| | | |
|---|---|---|
| University of Zurich UZH | **Test Plan for AutoTasks** | Author: David Stalder, Dominik Jurilj, Yuliya Khandozhko<br>Doc.No.:<br>Date: 16-12-2018<br>Page of Pages: 1 of 16 |

# Contents

# Revision History

| Date | Version | Description | Author(s) |
|---|---|---|---|
| 3.11.18 | 0.1 | First few test cases | Stalder, Khandozhko, Jurilj |
| 4.11.18 | 0.2 | More test cases added | Stalder, Khandozhko, Jurilj |
| 5.11.18 | 0.3 | More test cases added | Stalder, Khandozhko, Jurilj |
| 7.11.18 | 0.4 | Adjustment of test cases | Stalder, Khandozhko, Jurilj |
| 10.11.18 | 0.5 | Writing of coverage | Stalder, Khandozhko, Jurilj |
| 11.11.18 | 1.0 | Final changes | Stalder, Khandozhko, Jurilj |
| 14.12.18 | 1.1 | Revisited plan | Stalder, Khandozhko, Jurilj |

# 1 Public API Overview

## 1.1 Public classes to be tested

- Input_handler
- Element_library
- Element
- Constraint
- Topological_sort
- Graph

## 1.2 Public routines to be tested

- Input_handler.input_start
- Input_handler.read_elements_and_constraints
- Element_library.ele_lib
- Element_library.transfer
- Element_library.add_element
- Element_library.add_constraint
- Element_library.add_elements_input
- Element_library.add_constraints_input
- Element_library.loop_elements
- Element_library.loop_constraints
- Element_library.example_1
- Element_library.example_2
- Element_library.run_example
- Element_library.element_in_list
- Element_library.constraint_in_list
- Topological_sort.has_element
- Topological_sort.add_successor
- Topological_sort.record_element
- Topological_sort.record_constraint
- Topological_sort.find_initial_candidates
- Topological_sort.report_cycles
- Topological_sort.process
- Graph.choose_graph
- Graph.show_graph
- Graph.show_cycle
- Element.make
- Constraint.constraint

# 2 Test Suite Description

Description of all test cases using the following format:

| | |
|---|---|
| Test ID | ID of test |
| Requirements under test | ID(s) of the requirement(s) under test |
| Routines under test | Name(s) of the routine(s) under test |

| | |
|---|---|
| Description | Description of what is tested |
| Set-up | Operations before executing test (preparations) |
| Tear-Down | Operations after executing test (clean-ups) |
| Test data | Data used while executing test |
| Oracle | Pass/fail criteria |

## 2.1 Test cases

| Test ID | TC_01 |
|---|---|
| Requirements under test | None |
| Routines under test | INPUT_HANDLER.read_elements_and_constraints |
| Description | User interface gets started<br>Every interaction with the software is callable from here: Add/remove elements/constraints, run examples, topological sort |
| Set-up | None |
| Tear-Down | Close user interface |
| Test data | None |
| Oracle | The user interface can be opened and the inputs entered with the expected answer from the console. |

| Test ID | TC_02 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.transfer |
| Description | Transfer the user input from INPUT_HANDLER to ELEMENT_LIBRARY |
| Set-up | Create an elements list with two elements<br>Create two constraint lists with one element each |

| | |
|---|---|
| Tear-Down | Clear input_element, input_constraint_first and input_constraint_second |
| Test data | Elements: ‘el1’, ‘el2’<br>Constraint: [e1, el2] |
| Oracle | The input_element list contains elements ‘el1’ and ‘el2’. The input_constraint_first list contains ‘el1’ and the input_constraint_second contains ‘el2’. |

| Test ID | TC_03 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.add_element |
| Description | Add an element to the all_elements list |
| Set-up | Create an empty elements list |
| Tear-Down | None |
| Test data | Elements: ‘el1’ |
| Oracle | The elements list contains the elements ‘el1’. |

| Test ID | TC_04 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.add_constraint |
| Description | Add a constraint to the all_constraints list |
| Set-up | Create an empty constraints list |
| Tear-Down | None |
| Test data | Elements: ‘el1’,’el2’ |
| Oracle | The constraints list contains the constraint ‘[el1,el2]’. |

| Test ID | TC_05 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.add_elements_input |
| Description | Add multiple elements to the all_elements list |
| Set-up | Create an empty elements list |
| Tear-Down | Delete elements list |
| Test data | Element String: ‘el1 el2’ |
| Oracle | The elements list contains the elements ‘el1’ and ‘el2’. |

| Test ID | TC_06 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.add_constraints_input |
| Description | Multiple constraints can be added to a list of constraints. |
| Set-up | Create an elements list with three elements<br>Create an empty constraints list |
| Tear-Down | Delete constraints list<br>Delete elements list |
| Test data | Elements: ‘el1’, ‘el2’, ‘el3’<br>Constraint String:  el1 el2 el2 el3 |
| Oracle | The constraints list contains the constraints [el1, el2] and [el2, el3]. |

