EECS3311 - W Z - Project Report

Submitted electronically by: Sohrab Oryakhel

Team Members	**Name**	**Prism Login**	**Signature**
Member 1:	Sohrab Oryakhel	sohrab23	S.Oryakhel
Member 1:	Mehrzad Bazhdanzadeh	mehrzadb	M.Bazhdanzadeh

Contents

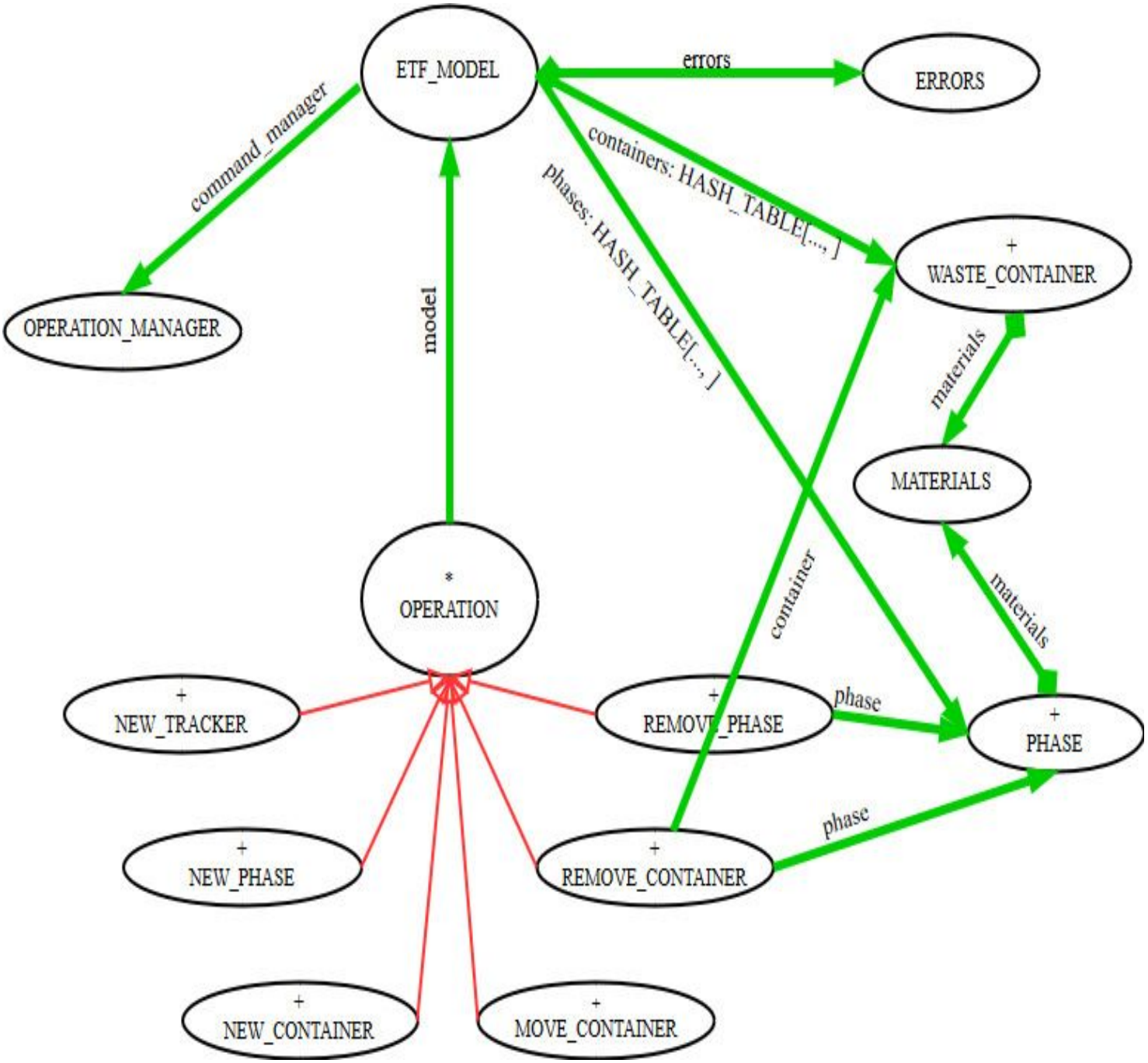
Requirements for Project Tracker

Nuclear Waste Tracking System:

A tracker that monitors the position of waste products in nuclear plants and ensures the safe handling thereof. This tracker is a software system that allows operators to manage safe tracking of waste that is radioactive in their plants.

