Module Interface Specification for SE 4G06
An AI-based Approach to Designing Board Games

Team #6, Board Gamers
Ilao Michael, ilaom
Bedi Hargun, bedih
Dang Jeffrey, dangj12
Ada Jonah, karaatan
Mai Tianzheng, mait6

April 6, 2023

1 Revision History

Date	Version	Notes
April 3rd	1.0	Split up Rev 0 into 3 documents and implemented feedback

2 Symbols, Abbreviations and Acronyms

2.1 Abbreviations and Acronyms

symbol	description
SRS	Software Requirements Specification
AI	Artificial Intelligence
A	Assumption
LC	Likely Change
FR	Functional Requirement
NFR	Non Functional Requirement
FSM	Finite State Machine
TA	Teaching Assistant

Contents

List of Tables

3 Introduction

The following document details the Module Interface Specifications for An AI-based Approach to Designing Board Games, a system that simulates thousands of board game simulations using AI to visualize winning strategies for game designers to help balance their game.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found here, under MG and DES.

See SRS Documentation here, for full list of requirements.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by SE 4G06.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
policy	Policy	A matrix of size N-by-N, where N is the game state size
string	String	A series of characters.
game state	GameState	A matrix of size N-by-N, where N is the game state size
simulation	Simulation	A JSON Object that has another JSON object with the following keys: player, $\text{turn}_n um, action, action_d etails, meta_d ata$.
actions	Actions	A JSON Object that has a string as the key and null as the value. The string is the action name.

The specification of SE 4G06 uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of

characters. Tuples contain a list of values, potentially of different types. In addition, SE 4G06 uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Module Type	**Module Name**	**Module Description**
AI	AI Agent Module	Trains AI Agents on the game and generates a policy
AI	Game Environment Module	Receives input from AI Agents to take action on the game
GE	Actions (Commands) Module	Describes the possible game moves that the AI Agents are able to take
GE	Game Loop Module	Continues the game loop for the game and checks if the game-over condition has been fulfilled or not.
DV	JSON Module	Responsible for recording each AI Agents moves and observation space and putting them into a JSON file.
DV	Graph Module	Produce a graph selected by the user.
DV	JSON Data Parser Module	Parses JSON files with AI Agents' move history

Table 1: Module Hierarchy

6 MIS of AI Agent Module

6.1 Module

AIAgent

6.2 Uses

Game Environment Module 7

6.3 Syntax

6.3.1 Exported Types

Policy = ?

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
run	$\mathbb{N}$ $\mathbb{N}$ $\mathbb{N}$ $\mathbb{N}$ String	Policy	PathDoesNotExist

6.4 Semantics

6.4.1 State Variables

GameEnvironment: GameEnvironment

6.4.2 Environment Variables

Device: The CPU or GPU the system will use to train the AI.

6.4.3 Assumptions

The constructor of GameEnvironment is called before the access routine is called.

6.4.4 Access Routine Semantics

run(training-num, test-num, n-step, epoch, resume-path):

- output: out:= Policy
- exception: $exc := \neg$ resume-path $\Rightarrow$ PathDoesNotExist

7 MIS of Game Environment Module

7.1 Module

GameEnvironment

7.2 Uses

Game Loop 9

7.3 Syntax

7.3.1 Exported Types

GameEnvironment = ?

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
GameEnvironment		GameLog	
observe	$\mathbb{N}$	Sequence of $\mathbb{N}$, GameState	AgentOutOfBounds
legalMoves	$\mathbb{N}$	Sequence of $\mathbb{N}$	AgentOutOfBounds
step	$\mathbb{N}$	$\mathbb{N}$	ActionOutOfBounds, AgentOutOfBounds

7.4 Semantics

7.4.1 State Variables

player: Player
engine: Game
Rewards: Sequence of $\mathbb{R}$
CurrentAgent: $\mathbb{N}$

7.4.2 Environment Variables

None

7.4.3 Assumptions

7.4.4 Access Routine Semantics

GameEnvironment():

- transition:=
  Agents:=GameLoop.GetAgents()
  Rewards:= Sequence of 0 of len(Agents)
- output: out:= GameLog
- exception: exc:= len(Agents) ¬ =len(Rewards) ⇒ RewardOutOfBounds

observe(agent):

- output: out:= GameLoop.getActionMask(agent), GameLoop.getGameState(agent)
- exception: exc:= agent > len(Agents) ⇒ AgentOutOfBounds

legalMoves(agent):

- output: out:= GameLoop.getActionMask(agent)
- exception: exc:= agent > len(Agents) ⇒ AgentOutOfBounds

step(action):

