Module Interface Specification for Flow

Team 9, min-cut
Ethan Patterson
Hussain Muhammed
Jeffrey Doan
Kevin Zhu
Chengze Zhao

November 13, 2025

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for Flow

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/ritual-17/flow.

4 Notation

The specification of Flow 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, Flow 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

Level 1	**Level 2**
Hardware-Hiding	6 Display Interface Module 7 Input Interface Module 8 File Interface Module
Behaviour-Hiding	9 Geometry State Parser Module 10 Geometry State Converter Module 11 Commands Parser Module 12 Mode Commands Module 13 Shape Interface Module 14 User Preference Module 9 Geometry State Parser Module
Software Decision	15 Text Buffer Module 16 Geometry State Module 17 Geometry State Mutator Module 18 Undo Redo Module 19 User Persistence Module

Table 1: Module Hierarchy

6 MIS of Display Interface Module

6.1 Module

Display for the system

This is the module that will handle the display for the system. This is a module that when called updates the display.

6.2 Uses

16 Geometry State Module

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
display	Shapes	Display to screen	-

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

- System Screen
- Rendering API

6.4.3 Assumptions

6.4.4 Access Routine Semantics

display(Shapes):

- output: Outputs Shapes visually on the screen

6.4.5 Local Functions

N/A

7 MIS of Input Interface Module

7.1 Module

Input Module. Deals with User inputs and passes them on to the 12Mode Commands Module

7.2 Uses

12 Mode Commands Module

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_keyboard_inputs		keyboard_inputs	-
get_mouse_inputs		mouse_inputs	-

7.4 Semantics

7.4.1 State Variables

7.4.2 Environment Variables

- Keyboard API
- Mouse API

7.4.3 Assumptions

7.4.4 Access Routine Semantics

get_keyboard_inputs():

- output: Data Structure containing all current keyboard inputs. All keys will be off if the user is not using a keyboard.

get_mouse_inputs():

- output: Data Structure containing mouse location and inputs. All inputs are off if the user does not have a mouse.

7.4.5 Local Functions

N/A

8 MIS of File Interface Module

8.1 Module

9 Geometry State Parser Module

8.2 Uses

12 Mode Commands Module

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
open	file_path	opened_file	Invalid File
save	file_path, opened_note	success	-

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 Environment Variables

- File System

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

open(file_path):

- output: The opened file
- exception: Invalid File if file is missing or of an incorrect type.

save(file_path, opened_note):

- output: If the Module was successful in saving opened_note to file_path

8.4.5 Local Functions

N/A

9 MIS of Geometry State Parser Module

9.1 Module

File parser takes the file input and parses it into data structure containing all shapes.

9.2 Uses

N/A

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
parse	opened_file	shapes	Invalid File

9.4 Semantics

9.4.1 State Variables

N/A

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

parse(opened_file):

- output: Data structure containing all shapes from the given file.
- exception: Invalid File: if the file given is able to be parsed.

9.4.5 Local Functions

10 MIS of Geometry State Converter Module

10.1 Module

Note converter that converts notes to files of different file types.

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
convert	opened_note, file_type	file	Invalid file_type

10.4 Semantics

10.4.1 State Variables

10.4.2 Environment Variables

10.4.3 Assumptions

10.4.4 Access Routine Semantics

convert(opened_note, file_type):

- output: file that is ready to be saved on the computer.
- exception: Invalid file_type if the given file type is not compatable with the ones implemented

10.4.5 Local Functions

convert_*(opened_note) functions that do the converting once the file type has been determined. (ex convert_pdf)

11 MIS of Commands Parser Module

11.1 Module

Command Parser, converts keyboard inputs into commands and passes them onto the shape mutator.

11.2 Uses

17Geometry State Mutator Module 14 User Preference Module

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
parse_commands	keyboard_inputs, mouse_inputs	commands	-

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

Inputs have been already cleaned by the other modules.

11.4.4 Access Routine Semantics

parse_commands(keyboard_inputs, mouse_inputs):

- output: commands, which would be used by the Geometry State Mutator Module 17

11.4.5 Local Functions

N/A

12 MIS of Mode Commands Module

12.1 Module

Main Module contains the commands usable in the current mode.

12.2 Uses

N/A

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_commands	mode	commands	-

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

Mode given will always be a valid mode.

12.4.4 Access Routine Semantics

get_commands(mode):

- output: commands that can be run in the given mode

12.4.5 Local Functions

N/A

13 MIS of Shape Interface Module

13.1 Module

Module that contains information about shapes.

13.2 Uses

13.3 Syntax

13.3.1 Exported Constants

Shapes: List of shapes

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
create_shape	shape_type, points	shape	Invalid Points
move_shape	shape, new_points	-	Invalid Points

13.4 Semantics

13.4.1 State Variables

N/A

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

N/A

13.4.4 Access Routine Semantics

create_shape(shape_type, points):

- output: Shape with type shape_type using points for location and size.
- exception: Invalid points if the points are not possible for the type of shape.

move_shape():

- transition: changes the points of the shape to the new points
- exception: Invalid points if the points are not possible for the type of shape.

13.4.5 Local Functions

N/A

14 MIS of User Preference Module

14.1 Module

User preferences, provides an interface for creating and manipulating user preferences in the system.

