University of Zurich[UZH]	**Test Plan for** **AutoTask Library**	Author: Ali, Chen, Liu Date: 08-11-2018 Page of Pages: 1

Revision History

Date	Version	Description	Author(s)
10/11/2018	Final Version	Test Plan Completed	Ali, Chen, Liu

1. Public API Overview

1.1. Public classes to be tested

AUTO_TASK

DIRECTED_GRAPH

CONVERTER

DISPLAYER

ALGORITHM

1.2. Public routines to be tested

AUTO_TASK

- add_element (element_name: STRING)
- add_constraint (element_name_1, element_name_2: STRING)
- delete_element (element_name: STRING)
- delete_constraint (element_name_1, element_name_2: STRING)
- topo_sort: LINKED_LIST [STRING]

DIRECTED_GRAPH

- add_node (node: INTEGER)
- add_edge (predecessor, successor: INTEGER)
- delete_node (node: INTEGER)
- delete_edge (predecessor, successor: INTEGER)

CONVERTER

- name2id (name: STRING): INTEGER
- id2name (id: INTEGER): STRING
- insert_name (name: STRING)
- delete_name (name: STRING)

DISPLAYER

- to_graphviz_format (graph: DIRECTED_GRAPH)

ALGORITHM

- cycle_detection (graph: DIRECTED_GRAPH): BOOLEAN
- topo_sort (graph: DIRECTED_GRAPH): LINKED_LIST [INTEGER]

2. Test Suite Description

2.1. Test Cases

Test ID	2.1.1
Requirements under test	3.2.01, 3.2.02, 3.2.04, 3.2.05, 3.2.09
Routines under test	Class: AUTO_TASK add_element (element_name: STRING) add_constraint (element_name_1, element_name_2: STRING) delete_element (element_name: STRING) delete_constraint (element_name_1, element_name_2: STRING) display() Class: CONVERTER, DIRECTED_GRAPH, DISPLAYER (all routines)
Description	The library shall be able to add or remove elements and constraints with strings as an identifiers of elements.
Set-up	An instance of AUTO_TASK should be created
Tear-Down	The instance of AUTO_TASK should be deleted automatically by the Eiffel garbage collector after termination of testing.
Test data	add_element(“event1”) add_element(“event2”) add_element(“event3”) add_constraint(“event1”, “event2”) add_constraint(“event1”, “event3”) add_constraint(“event2”, “event3”) delete_constraint(“event1”, “event3”) delete_element(“event3”)
Oracle	Call AUTO_TASK.display(), and a GraphViz format text file will be generated containing “event1 -> event2” only.

Test ID	2.1.2
Requirements under test	3.2.11, 3.2.12, 3.2.13, 3.2.14
Routines under test	Class: AUTO_TASK add_element (element_name: STRING) add_constraint (element_name_1, element_name_2: STRING) delete_element (element_name: STRING) delete_constraint (element_name_1, element_name_2: STRING)
Description	The library shall be print warning to the console if any aspect of the input integrity is violated.
Set-up	An instance of AUTO_TASK should be created
Tear-Down	The instance of AUTO_TASK should be deleted automatically by the Eiffel garbage collector after termination of testing.
Test data	add_element(“event1”) add_element(“event1”) add_element(“e.....e”) (101 characters of ‘e’) add_constraint(“event1”, “event2”)
Oracle	The following messages would be printed on the console: Element Duplication: event1 Element Max Length Exceed: eee...eee Element Not Exist: event2

Test ID	2.1.3
Requirements under test	3.2.01, 3.2.02, 3.2.04, 3.2.05
Routines under test	Class: CONVERTER name2id (name: STRING): INTEGER id2name (id: INTEGER): STRING insert_name (name: STRING) delete_name (name: STRING)
Description	Every element is uniquely identified by a name(STRING) or by an id (INTEGER). The converter should be able to convert an element name to its corresponding id or vice versa.
Set-up	An instance of CONVERTER should be created
Tear-Down	The instance of CONVERTER should be deleted automatically by the Eiffel garbage collector after termination of testing.
Test data	insert_name(“event1”) insert_name(“event2”) insert_name(“event3”) delete_name(“event3”)
Oracle	name2id(“event1”) = 1 name2id(“event2”) = 2 id2name(2) = “event2” name2id(“event3”) = 0 (which means not exists or deleted, error message will be printed in the console) id2name(3) = “” (which means not exists or deleted, error message will be printed in the console)

