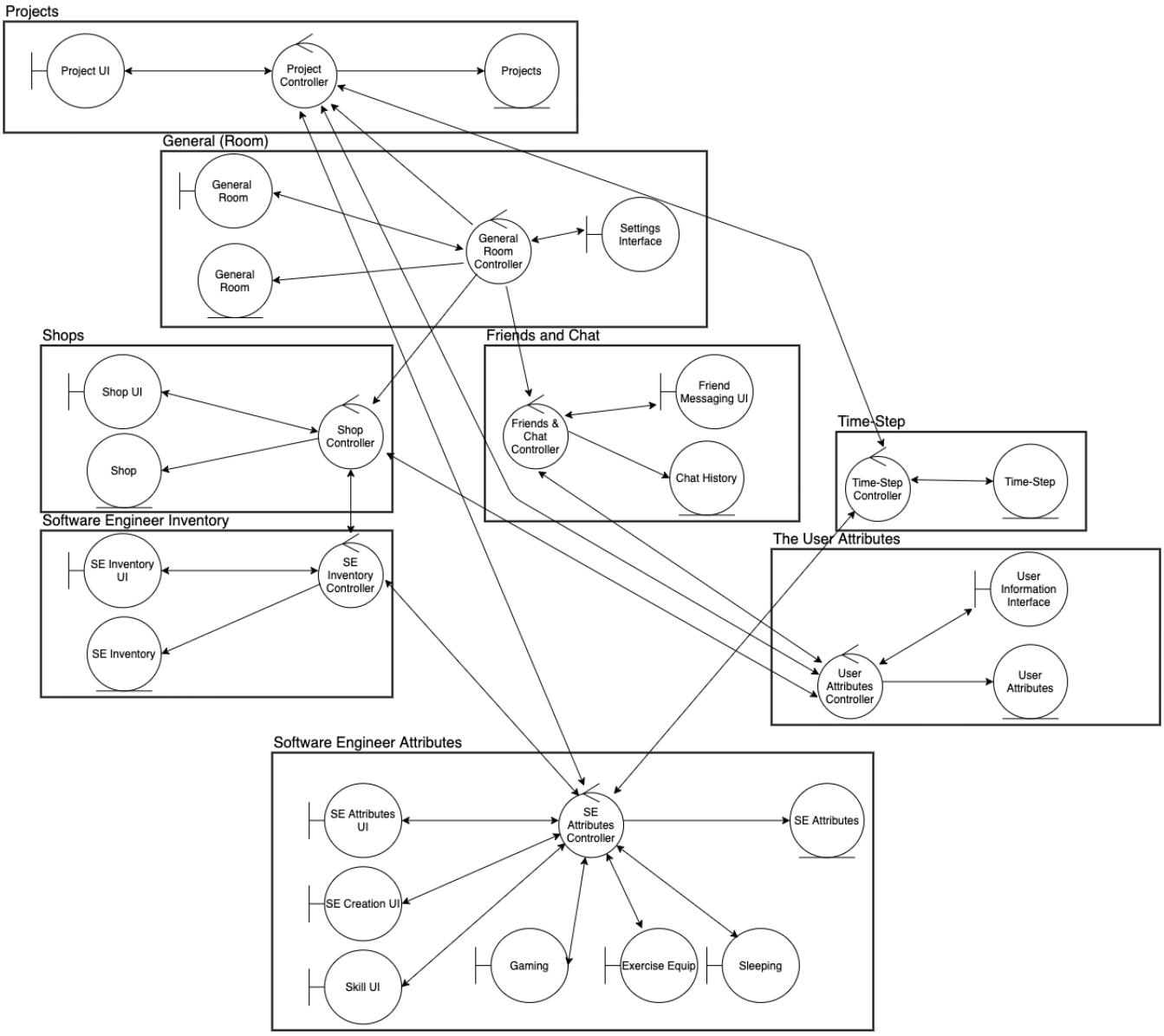

Figure 1: Analysis Class Diagram for Hacker Sim

3 Architectural Design

3.1 System Architecture

The architecture style that we will be using for our system is an architecture that is interaction-oriented; Presentation-Abstraction-Control (PAC). Among the other architectural styles, data-centric and data-flow architecture, we felt that an interaction-oriented architecture would fit our project the best. We believe this because this type of architecture is most appropriate when there is a need to separate the user's interactions from the data abstraction and data processing. This type of architecture effectively divides the data into logical sections and allows an effective presentation of content based on changes in the data utilizing the view modules. From interaction-oriented architecture, there were two possible routes to take, Model-View-

Controller (MVC) architecture and Presentation-Abstraction-Control (PAC) architecture. We chose to use Presentation-Abstraction-Control as it emphasizes a hierarchical architecture that allows low-coupling of agents protecting other agents from being modified by the changes of one. In essence, this will provide us with an effective way to identify our core entities and link them to relevant agents with ease. It will also allow multiple agents to perform their tasks, and update their view accordingly without affecting the other agents. MVC highly depends upon a uniform data set for all views that changes at a constant rate. Given that we have several different entities, each with their unique data set that will be updated at different rates, the need for PAC architecture for our system is evident.

