Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A31

Game C/S Model – Collaboration Diagram

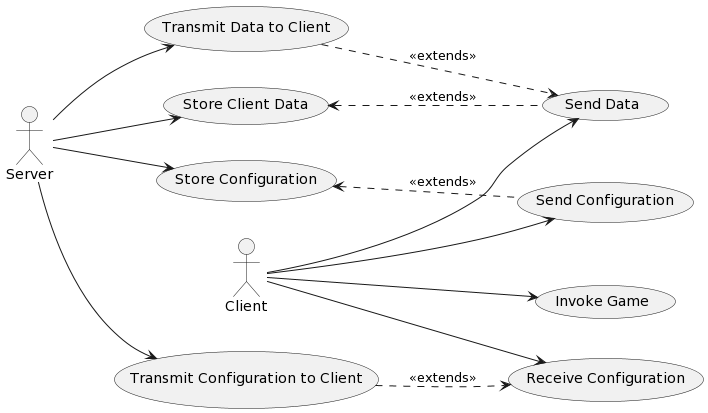
Team:

[Meet Kotadiya] - Id: [041070018] / [Dhanush Bajaj] - Id: [041067058]

TM Model C/S Proposal

|  |  |
| --- | --- |
| **Part**  **1** | **C/S Architecture** |

* 1. **UC Model (1pt)**

****

**Fig. 1** – C/S Model[[1]](#footnote-1)

**Actors table**

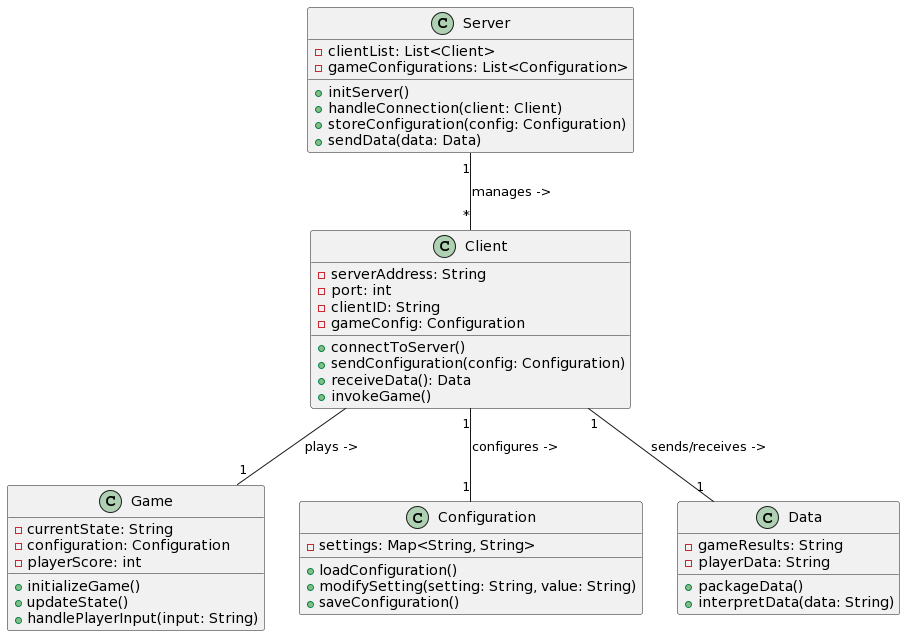
|  |  |
| --- | --- |
| **Actors** |  |
| Client | Represents the user or system that interacts with the server to play the game.  - Sends game configurations to the server.  - Receives game data.  - Invokes game logic. |
| Server | Manages the game configurations and client information.  - Stores and manages game configurations.  - Sends configurations to clients.  - Handles client connections. |

**UC table** (example):

|  |  |
| --- | --- |
| **Use Cases** |  |
| Send Configuration | The client sends game configuration data to the server. |
| Receive Configuration | The client receives configuration data from the server. |
| Send Data | The client sends data (game results, player data) to the server. |
| Store Configuration | The server stores configuration data for the game. |
| Store Client Data | The server maintains basic information about each client. |
| Transmit Configuration to Client | The server sends game configuration data to the client. |
| Transmit Data to Client | The server sends relevant data back to the client (could include game status updates or responses). |
| Invoke Game | The client starts the game using the received configuration. |