Test ID	2.1.4
Requirements under test	3.2.01, 3.2.02, 3.2.04, 3.2.05, 3.2.09
Routines under test	Class: DIRECTED_GRAPH add_node (node: INTEGER) add_edge (predecessor, successor: INTEGER) delete_node (node: INTEGER) delete_edge (predecessor, successor: INTEGER) Class: DISPLAYER to_graphviz_format (graph: DIRECTED_GRAPH)
Description	Every node is uniquely identified by an id (INTEGER) in the directed graph. The DIRECTED_GRAPH class should be manipulated with node/edge addition/deletion.
Set-up	An instance of DIRECTED_GRAPH should be created.
Tear-Down	The instance of DIRECTED_GRAPH should be deleted automatically by the Eiffel garbage collector after termination of testing.
Test data	add_node(1) add_node(2) add_node(3) add_edge(1, 2) add_edge(1, 3) add_edge(2, 3) delete_edge(1, 3) delete_node(3)
Oracle	By calling DISPLAYER.to_graphviz_format(), with the DIRECTED_GRAPH instance, a GraphViz format text file will be generated containing “event1 -> event2” only.

Test ID	2.1.5
Requirements under test	3.2.06, 3.2.07, 3.2.08, 3.2.09, 3.2.10
Routines under test	Class: ALGORITHM +cycle_detection (graph: DIRECTED_GRAPH) : BOOLEAN +topo_sort (graph: DIRECTED_GRAPH) : LINKED_LIST[INT] Class: DIRECTED_GRAPH add_node (node: INTEGER) add_edge (predecessor, successor: INTEGER) delete_node (node: INTEGER) delete_edge (predecessor, successor: INTEGER) Class: DISPLAYER to_graphviz_format (graph: DIRECTED_GRAPH)
Description	Every node is uniquely identified by an id (INTEGER) in the directed graph. The ALGORITHM must be able to detect the cycle between entered elements/constraints. The DISPLAYER shall be able to display the graph of cyclic parts if detected.
Set-up	An instance of ALGORITHM should be created
Tear-Down	The instance of ALGORITHM should be deleted automatically by the Eiffel garbage collector after termination of testing.
Test data	add_node(1), add_node(2), add_node(3), add_node(4) add_edge(1, 2), add_edge(2, 3), add_edge(3, 4), add_edge(4, 1) Cycle_detection (graph: DIRECTED_GRAPH) : BOOLEAN
Oracle	By calling DISPLAYER.to_graphviz_format(), the cycle containing node 1, 2, 3, and 4 will be printed on the console.

Test ID	2.1.6
Requirements under test	3.2.06, 3.2.07, 3.2.08
Routines under test	Class: ALGORITHM +cycle_detection (graph: DIRECTED_GRAPH) : BOOLEAN +topo_sort (graph: DIRECTED_GRAPH) : LINKED_LIST[INT] Class: DIRECTED_GRAPH add_node (node: INTEGER) add_edge (predecessor, successor: INTEGER) delete_node (node: INTEGER) delete_edge (predecessor, successor: INTEGER)
Description	Every node is uniquely identified by an id (INTEGER) in the directed graph. The ALGORITHM must be able to detect the cycle between entered elements/constraints. If there is no-cycle, the ALGORITHM must be able to perform TOPOLOGICAL SORT on it.
Set-up	An instance of ALGORITHM should be created.
Tear-Down	The instance of ALGORITHM should be deleted automatically by the Eiffel garbage collector after termination of testing.
Test data	add_node(1), add_node(2), add_node(3) add_edge(1, 2), add_edge(2, 3),
Oracle	By calling topo_sort (graph: DIRECTED_GRAPH), a linked list of 2 integers will be returned, with 1 as the head, and 2 as the tail of the list.

3. Expected Coverage

We expect that the tests described in section 2 covers all public routines of the 5 classes. Also the expected coverage rate over all branches would be 80%.

4. Benchmarks

We will use a randomly generated acyclic directed graph with 2,000 elements and 100,000 constraints as the benchmark. The library should be able to generate a text file for Graphviz in 500ms, and perform topological sort in 1,000ms. The results would be compared among different operating systems, including MacOSX, Windows, Ubuntu.