Containers or materials consisting of radioactive waste pass through various phases in the tracking system. Containers and phases have radiation capacities and allowed materials which differentiate between the phases. The tracking system keeps track of all the phases and containers it has, materials in containers and phases and the amount of radioactivity so that nothing dangerous occurs. In case of such events, the tracker is also equipped with undo-redo capabilities. Now if accidents occur, investigators can replay the circumstances under which it did so.

Bon class diagram overview (architecture of the design)



Design decisions

The main design pattern used in this project is Command Pattern. In this way classes and modules are designed to to serve as the commands themselves. Each command used by the user initiates an object instance representing that very command and this is the essence of command pattern. All of the commands inherit from OPERATION which is the superclass providing the abstract features of an executable command. Since undo/redo is also a requirement for this project, those capabilities are abstract features of OPERATION and together this allows a subclass of OPERATION to execute, undo and redo itself.

The ETF_MODEL class is the actual tracker which has the phases (HASH_TABLE), containers(HASH_TABLE) and error messages in it. This class also has a manager for the operations so that it can call the proper routines to begin execution of the commands. The design of this class is kept simple and this class does only the work it is responsible for.

PHASE and WASTE_CONTAINER are classes that each represent a phase and container. Their design is also very simple in that those classes only contain information about themselves and not any other class. An instance of WASTE_CONTAINER and PHASE only has things like identification, capacity and other crucial information. The ERRORS is one that contains strings representing the error messages and the notable thing about this class is that those strings are immutable. The class MATERIALS is an expanded singleton class so that we do not have to create arrays multiple times, and can instead just use the one.

Table of modules (responsibilities and information hiding)

1.	Class: ETF_MODEL Implementation: Concrete.	Responsibility: Main tracker class. secrete: none.	Alternative: none.

2.	Class: ERRORS Abstract.	Responsibility: Error messages. Secret: messages are immutable.	Alternative: Concrete class with mutable error strings.

3.	Class: MATERIALS Abstract.	Responsibility: Array of strings of materials. Secret: Singleton.	Alternative: Multiton. Create array each time.

4.	Class: PHASE Concrete.	Responsibility: Represent a phase in tracking system. Secret: materials array is singleton.	Alternative: materials array is multiton. Create array each time.

5.	Class: WASTE_CONTAINER Concrete.	Responsibility: Represent a container in tracking system. Secret: materials array is singleton.	Alternative: materials array is multiton. Create array each time.

6.	Class: OPERATION Deferred.	Responsibility: Abstract class for commands. Secret: none.	Alternative: Different Design Pattern.
6.1	Command Module: NEW_TRACKER concrete.	Responsibility: Execution of new_tracker	Alternative: Explicitly coded in ETF_MODEL, not

		command. Secret: attributes.	Command Pattern.
6.2	NEW_PHASE concrete.	Responsibilities: same as above.	Alternative: same as above.
6.3	NEW_CONTAINER concrete.	Responsibility: Same as above.	Alternative: same as above.
6.4	NEW_CONTAINER concrete.	Responsibility: Same as above.	Alternative: same as above.
6.5	MOVE_CONTAINER concrete.	Responsibility: Same as above.	Alternative: same as above.
6.6	REMOVE_CONTAINER concrete.	Responsibility: Same as above.	Alternative: same as above.
6.7	REMOVE_PHASE concrete.	Responsibility: Same as above.	Alternative: same as above.

7.	Class: OPERATION_MANAGER concrete.	Responsibility: Manager of the history lists and undo/redo routines.	Alternative: Have history lists and their corresponding operations in ETF_MODEL.