14.2 Uses

6 Display Interface Module 19 User Persistence Module

14.3 Syntax

14.3.1 Exported Constants

N/A

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_command_prefrences	preferences	commands	-
modify	-	-	-
get_theme_prefrences		theme	-

14.4 Semantics

14.4.1 State Variables

Command preferences: users defined commands Theme preferences: user theme prefrences

14.4.2 Environment Variables

14.4.3 Assumptions

14.4.4 Access Routine Semantics

get_command_prefrences(mode):

- output: Any user defined preferences for the current mode

modify():

- transition: Has the user input their preferences where the moudles saves it to a file.

get_theme_prefrences():

- output: Theme preferences for the system

14.4.5 Local Functions

15 MIS of Text Buffer Module

15.1 Module

Text, Module for Text representation in notes.

15.2 Uses

N/A

15.3 Syntax

15.3.1 Exported Constants

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
new_text	location,text,formatting	text_box	Invalid Text, Invalid Location
modify_text	text_box,text,formatting	-	Invalid Text
move_text	text_box,location	-	Invalid Location

15.4 Semantics

15.4.1 State Variables

N/A

15.4.2 Environment Variables

N/A

15.4.3 Assumptions

15.4.4 Access Routine Semantics

new_text(location,text,formatting):

- output: Text box at location with text and formatting
- exception: Invalid Text, Invalid Location

modify_text(text_box,text,formatting):

- transition: Text in text_box is modified to now contain text with formatting
- exception: Invalid Text

move_text(text_box,location):

- transition: text_box is moved to location.
- exception: Invalid Location

15.4.5 Local Functions

N/A

16 MIS of Geometry State Module

16.1 Module

Geometry State, contains info on all current geometry

16.2 Uses

13 Shape Interface Module 15 Text Buffer Module

16.3 Syntax

16.3.1 Exported Constants

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_shapes	-	shapes	-
get_text	-	text_boxes	-

16.4 Semantics

16.4.1 State Variables

Shapes: data structure holding all shapes Text_Boxes: data structure holding all Text Boxes

16.4.2 Environment Variables

N/A

16.4.3 Assumptions

N/A

16.4.4 Access Routine Semantics

get_shapes():

- output: Data Structure containing all shapes

get_text():

- output: Data structure containing all text boxes

16.4.5 Local Functions

N/A

17 MIS of Geometry State Mutator Module

17.1 Module

Geometry State Mutator Module, Provides methods to mutate the geometry state (add, delete, modify shapes).

17.2 Uses

11 Commands Parser Module 13 Shape Interface Module 15 Text Buffer Module 16 Geometry State Module

17.3 Syntax

17.3.1 Exported Constants

N/A

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
delete_text	text_box	-	-
delete_shapes	shape	-	-
add_text	text_box	-	-
add_shape	shape	-	-
modify_text	text_box,text,formatting	-	Invalid Text
move_shape	shape, new_points	-	Invalid Points

17.4 Semantics

17.4.1 State Variables

Geometry_State_Module:Stores the info on the shapes and Text boxes

17.4.2 Environment Variables

N/A

17.4.3 Assumptions

N/A

17.4.4 Access Routine Semantics

add_text(text_box):

- transition: Adds the text_box to Text_Boxes

add_shape(shape):

- transition: Adds the shape to Shapes

delete_text(text_box):

- transition: Removes the text_box to Text_Boxes

delete_shape(shape):

- transition: Removes the shape to Shapes

modify_text(text_box,text,formatting):

- transition: Text in text_box is modified to now contain text with formatting
- exception: Invalid Text

move_shape():

- transition: changes the points of the shape to the new points
- exception: Invalid points if the points are not possible for the type of shape.

17.4.5 Local Functions

N/A

18 MIS of Undo Redo Module

18.1 Module

Undo/Redo Module: manages undo and redo commands.

18.2 Uses

11 Commands Parser Module 17 Geometry State Mutator Module 8 File Interface Module

18.3 Syntax

18.3.1 Exported Constants

N/A

18.3.2 Exported Access Programs

Name	In	Out	Exceptions
undo	-	-	Cant Undo
redo	-	-	Cant Redo
add_command	command	-	-

18.4 Semantics

18.4.1 State Variables

Previous_Commands: previous command that can be used for undo Redone_Commands: commands that can be used for redo

18.4.2 Environment Variables

N/A

18.4.3 Assumptions

N/A

18.4.4 Access Routine Semantics

undo():

- transition: Uses the Geometry State mutator to undo a previous command.

redo():

- transition: Uses the Geometry State mutator to redo a previous command.

redo():

- transition: Uses the Geometry State mutator to redo a previous command.

add_command(command):

- transition: Stores the command to use later in case of an undo.

18.4.5 Local Functions

N/A

19 MIS of User Persistence Module

19.1 Module

User Persistence Module: Module for saving and loading user preferences and short-cuts.

19.2 Uses

19.3 Syntax

19.3.1 Exported Constants

N/A

19.3.2 Exported Access Programs

Name	In	Out	Exceptions
save	commands,theme	-	-
get_commands	-	commands	-
get_theme	-	theme	-

19.4 Semantics

19.4.1 State Variables

N/A

19.4.2 Environment Variables

N/A

19.4.3 Assumptions

N/A

19.4.4 Access Routine Semantics

save(commands,theme):

- transition: Saves the commands and theme as a file

get_commands():

- output: User defined commands read from the file

get_theme():

- output: User preferred themes read from the file

19.4.5 Local Functions

N/A

References

20 Appendix

Appendix — Reflection

[Not required for CAS 741 projects —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?

4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?

5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)