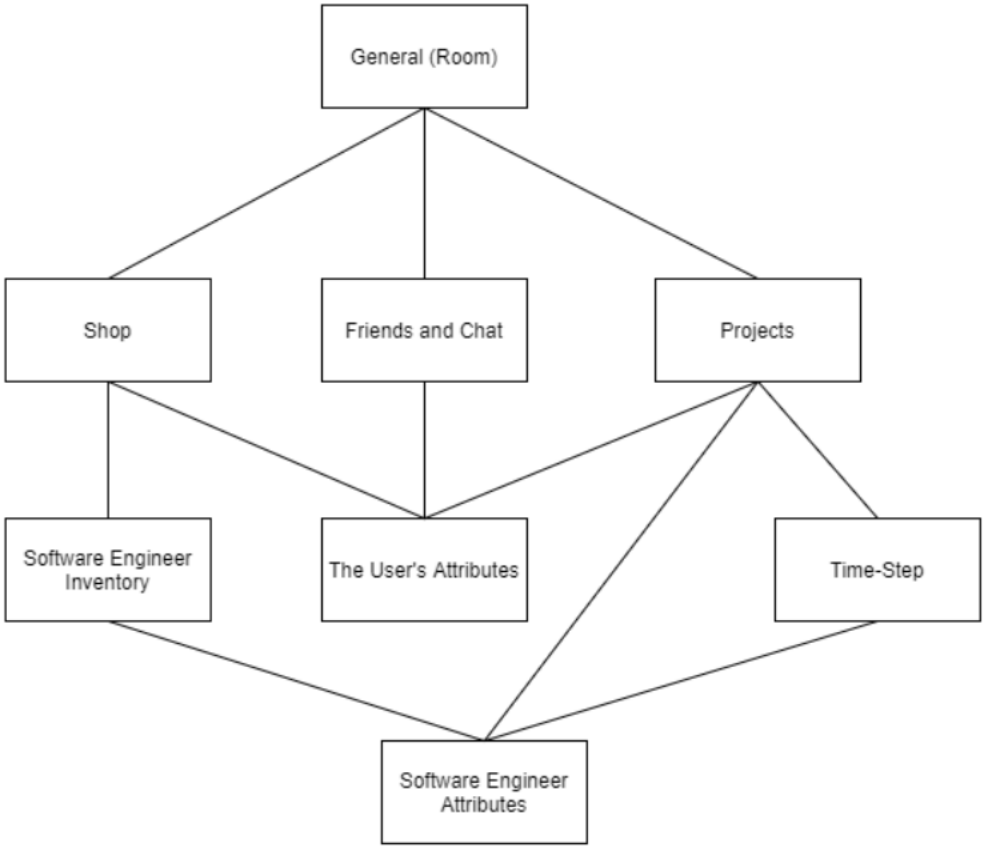

Figure 2: PAC Structural Architecture Diagram

3.2 Subsystems

The following are explanations of the subsystems for our project that outline their purpose and relationship to other subsystems.

1. General (Room)
   Purpose: This agent is the highest-level agent that controls every other agent and presents the default 'room' view as well as ability to interact with the objects, shop, friends and the SE
   Direct Relationships: Shop, Friends and Chat, Projects

2. Shop
   Purpose: This agent is an intermediate-level agent that controls the user's interactions with the shop, including purchasing and browsing items. This agent may then update the SE's inventory if the user chooses to purchase an item.
   Direct Relationships: SE Inventory, User Attributes, General (Room)

3. Friends & Chat
   Purpose: This agent is an intermediate-level agent that stores data on chat history, and provides the user with the ability to message friends based on the user's friends list.
   Direct Relationships: User's Attributes, General (Room)

4. Projects
   Purpose: This intermediate-level agent stores information regarding current and future projects the SE may work on to gain currency.
   Direct Relationships: User's Attributes, SE Attributes, General (Room), Time-Step

5. Time-Step
   Purpose: This intermediate-level agent controls the passage of time, and signals the SE Attributes accordingly with changes in their current state or information. These changes include hunger, project completion, tiredness level, happiness change, etc.
   Direct Relationships: SE Attributes, Projects

6. Software Engineer (SE) Inventory
   Purpose: This bottom-level agent stores data based on the items in the SE's inventory and adjusts the data and view based upon the user's interactions with the store or through the use of items.
   Direct Relationships: Shop, SE Attributes

7. Software Engineer (SE) Attributes
   Purpose: This bottom-level agent stores data based on the SE's current attributes such as their happiness index, how tired they are, if they are hungry, if they completed a project, etc. These attributes may be adjusted by the Time-Step agent or the Project agent, or changed directly by the user.
   Direct Relationships: Projects, Time-Step, SE Inventory

8. User Attributes
   Purpose: This bottom-level agent stores data based on the user's current attributes. Some data is immutable and may not be changed throughout the game, however, there is data that may be manipulated by the project controller, providing the user with skill points to increase the SE's ability, a change in their current balance, and addition or removal or friends.
   Direct Relationships: Shop, Friends and Chat, Projects

4 Class Responsibility Collaboration (CRC) Cards

This section should contain all of your CRC cards.

a) Provide a CRC Card for each identified class

b) Please use the format outlined in tutorial, i.e.,

Class Name:	
Responsibility:	**Collaborators:**

A Division of Labour

Include a Division of Labour sheet which indicates the contributions of each team member. This sheet must be signed by all team members.

IMPORTANT NOTES

- Please document any non-standard notations that you may have used
  - *Rule of Thumb*: if you feel there is any doubt surrounding the meaning of your notations, document them
- Some diagrams may be difficult to fit into one page
  - It is OK if the text is small but please ensure that it is readable when printed
  - If you need to break a diagram onto multiple pages, please adopt a system of doing so and thoroughly explain how it can be reconnected from one page to the next; if you are unsure about this, please ask about it
- Please submit the latest version of Deliverable 1 with Deliverable 2
  - It does not have to be a freshly printed version; the latest marked version is OK
- If you do NOT have a Division of Labour sheet, your deliverable will NOT be marked