Expanded description of design decisions

The undo/redo design in this project is such that *undo* and *redo* are user commands which are called as routines in the model, the command_manager, the actual command instance. Commands like *new_tarcker and remove_phase()* inherit from OPERATION, meaning they will exists as instances of OPERATION once they are called. They will also have their own execute, undo, and redo routines. Once these commands are executed they will be popped onto the *undo_list* in OPERATION_MANAGER, the manager class of the lists. The two lists, *undo_list* and *redo_list* are implemented as stacks which makes additions and removals simple. The crucial thing is that once the commands are in the *undo_list*, the *undo* command will be a routine in OPERATION_MANAGER which will run the the *undo* routine in the actual command instance. After this, OPERATION_MANAGER will put the undone command in the *redo_list* and that list works the same as its counterpart.

The command classes are all implemented similarly as they all inherit from OPERATION. The error checking for each command is done it its own execution routine which makes it a good separation of concern and uses defensive programming. They each contain the information to execute the command and also the previous output incase that command is undone.

A crucial class is the ETF_MODEL class which contains the phases in a hash table as it does with the containers. Hash tables seemed like the obvious choice when designing the class. The class also contains maximum radiation values and

error strings. The class contains routines which call other routines to execute the commands.

Significant contracts

OPERATION_MANAGER: This class has contracts for its main routines. The first routine of the class is the execute routine and it has an ensure statement which makes sure that after a command has been executed and put on the undo list, the undo list should *have* that command and also it should have a size that is greater than the previous one. The next routine is the undo routine which also has an ensure statement. This routine ensures two things; the size of redo list has increased and size of undo list has decreased. This is because when a command is undone it is popped from the stack and hence removed. That same command is then push onto or added to the redo stack which in turn increases the size of it. The third and last command in the class with contracts is the redo routine which does and ensures the same things but just in the opposite direction as the undo routine.

Summary of testing procedures

Test File	**Description**	**Passed**
at1.txt	Tests all the commands and their undo and redo routines under non-erroneous input	yes
at2.txt	A very simple test that tests undoing an error outputted by new_container command	yes
at3.txt	Simple test that tests the operation of commands and also error outputs. No undo/redo-ing.	yes
at4.txt	Tests undoing and redoing beyond the commands executed.	yes
at5.txt	More testing of the undo/redo feature of several commands.	yes
at6.txt	Testing of erroneous output of new_tracker command and the undo/redo-ing thereof.	yes
at7.txt	More testing of undoing and redoing beyond the commands executed.	yes

Appendix (contract view of all classes)

ETF_MODEL

```
note
	description: "A default business model."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

class interface
	ETF_MODEL

create {ETF_MODEL_ACCESS}
	make

feature -- model attributes

	i: INTEGER_32

	phases: HASH_TABLE [PHASE, STRING_8]

	max_phase_radiation: VALUE

	containers: HASH_TABLE [WASTE_CONTAINER, STRING_8]

	max_container_radiation: VALUE

	materials: MATERIALS

	errors: ERRORS

	status: STRING_8
		-- ouput status. either "ok" or one of e1 to e20

	use_past_state: BOOLEAN
		-- a state representing whether or not we use the substring "(to x)" in output

	command_manager: OPERATION_MANAGER

	i_of_nt: INTEGER_32
		-- integer representing the state (i) of new_tracker command
```

```
feature -- model operations

	default_update
		-- Perform update to the model state.

	reset
		-- Reset model state.

	new_tracker (m_p_r: VALUE; m_c_r: VALUE)

	new_phase (pid: STRING_8; pn: STRING_8; cap: INTEGER_64; em: ARRAY
[INTEGER_64])

	new_container (cid: STRING_8; c: TUPLE [material: INTEGER_64; radioactivity: VALUE];
pid: STRING_8)

	move_container (cid: STRING_8; pid1: STRING_8; pid2: STRING_8)

	remove_phase (pid: STRING_8)

	remove_container (cid: STRING_8)

	undo

	redo

	set_status (s: STRING_8)
		-- setter for output status

	set_max_radiation (mpr: VALUE; mcr: VALUE)
		-- setter for max radiation use by new_tracker command

	set_past_state (b: BOOLEAN)
		-- setter for allowing us to use the past state substring "(to x)" in output

	set_iont (f: INTEGER_32)

feature -- queries

	to_state: STRING_8

	print_phases: STRING_8

	print_containers: STRING_8

	out: STRING_8
		-- New string containing terse printable representation
		-- of current object

end -- class ETF_MODEL
```

