School of Computer Science

COMP20050: Software Engineering Project-II

Week2 Laboratory Practical

Module Coordinator	Ravi Reddy Manumachu
Module TA	Maryam Gillani

HexOust Software Architectural Design 2
Interaction Modelling 2
Structural Modelling 3
Architectural Pattern 4
Project Plan 5

HexOust Software Architectural Design

In the lab sessions this week, you will continue the software architectural design of your project.

You must first understand **how to model** your system and what tools to use. There are several modelling tools that you can employ for this purpose (UML tools, draw.io, Lucidchart, and Magicdraw).

Different models represent the system from different perspectives. I would like you to focus on two perspectives.

An **interaction perspective** where you model the interactions between a system and its environment or between the components of a system.
A **structural perspective** where you model the organization of a system or the structure of the data that is processed by the system.

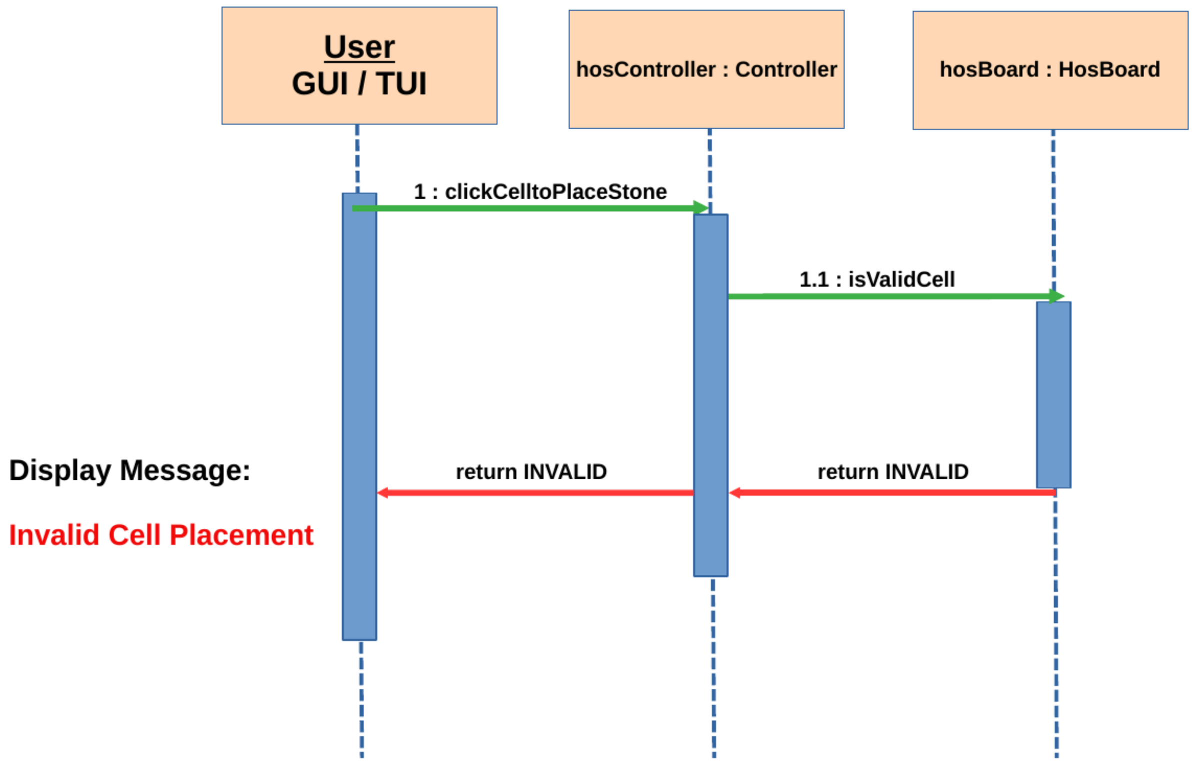
Interaction Modelling

User interaction with the system and interaction between system components can be modelled using two approaches:

Use Case Modelling: Used to model interactions between a system and users.
Sequence diagrams: Model interactions between system components.

UML Sequence diagrams are primarily used to model the interactions between the actors and the objects in a system, and the interactions between the objects.

The following is a UML sequence diagram to address the requirement SR4.1 (Lecture 2, Project Description).

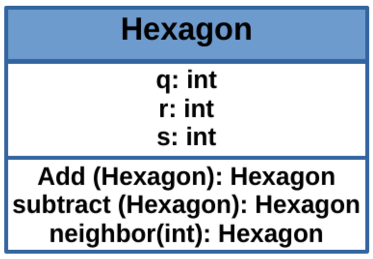


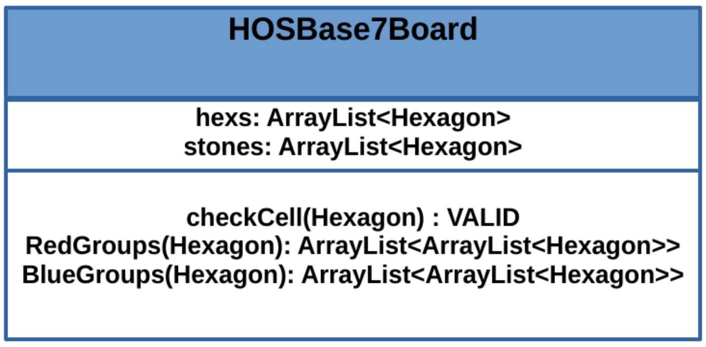
Structural Modelling

Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.