| Test ID | TC_09 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.example_1 |
| Description | Runs the first example |
| Set-up | None |

| | |
|---|---|
| Tear-Down | None |
| Test data | None |
| Oracle | The first example was completed and the graph was printed. |

| Test ID | TC_10 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.example_2 |
| Description | Runs the second example |
| Set-up | PLAIN_TEXT_FILE Example |
| Tear-Down | None |
| Test data | None |
| Oracle | The second example was completed and the output is printed. |

| Test ID | TC_13 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.element_in_list(el_in_list) and ELEMENT_LIBRARY.element_in_list(el_not_in_list) |
| Description | Checks if the elements list contains the input element. |
| Set-up | Create an elements list with 3 elements |
| Tear-Down | Delete elements list |
| Test data | Elements: ‘el1’, ‘el2’, ‘el_in_list’ |

| | |
|---|---|
| Oracle | Returns either true (case 1) or false (case 2) |

| Test ID | TC_14 |
|---|---|
| Requirements under test | None |
| Routines under test | ELEMENT_LIBRARY.constraint_in_list(co_in_list)<br>and<br>ELEMENT_LIBRARY.constraint_in_list(co_not_in_list) |
| Description | Checks if the constraints list contains the input constraint. |
| Set-up | Create an elements list with 4 elements<br>Create an constraints list with 3 constraints |
| Tear-Down | Delete elements list<br>Delete constraints list |
| Test data | Elements: ‘el1’, ‘el2’, ‘el3’, ‘el4’<br>Constraints: ‘co1’, ‘co2’, ‘co_in_list’ |
| Oracle | Returns either true (case 1) or false (case 2) |

| Test ID | TC_07 |
|---|---|
| Requirements under test | 3.2.2.17 |
| Routines under test | ELEMENT_LIBRARY.loop_elements |
| Description | A list with 10 random distinct elements can be constructed. |
| Set-up | Create an empty list of elements |
| Tear-Down | Delete elements list |
| Test data | Input integer: 10 |

| | |
|---|---|
| Oracle | The elements list contains 10 distinct elements. |

| Test ID | TC_08 |
|---|---|
| Requirements under test | 3.2.2.18 |
| Routines under test | ELEMENT_LIBRARY.loop_constraints |
| Description | A list with 10 random distinct constraints can be constructed. |
| Set-up | Create an elements list with 10 elements<br>Create an empty constraints list |
| Tear-Down | Delete constraints list<br>Delete elements list |
| Test data | Integer input: 10<br>Elements list: ‘all_elements’ |
| Oracle | The constraints list contains 10 distinct constraints consisting of elements from a given elements list. |

| Test ID | TC_12 |
|---|---|
| Requirements under test | 3.2.2.12 / 3.2.2.13 / 3.2.2.14 / 3.2.2.15 / 3.2.2.16 |
| Routines under test | ELEMENT_LIBRARY.run_example(n) |
| Description | Runs one of the 5 predefined examples. |
| Set-up | None |
| Tear-Down | Delete elements list<br>Delete constraints list |
| Test data | Predefined |
| Oracle | Each example returns the expected output. |

| Test ID | TC_16 |
|---|---|
| Requirements under test | None |
| Routines under test | TOPOLOGICAL_SORT.has_element |
| Description | Is e one of the elements to be topologically sorted? |
| Set-up | None |
| Tear-Down | None |
| Test data | el:STRING |
| Oracle | TRUE: element is there<br>FALSE: element is not there |

| Test ID | TC_17 |
|---|---|
| Requirements under test | None |
| Routines under test | TOPOLOGICAL_SORT.add_successor |
| Description | y should be noted as successor of x |
| Set-up | None |
| Tear-Down | Add y to the successor list of x. |
| Test data | x:INTEGER<br>y:INTEGER |
| Oracle | y has been added to the successors of x. |

| Test ID | TC_22 |
|---|---|
| Requirements under test | 3.2.2.7 |
| Routines under test | TOPOLOGICAL_SORT.process |
| Description | Correct topological sort returning a list of sorted elements and a list with cycle elements. |
| Set-up | Create an elements list with 6 elements<br>Create a constraints list with 6 constraints |

| | |
|---|---|
| Tear-Down | Delete elements list<br>Delete constraints list<br>Delete sorted list<br>Delete cycle list |
| Test data | Elements: ‘el1’, ‘el2’, ‘el3’, ‘el4’, ‘el5’, ‘el6’<br>Constraints: [el1, el2], [el2, el3], [el3, el4], [el4, el2], [el4, el5], [el5, el6] |
| Oracle | Lists containing the non-cyclical part and the cycle respectively. |

| | |
|---|---|
| Test ID | TC_18 |
| Requirements under test | None |
| Routines under test | TOPOLOGICAL_SORT.record_element |
| Description | Add e to the set of elements |
| Set-up | index_of_element: HASH_TABLE [INTEGER, STRING]<br><br>element_of_index: ARRAY [STRING] |
| Tear-Down | x has been added into the two lists |
| Test data | None |
| Oracle | x is recorded |