PHASE

note
description: "A phase in the nuclear waste tracking system."
author: "Sohrab Oryakhel"
date: "$Date$"
revision: "$Revision$"

class interface
PHASE

create
make

feature -- attributes

id: STRING_8
-- phsae identification

name: STRING_8
-- phase name

capacity: INTEGER_32
-- phase capacity

container_count: INTEGER_32
-- current containers in phase

rad_count: VALUE
-- perliminary radiation count

expected_materials: ARRAY [INTEGER_64]
-- materials expected by phase

materials: MATERIALS

feature -- commands

add_material (rad: VALUE)

remove_material (rad: VALUE)

feature -- queries

accepts_material (material: INTEGER_32): BOOLEAN

will_exceed_capacity: BOOLEAN

out: STRING_8

```
                -- New string containing terse printable representation
                -- of current object

        materials_list: STRING_8

        infix "<" (other: like Current): BOOLEAN
                -- Is current object less than other?

end -- class PHASE
WASTE_CONTAINER

note
        description: "A container of nuclear waste."
        author: "Sohrab Oryakhel"
        date: "$Date$"
        revision: "$Revision$"

class interface
        WASTE_CONTAINER

create
        make

feature -- attributes

        id: STRING_8
                -- container identification

        pid: STRING_8
                -- phase identification

        rad_count: VALUE
                -- perliminary radiation count

        material: INTEGER_32
                -- material in container

        materials: MATERIALS

feature -- commands

        transfer_to_phase (p: STRING_8)

feature -- queries
-- getters below are for information hidding.

        get_material: STRING_8

        out: STRING_8
                -- New string containing terse printable representation
                -- of current object
```

```
	infix "<" (other: like Current): BOOLEAN
		-- Is current object less than other?

end -- class WASTE_CONTAINER
```

MATERIALS

```
note
	description: "Summary description for {MATERIALS}."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

expanded class interface
	MATERIALS

create
	default_create

feature

	M: ARRAY [STRING_8]

invariant
	M = M

end -- class MATERIALS
```

ERRORS

```
note
	description: "This class contains all the possible error outputs."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

expanded class interface
	ERRORS

create
	default_create

feature -- attributes

	Ok: STRING_8 = "ok"

	E1: STRING_8 = "e1: current tracker is in use"
```

```
		E2: STRING_8 = "e2: max phase radiation must be non-negative value"

		E3: STRING_8 = "e3: max container radiation must be non-negative value"

		E4: STRING_8 = "e4: max container must not be more than max phase radiation"

		E5: STRING_8 = "e5: identifiers/names must start with A-Z, a-z or 0..9"

		E6: STRING_8 = "e6: phase identifier already exists"

		E7: STRING_8 = "e7: phase capacity must be a positive integer"

		E8: STRING_8 = "e8: there must be at least one expected material for this phase"

		E9: STRING_8 = "e9: phase identifier not in the system"

		E10: STRING_8 = "e10: this container identifier already in tracker"

		E11: STRING_8 = "e11: this container will exceed phase capacity"

		E12: STRING_8 = "e12: this container will exceed phase safe radiation"

		E13: STRING_8 = "e13: phase does not expect this container material"

		E14: STRING_8 = "e14: container radiation capacity exceeded"

		E15: STRING_8 = "e15: this container identifier not in tracker"

		E16: STRING_8 = "e16: source and target phase identifier must be different"

		E17: STRING_8 = "e17: this container identifier is not in the source phase"

		E18: STRING_8 = "e18: this container radiation must not be negative"

		E19: STRING_8 = "e19: there is no more to undo"

		E20: STRING_8 = "e20: there is no more to redo"

end -- class ERRORS
```