UML class diagrams can be used for modeling the static structure of the object classes in a software system.

The following are class diagrams for some components in an example HexOust software system:




HOSBase7Board
hexs: ArrayList<Hexagon>
stones: ArrayList<Hexagon>
checkCell(Hexagon) : VALID
RedGroups(Hexagon): ArrayList<ArrayList<Hexagon>>
BlueGroups(Hexagon): ArrayList<ArrayList<Hexagon>>

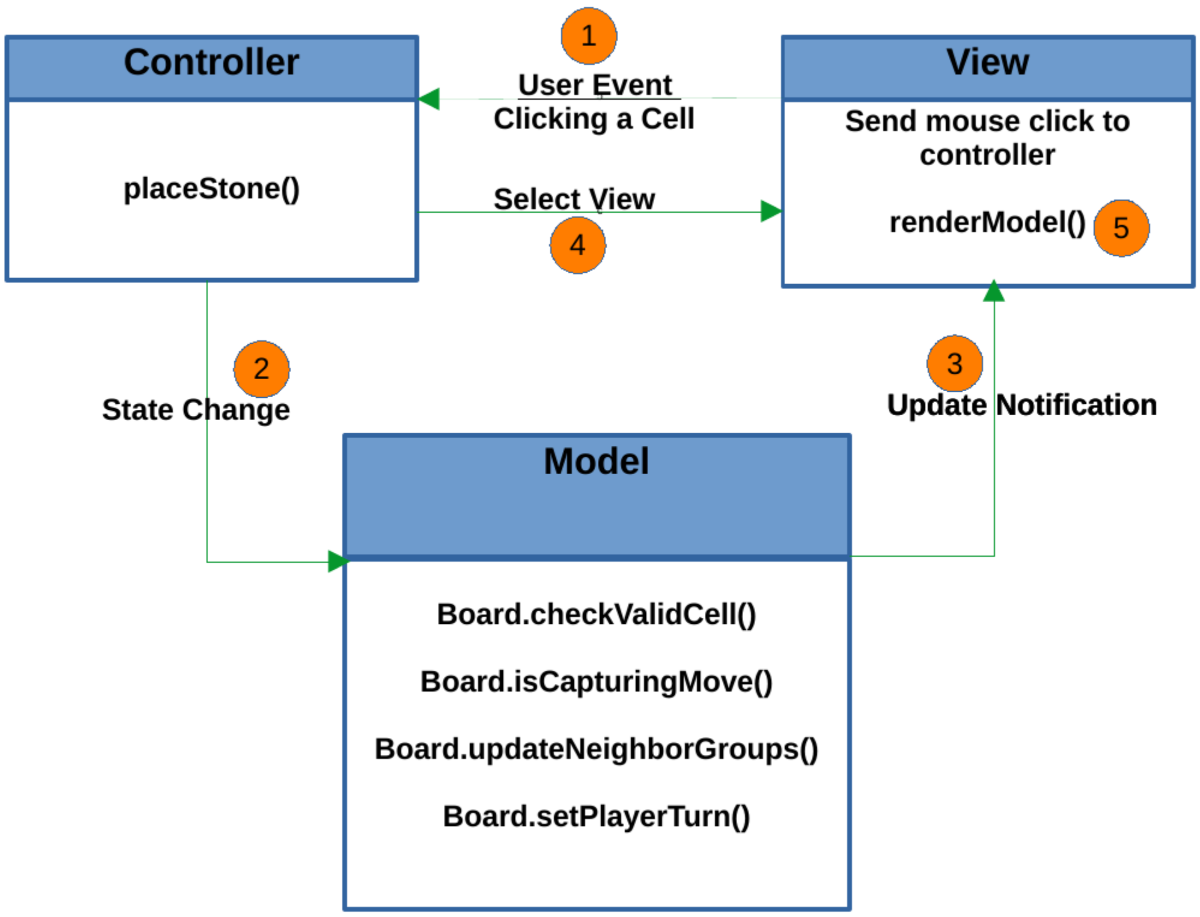
Architectural Pattern

An **architectural pattern** is a reusable, proven solution to a specific, recurring problem, which can be applied within various architectural styles.

Patterns are more specific, often detailing interactions, components, responsibilities, and relationships.

You can use **Model-View-Controller (MVC)** or a **layered** or a **modular** pattern. A layered pattern will have your system organized into a hierarchy of layers. A modular pattern will have modules with well-defined software programming interfaces.

The following is a requirement described through the MVC pattern for a sample HexOust software system:



Project Plan

Using your software architectural design as a basis, create your project plan, which will have four sprints of features to completely implement the project.