| | |
|---|---|
| Test ID | TC_19 |
| Requirements under test | None |
| Routines under test | TOPOLOGICAL_SORT.record_constraint |
| Description | Add the constaint[e,f] |
| Set-up | index_of_element: HASH_TABLE [INTEGER, STRING] |

| | |
|---|---|
| | predecessor_count: ARRAY [INTEGER] |
| Tear-Down | the predecessor count of y has been extended<br><br>y has been added as successor of x |
| Test data | None |
| Oracle | constraint is recorded |

| Test ID | TC_20 |
|---|---|
| Requirements under test | None |
| Routines under test | TOPOLOGICAL_SORT.find_initial_candidates |
| Description | Insert into candidates any elements without predecessors |
| Set-up | candidates: LINKED_PRIORITY_QUEUE [INTEGER]<br><br>processed_count:INTEGER |
| Tear-Down | None |
| Test data | None |
| Oracle | All the candidates were found. |

| Test ID | TC_20 |
|---|---|
| Requirements under test | None |
| Routines under test | TOPOLOGICAL_SORT.report_cycles |

| | |
|---|---|
| Description | Prints out if a cycle was found during the sort (cf. 3.2.2.7) |
| Set-up | Set new boolean cycle_found to TRUE |
| Tear-Down | Delete boolean |
| Test data | None |
| Oracle | String printed into console. |

| Test ID | TC_23 |
|---|---|
| Requirements under test | None |
| Routines under test | GRAPH.choose_graph |
| Description | Choosing desired action of either showing a graph or a cycle |
| Set-up | Empty linked list<br>New integer |
| Tear-Down | Delete linked list<br>Delete integer |
| Test data | Integer: 1 |
| Oracle | Correctly called next routine. |

| Test ID | TC_24 |
|---|---|
| Requirements under test | 3.2.2.6 |
| Routines under test | GRAPH.show_graph |
| Description | Correct graphical representation of a topologically sorted list. |
| Set-up | Create an elements list with 6 elements<br>Create a constraints list with 6 constraints |

| | |
|---|---|
| | Topological sort of the lists |
| Tear-Down | Delete elements list<br>Delete constraints list<br>Delete list of sorted elements<br>Delete cycle list |
| Test data | Elements: ‘el1’, ‘el2’, ‘el3’, ‘el4’, ‘el5’, ‘el6’<br>Constraints: [el1, el2], [el2, el3], [el3, el4], [el4, el2], [el4, el5], [el5, el6] |
| Oracle | Correctly displayed graph. |

| | |
|---|---|
| Test ID | TC_25 |
| Requirements under test | None |
| Routines under test | GRAPH.show_cycle |
| Description | Correct graphical representation of an occurring cycle in the topological sorting process. |
| Set-up | Create an elements list with 6 elements<br>Create a constraints list with 6 constraints<br>Topological sort of the lists |
| Tear-Down | Delete elements list<br>Delete constraints list<br>Delete list of sorted elements<br>Delete cycle list |
| Test data | Elements: ‘el1’, ‘el2’, ‘el3’, ‘el4’, ‘el5’, ‘el6’<br>Constraints: [el1, el2], [el2, el3], [el3, el4], [el4, el2], [el4, el5], [el5, el6] |
| Oracle | Correctly displayed cyclic graph. |

| | |
|---|---|
| Test ID | TC_14 |
| Requirements under test | None |
| Routines under test | ELEMENT.make |
| Description | Create a single element. |

| | |
|---|---|
| Set-up | None |
| Tear-Down | Delete the elements list |
| Test data | None |
| Oracle | There is a new element object. |

| Test ID | TC_15 |
|---|---|
| Requirements under test | None |
| Routines under test | CONSTRAINT.constraint |
| Description | Create a single constraint |
| Set-up | None |
| Tear-Down | Delete elements list<br>Delete constraints list |
| Test data | None |
| Oracle | There is a new constraint object. |

# 3 Expected Coverage

Functional coverage:
24 tested functions, 1 untested (96% coverage).

Statement coverage:
By running anything in the program, most statements are naturally covered, which leads to our tests making use of a bigger part of the statements. Expected statement coverage is around 80%

Branch Coverage and Condition Coverage are for some parts given by the statement coverage, but were not a focus point of the tests. Having no concrete tests for the coverage we expect it to be on a rather low point at 30-40%.

# 4 Benchmarks

Speed benchmark (mainly example 5):

| | |
|---|---|
| Test ID | TC09, TC10, TC12 |
| Requirements under test | 3.2.2.12 / 3.2.2.13 / 3.2.2.14 / 3.2.2.15 / 3.2.2.16 |
| Routines under test | Element_library.run_example(n) |
| Description | Runs one of the 5 predefined examples. |
| Set-up | None |
| Tear-Down | Delete elements list<br>Delete constraints list |
| Test data | Predefined |
| Oracle | Each example returns the expected output. |

Since the examples are rather long by definition, we can use them for running speed benchmarks.