- transition:=
  GameLoop.playTurn(action, CurrentAgent)
  CurrentAgent := NextAgent()
  Rewards[CurrentAgent] := GameLoop.getReward(CurrentAgent)
- output: out:= GameLoop.isGameOver()
- exception: exc:=
  CurrentAgent > len(Agents) ⇒ AgentOutOfBounds
  action > len(GameLoop.getActionMask(agent)) ⇒ ActionOutOfBounds

7.4.5 Local Functions

NextAgent():

- output: out:= (CurrentAgent + 1) mod len(Agents)

8 MIS of Actions (Commands) Module

8.1 Module

Command

8.2 Uses

Game Loop 9

8.3 Syntax

8.3.1 Exported Types

Command = ?

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
Command	Player, Game Engine	Action	
execute			
check		Boolean	

8.4 Semantics

8.4.1 State Variables

player: Player
engine: GameLoop
action: String
action_details: String

8.4.2 Environment Variables

None

8.4.3 Assumptions

There can be more state variables in the child classes based on the complexity of the action. However, this MIS describes the parent Command module.

8.4.4 Access Routine Semantics

Command(player, engine):

- transition :=
  player := player
  engine := engine
- output: out:= Action
- exception: N/A

execute():

- transition: := engine := engine_new_state
  The transition will be a new state to engine based on the action so only defined semi-formally here as general as possible
- output: N/A
- exception: N/A

check():

- output: out:= Boolean
- exception: N/A

9 MIS of Game Loop Module

9.1 Module

GameLoop

9.2 Uses

N/A

9.3 Syntax

9.3.1 Exported Types

GameLoop = ?

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
GameLoop			
getAgents		Sequence of Player Objects	
getActionMask	$\mathbb{N}$		NoCommandModuleFound, AgentOutOfBounds
getGameState	$\mathbb{N}$		AgentOutOfBounds
getReward	$\mathbb{N}$		AgentOutOfBounds
playTurn	Command Object, $\mathbb{N}$		AgentOutOfBounds
checkGameOver		Boolean	

9.4 Semantics

9.4.1 State Variables

turn_counter: int
agents: Sequence of Players
state: Sequence of Sequences of int
turn_state: Enum

9.4.2 Environment Variables

None

9.4.3 Assumptions

There will be more state variables and access routines defined to implement the game rules but they are not listed here since they are game dependent.

9.4.4 Access Routine Semantics

GameLoop(agent_list, initial_state):

- transition :=
  turn_counter := 0
  agents := agent_list
  state := initial_state
- output: out := GameLoop
- exception: N/A

getAgents():

- output: out := agents
- exception: N/A

getActionMask(agent):

- output: out := Sequence of Command Objects
- exception:
  exc1 := agent > len(Agents) ⇒ AgentOutOfBounds
  exc2 := NoCommandModuleFound

getGameState(agent):

- output: out := state
- exception: exc := agent > len(Agents) ⇒ AgentOutOfBounds

getReward(agent):

- output: out := agent.get_rewards()
- exception: exc := agent > len(Agents) ⇒ AgentOutOfBounds

playTurn(action, agent):

- transition engine := action.execute()
- exception: exc := agent > len(Agents) ⇒ AgentOutOfBounds

checkGameOver():

- output out := if state.game_complete ⇒ True else ⇒ False
- exception: exc := agent > len(Agents) ⇒ AgentOutOfBounds

10 MIS of JSON Module

10.1 Module

JSON

10.2 Uses

Game Environment Module 7

10.3 Syntax

10.3.1 Exported Types

JSON = ?

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
jsonNamer	String	String	
jsonDirectory	String	String	
jsonWriter	String, String	JSON	PathDoesNotExist
jsonDump	String, String		FileDoesNotExist
jsonActionConverter	String, Sequence of String	JSON	

10.4 Semantics

10.4.1 State Variables

cur_directory : String
cur_time : String

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

N/A

10.4.4 Access Routine Semantics

jsonNamer(folder_name):

- transition: N/A
- output: out:= String
- exception: N/A

jsonDirectory(folder_path, json_name):

- transition: N/A
- output: out:= String
- exception:

jsonWriter(folder_name, json_name):

- transition: N/A
- output: out:= JSON
- exception: exc := if path.exists(folder_path) ⇒ False ⇒ PathDoesNotExist

jsonDump(simulation_history, json_name):

- transition:= := JSON
- output: out:= N/A
- exception: exc := if file.exists(json_name) ⇒ False ⇒ FileDoesNotExist

jsonActionConverter(folder_name, action_list):

- transition:= N/A
- output: out:= JSON
- exception: exc := if path.exists(folder_name) ⇒ False ⇒ PathDoesNotExist

11 MIS of Graph Module

11.1 Module

Graph

11.2 Uses

JSONDataParser 12

11.3 Syntax

N/A

11.3.1 Exported Types

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
render		HTML	

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

This module will render a graph using the JSONDataParser methods. It will be upto the developer to create their custom graphs.

