Module Interface Specification for Game of Continuous Life

Baptiste Pignier

March 18, 2025

1 Revision History

Date	Version	Notes
03/17/2025	1.0	First Release

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/BaptistePignier/CAS741-GameOfLife/blob/main/docs/SRS/SRS.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for Game of Continuous Life, a continuous version of Conway's Game Of Life.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/BaptistePignier/CAS741-GameOfLife.

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 Game of Continuous Life.

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 (-∞, ∞)

The specification of Game of Continuous Life 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, Game of Continuous Life 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.

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	US View Module US Control Module FI View Module FI Control Module Simulation View Module Simulation Control Module Window Manager Module
Software Decision Module	US Model Module FI Model Module Simulation Model Module

Table 1: Module Hierarchy

6 MIS of User Settings Model Module

According to Table 2 in the Module Guide document, this module is linked to requirement R2.

6.1 Module

us_model

6.2 Uses

None

6.3 Syntax

6.3.1 Exported Constants

TODO

6.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg—SS]	-	-	-

6.4 Semantics

6.4.1 State Variables

TODO [Not all modules will have state variables. State variables give the module a memory. —SS]

6.4.2 Environment Variables

None

6.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

6.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

7 MIS of User Settings View Module

According to Table 2 in the Module Guide document, this module is linked to requirements R1 and R3.

7.1 Module

us_view

7.2 Uses

This module uses the User Setting Model (Section 6) module for representing user settings.

7.3 Syntax

7.3.1 Exported Constants

TODO

7.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

7.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

8 MIS of User Settings Control Module

According to Table 2 in the Module Guide document, this module is linked to requirements R1, R2 and R3.

8.1 Module

us_control

8.2 Uses

This module uses the User Setting Model (Section 6) module for representing user settings.

8.3 Syntax

8.3.1 Exported Constants

TODO

8.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

TODO [This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

8.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

8.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

9 MIS of Functionnal Inputs Model Module

According to Table 2 in the Module Guide document, this module is linked to requirement R3.

9.1 Module

fi_model

9.2 Uses

None

9.3 Syntax

9.3.1 Exported Constants

TODO

9.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

9.4 Semantics

9.4.1 State Variables

TODO [Not all modules will have state variables. State variables give the module a memory. —SS]

9.4.2 Environment Variables

None

9.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

9.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

10 MIS of Functionnal Inputs View Module

According to Table 2 in the Module Guide document, this module is linked to requirement R3.

10.1 Module

fi_view

10.2 Uses

This module uses the Functionnal Inputs Model (Section 9) module for representing functionnal inputs.

10.3 Syntax

10.3.1 Exported Constants

TODO

10.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

None

10.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

10.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

11 MIS of Functionnal Inputs Control Module

According to Table 2 in the Module Guide document, this module is linked to requirements R2 and R3.

11.1 Module

fi_control

11.2 Uses

This module uses the Functionnal Inputs Model (Section 9) module for representing functionnal inputs.

11.3 Syntax

11.3.1 Exported Constants

TODO

11.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

TODO [This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

11.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

11.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

12 MIS of Simulation Model Module

According to Table 2 in the Module Guide document, this module is linked to requirement R4.

12.1 Module

sim_model

12.2 Uses

None

12.3 Syntax

12.3.1 Exported Constants

TODO

12.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

12.4 Semantics

12.4.1 State Variables

TODO [Not all modules will have state variables. State variables give the module a memory. —SS]

12.4.2 Environment Variables

None

12.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]

- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

12.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

13 MIS of Simulation View Module

According to Table 2 in the Module Guide document, this module is linked to requirement R5.

13.1 Module

sim_view

13.2 Uses

This module uses the Simulation Model (Section 12) module for representing the simulation.

13.3 Syntax

13.3.1 Exported Constants

TODO

13.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

13.4 Semantics

13.4.1 State Variables

None

13.4.2 Environment Variables

None

13.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

13.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

14 MIS of Simulation Control Module

According to Table 2 in the Module Guide document, this module is linked to requirements R2 and R4.

14.1 Module

sim_control

14.2 Uses

This module uses the Simulation Model (Section 12) module for representing the simulation.

14.3 Syntax

14.3.1 Exported Constants

TODO

14.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

14.4 Semantics

14.4.1 State Variables

None

14.4.2 Environment Variables

TODO [This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

14.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

14.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

15 MIS of Window Manager Module

According to Table 2 in the Module Guide document, this module is linked to requirement R1.

15.1 Module

win_manager

15.2 Uses

This module uses the User Setting View (Section 7), the Functional Inputs View (Section 10) and the Simulation View (Section 13) to arrange views on GUI.

15.3 Syntax

15.3.1 Exported Constants

TODO

15.3.2 Exported Access Programs

TODO

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

15.4 Semantics

15.4.1 State Variables

TODO [Not all modules will have state variables. State variables give the module a memory. —SS]

15.4.2 Environment Variables

TODO [This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

15.4.3 Access Routine Semantics

TODO
[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

15.4.4 Local Functions

TODO

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

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.