* 1. **ClassD (2pt)**

**Class Diagram**



**Fig. 3** – ClassD[[2]](#footnote-2)

**Class table**

|  |  |
| --- | --- |
| **Class name** | **Server** |
| Inner Fields[[3]](#footnote-3) | - clientList: List<Client>  - gameConfigurations: List<Configuration> |
| Relationships[[4]](#footnote-4) | - Manages connections with multiple Clients  - Stores and retrieves Configurations |
| Methods | - initServer()  - handleConnection(client: Client)  - storeConfiguration(config: Configuration)  - sendData(data: Data) |

|  |  |
| --- | --- |
| **Class name** | **Client** |
| Inner Fields[[5]](#footnote-5) | - serverAddress: String  - port: int  - clientID: String  - gameConfig: Configuration |
| Relationships[[6]](#footnote-6) | - Connects to Server  - Interacts with Game for gameplay |
| Methods | - connectToServer()  - sendConfiguration(config: Configuration)  - receiveData(): Data  - invokeGame() |

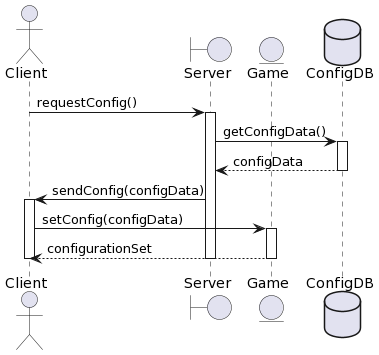
|  |  |
| --- | --- |
| **Class name** | **Game** |
| Inner Fields[[7]](#footnote-7) | - currentState: String  - configuration: Configuration  - playerScore: int |
| Relationships[[8]](#footnote-8) | - Utilizes Configuration for game setup  - Interacts with Client for game inputs and state updates |
| Methods | - initializeGame()  - updateState()  - handlePlayerInput(input: String) |

|  |  |
| --- | --- |
| **Class name** | **Configuration** |
| Inner Fields[[9]](#footnote-9) | - settings: Map<String, String> |
| Relationships[[10]](#footnote-10) | - Used by Server for storing configurations  - Used by Client and Game for game setup and logic |
| Methods | - loadConfiguration()  - modifySetting(setting: String, value: String)  - saveConfiguration() |

|  |  |
| --- | --- |
| **Class name** | **Data** |
| Inner Fields[[11]](#footnote-11) | - gameResults: String  - playerData: String |
| Relationships[[12]](#footnote-12) | - Sent between Client and Server for data transmission |
| Methods | - packageData()  - interpretData(data: String) |

*Create tables for all classes.*

* 1. **UML Collaboration Diagram**



**Fig. 4** – Collab model[[13]](#footnote-13)

**References**

“Client-Server Architecture - Design Your Software Architecture Using...,” OpenClassrooms. [Online]. Available: https://openclassrooms.com/en/courses/5564316-design-your-software-architecture/5578251-client-server-architecture. [Accessed: Nov. 23, 2023].

“Software Architecture Diagramming and Patterns - Educative,” Educative. [Online]. Available: https://www.educative.io/courses/software-architecture-diagramming-patterns. [Accessed: Nov. 23, 2023].

“Modeling a Client/Server System using UML Packages and Deployment Diagram,” Visual-Paradigm Guides. [Online]. Available: https://guides.visual-paradigm.com/uml-deployment-diagram/modeling-client-server-system-using-uml-packages-and-deployment-diagram/. [Accessed: Nov. 23, 2023].

“UML Communication Diagram - Client Server Access,” ConceptDraw. [Online]. Available: https://www.conceptdraw.com/How-To-Guide/uml-communication-diagram. [Accessed: Nov. 23, 2023].

“Client-Server Architecture Diagram | Creately,” Creately. [Online]. Available: https://creately.com/diagram/example/hrlzxtcy2/client-server-architecture-diagram. [Accessed: Nov. 23, 2023].

ChatGPT

Algonquin College

Fall, 2023

1. See <https://www.researchgate.net/figure/System-Architecture-Use-Case-Diagram-Client-side-Functionality_fig2_318502492>. [↑](#footnote-ref-1)
2. See <https://www.codeproject.com/Articles/443660/Building-a-basic-HTML-client-server-application>. [↑](#footnote-ref-2)
3. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-3)
4. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-4)
5. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-5)
6. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-6)
7. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-7)
8. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-8)
9. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-9)
10. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-10)
11. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-11)
12. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-12)
13. See <https://www.researchgate.net/figure/Collaboration-diagram_fig1_268362338>. [↑](#footnote-ref-13)