11.4.4 Access Routine Semantics

render():

- transition := N/A
- output: out:= HTML
- exception: N/A

12 MIS of JSON Data Parser Module

12.1 Module

JsonDataParser

12.2 Uses

JSON Module 10

12.3 Syntax

12.3.1 Exported Types

DataParser

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
DataParser	Simulation[]	DataParser	
setAllData	Simulation[]		
setAllActions	Actions JSON Object		
getAllData		Simulation[]	
getAllActions		Actions JSON Object	
getAllDataExEnd	Simulation[]	Simulation[]	
getDataWithMergedActions	Simulation[]	Simulation[]	
getSimulationData	Simulation[], $\mathbb{Z}, \mathbb{Z}$	Simulation[]	
getPlayerData	Simulation, $\mathbb{Z}$	JSON Object	
getScores	Simulation[], $\mathbb{Z}$, $\mathbb{Z}$	JSON Object[]	
getNumberOfPlayers	Simulation[]	String[]	
getNumberOfSimulations	Simulation[]	$\mathbb{Z}$	

12.4 Semantics

12.4.1 State Variables

data: Simulation[] allActions: Actions JSON Object

12.4.2 Environment Variables

None

12.4.3 Assumptions

This module assumes that the developer will be able to parse the JSON files properly.

12.4.4 Access Routine Semantics

DataParser()(data):

- transition := data := data
- output: out:= self
- exception: N/A

setAllData(data):

- transition := data := data
- output: N/A
- exception: N/A

setAllActions(allActions):

- transition := allActions := allActions
- output: N/A
- exception: N/A

getAllData():

- transition := N/A
- output: data
- exception: N/A

getAllActions():

- transition := N/A
- output: allActions
- exception: N/A

getAllDataExEnd(inputData):

- transition :=
  result := []
  $\forall s : inputData.s < |inputData| - 1.sim = \{\}; (\forall i : simulationData[s].i < |simulationData[s]| - 1.simulationData[s][i].action \neq$ "End Turn" $\Rightarrow sim.add(simulationData[s][i])) \Rightarrow result.add(sim)$

- output: result

- exception: N/A

getDataWithMergedActions(inputData):

- transition :=
  result := []
  $\forall s : inputData.s < |inputData| - 1.sim = \{\}; (\forall i : simulationData[s].i < |simulationData[s]| - 1.simulationData[s][i].action \neq$ "meta_data" $\Rightarrow simulationData[s][i].action = simulationData[s][i]$
  $simulationData[s][i].action\_detail; sim.add(simulationData[s][i])) \Rightarrow result.add(sim)$

- output: result

- exception: N/A

getSimulationData(inputData, startIndex, endIndex):

- transition :=
  result := []
  $\forall s : inputData.s >= startIndex \& s < endIndex. \Rightarrow result.add(inputData[s])$

- output: result

- exception: N/A

getPlayerData(inputData, player):

- transition :=
  result := {}
  $\forall s : inputData.s < |inputData| - 1.inputData[s].player = player \Rightarrow result.add(s)$

- output: result

- exception: N/A

getScores(inputData, startSim, endSim):

- transition :=
  result := []
  simulationData := getSimulationData(inputData, startSim, endSim)
  $\forall s : simulationData.s < |simulationData| - 1.sim = \{\}; (\forall i : simulationData[s].i < |simulationData[s]| - 1.i.action =$ "meta_data" $\Rightarrow sim.add(simulationData[s][i])) \Rightarrow result.add(sim)$

- output: result
- exception: N/A

getNumberOfPlayers(inputData):

- transition :=
  result := []
  $\exists s : inputData.(i : inputData[s].i.action = \text{"meta\_data"} \Rightarrow i) \Rightarrow \forall p : i.result.add(p.split(\text{"\_"})[1])$
- output: result
- exception: N/A

getNumberOfSimulations(inputData):

- transition :=
  result := []
  $\forall s : inputData.s < |inputData| - 1. \Rightarrow result.add(s)$
- output: result
- exception: N/A

13 Appendix

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.