Choosing The Best Design Pattern For *Where In The Universe Is Classroom 205?’*s Data Model

Author: Evelyn Livant

*Where In The Universe is Classroom 205?* is a game written using the Unity IDE which engages the player in solving a series of puzzles as to reach the final goal. This paper evaluates and analyzes various design patterns to determine the best design pattern for my part of the coding of this project.

A close up of a logo

Description generated with high confidenceThe scope of this task is handling the data for a userid and current game level written to and read from a binary file. The data must be persistent through all levels of gameplay including the main menu system. To begin assessing which design pattern would best suit this requirement, I drew a UML diagram. An activity diagram seemed the best suited to describe the behavior of this part of the system because it would illustrate the data flow throughout the system and better illustrate how this can be incorporated into the system.

Based on this design and how Unity controls flow of communication between objects, I propose that the mediator design pattern is best suited to this project. The mediator pattern is a behavioral pattern originally described in *Design Patterns: Elements of Reusable Object-Oriented Software* 1 that defines and object that encapsulates how a set of objects interacts. "With the mediator pattern, communication between objects is encapsulated within a mediator object. Objects no longer communicate directly with each other, but instead communicate through the mediator. This reduces the dependencies between communicating objects, thereby reducing coupling."2 Although this design pattern was initially published in 1994, it remains relevant. The authors have been accepted as experts in the field and the original text has been established as a standard for software design.

In general, this design pattern is useful in different scenarios such as an asynchronous system where there are delays between different components communicating. The mediator would prevent system faults or failures because of lag or degradation due to different communication times. This structure is also useful during periods of system refactoring. Developers could rewrite large sections of code without breaking the entire system.

In the case of our project, there are a few events that trigger the flow of data between game objects including menus and levels indicating need for communication between objects. The other components of the system need the data available and persistent. In addition to efficient facilitation of communication, the mediator pattern will prevent “bad smells” in this system3 by eliminating tight coupling between objects and allowing independent change between a set of objects.4 This works well with Unity because the architects recommend creating an object that can be serialized to read and write data and the object can be used to communicate that data with other game components.5

This also works well for our team because we are working remotely most of the time and on different schedules. This pattern also increases flexibility of communication between objects and provides the flexibility necessary for future change. A test-driven development method could be easily developed and implemented from this design. This would lend itself to an agile, iterative development process best suited to the requirements of this course and to future development. More than likely, the Unity architects designed the IDE with this pattern in mind to solve these design and development problems.

Based on my activity diagram, research, and analysis of our project; I recommend use of the mediator design pattern because it creates a decoupled communication flow between objects, facilitates asynchronous development, and seamlessly integrates with our chosen IDE.

**Notes**

1 Gamma et al., Design Patterns: Elements of Reusable Object-Oriented Software. 1996.

2 “Mediator Pattern.” *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation, Inc. 4 Dec. 2017, https://en.wikipedia.org/wiki/Mediator\_pattern

3 Co-Occurrence of Design Patterns and Bad Smells in Software Systems: An Exploratory Study

4 “Mediator Pattern.” *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation, Inc. 4 Dec. 2017, <https://en.wikipedia.org/wiki/Mediator_pattern>

5 “Persistence – Saving and Loading Data.” *Unity 3D.*Unity Technologies, 4 Dec. 2017, https://unity3d.com/learn/tutorials/topics/scripting/persistence-saving-and-loading-data

**Works Cited**

Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software.* 1996.

“Mediator Pattern.” (*Wikipedia: The Free Encyclopedia.* Wikimedia Foundation, Inc. 4 Dec. 2017).

Bruno Cardoso, and Eduardo Figueiredo. “Co-Occurrence of Design Patterns and Bad Smells in Software Systems: An Exploratory Study.” *Proceedings of the annual conference on Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective - Volume 1.* May 2015. https://dl-acm-org.uri.idm.oclc.org/results.cfm?query=(%252Bdesign%20%252Bpattern)&within=owners.owner=HOSTED&filtered=&dte=&bfr=

 Geig, Mike. “Persistence – Saving and Loading Data.” *Unity 3D.*Unity Technologies, 4 Dec. 2017. https://unity3d.com/learn/tutorials/topics/scripting/persistence-saving-and-loading-data