OPERATION_MANAGER

```
note
	description: "Manager class of the operations and the undo-redo lists."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

class interface
	OPERATION_MANAGER
```

```
create
	make

feature

	make

feature -- attributes

	undo_list: ARRAYED_STACK [detachable OPERATION]

	redo_list: ARRAYED_STACK [detachable OPERATION]

feature -- commands

	execute_command (command: OPERATION)
	ensure
		undo_list.has (command)

	undo
	ensure
		redo_list.count > old redo_list.count
		undo_list.count < old undo_list.count

	redo
	ensure
		redo_list.count < old redo_list.count
		undo_list.count > old undo_list.count

	clear_lists

end -- class OPERATION_MANAGER
```

OPERATION

```
note
	description: "deferred class representing the common operations of the program."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

deferred class interface
	OPERATION

feature

	make (arg: TUPLE)

feature
```

```
		i: INTEGER_32

		model: ETF_MODEL

		prev_status: STRING_8

		input_incorrect: BOOLEAN

invariant
		model = model

end -- class OPERATION
```

NEW_TRACKER

```
note
		description: "Summary description for {NEW_TRACKER}."
		author: "Sohrab Oryakhel"
		date: "$Date$"
		revision: "$Revision$"

class interface
		NEW_TRACKER

create
		make

feature -- creation

		make (args: TUPLE [m_p_r: VALUE; m_c_r: VALUE])

feature -- attributes

		mpr: VALUE
				-- max phase radiation of tracker

		mcr: VALUE
				-- max container radiation of tracker

feature -- commands

		execute

		undo

		redo

end -- class NEW_TRACKER
```

NEW_PHASE

```
note
	description: "Summary description for {NEW_PHASE}."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

class interface
	NEW_PHASE

create
	make

feature -- creation

	make (args: TUPLE [pid: STRING_8; pn: STRING_8; cap: INTEGER_64; em: ARRAY
[INTEGER_64]])

feature -- attributes

	phase_id: STRING_8

	phase_name: STRING_8

	capacity: INTEGER_64

	expected_materials: ARRAY [INTEGER_64]

feature -- commands

	execute

	undo

	redo

end -- class NEW_PHASE
```

NEW_CONTAINER

```
note
	description: "Summary description for {NEW_CONTAINER}."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

class interface
```

NEW_CONTAINER

create
make

feature -- creation

make (args: TUPLE [cid: STRING_8; mat: INTEGER_64; rad: VALUE; pid: STRING_8])

feature -- attributes

container_id: STRING_8

material: INTEGER_32

radiation_count: VALUE

phase_id: STRING_8

feature -- commands

execute

undo

redo

end -- class NEW_CONTAINER

MOVE_CONTAINER

note
description: "Summary description for {MOVE_CONTAINER}."
author: "Sohrab Oryakhel"
date: "$Date$"
revision: "$Revision$"

class interface
MOVE_CONTAINER

create
make

feature -- creation

make (args: TUPLE [c_id: STRING_8; p_id1: STRING_8; p_id2: STRING_8])

feature -- attributes

cid: STRING_8

```
        pid1: STRING_8

        pid2: STRING_8

feature -- commands

        execute

        undo

        redo

end -- class MOVE_CONTAINER
```

REMOVE_CONTAINER

```
note
        description: "Summary description for {REMOVE_CONTAINER}."
        author: "Sohrab Oryakhel"
        date: "$Date$"
        revision: "$Revision$"

class interface
        REMOVE_CONTAINER

create
        make

feature -- creation

        make (args: TUPLE [c_id: STRING_8])

feature -- attributes

        cid: STRING_8

        phase: PHASE

        container: WASTE_CONTAINER

feature -- commands

        execute

        undo

        redo

end -- class REMOVE_CONTAINER
```

REMOVE_PHASE

```
note
	description: "Summary description for {REMOVE_PHASE}."
	author: "Sohrab Oryakhel"
	date: "$Date$"
	revision: "$Revision$"

class interface
	REMOVE_PHASE

create
	make

feature -- creation

	make (args: TUPLE [p_id: STRING_8])

feature -- attributes

	pid: STRING_8
		-- phase id for removal

	phase: PHASE
		-- phase for undoing removal

feature -- commands

	execute

	undo

	redo

end -- class REMOVE_PHASE
```