Contents

Revision History

Date	Version	Description	Author(s)
25.09.2018	0.1	Initializing document structure	Reto Schönenberger
02.10.2018	0.2	Adding core requirements	Michael Blum
05.10.2018	0.3	Revision of requirements	Yannick Trunz
08.10.2018	0.4	Reworked Glossary and Introduction	Reto Schönenberger
09.10.2018	0.5	Reworked Glossary	Reto Schönenberger
12.10.2018	0.6	Final Revision	Yannick Trunz / Michael Blum

1 Introduction

1.1 Purpose

This document represents the Software Requirements Specification (SRS) for the Eiffel and the example. It describes scope of the system, both functional. It describes scope of the system, both functional and non-functional requirements for the software and design constraints.

1.2 Scope

AutoTasks which allows you to specify tasks and ordering *task orderings* compatible with the constraints. To create these task orderings, a topological sort will be implemented. To create these task orderings, a topological sort will be implemented.
It also describes the five example..
Communication between the people involved in this project will not be part of this specification.

1.3 Definitions, Acronyms and Abbreviations

Word/ Term	Explanation
Library	A collection of software or data reflecting a specific theme
AutoTasks	Library created with Eiffel
Eiffel	Object-oriented programming language designed by Bertrand Meyer
Constraints	Two elements dependant in the form of <e1, e2> for two elements e1 and e2, where e2 is dependent on e1.
Project	AutoTasks[2][2]
GUI	Responsible for displaying all the relevant information for the user
User	People interacting with the system
Developer	Students working on this project
Element	An integer or a string
String	Any finite sequence of characters (i.e., letters, numerals, and punctuation marks)
List	A list which is empty, contains one or multiple elements.
Message	An output of the program telling the user something about the actual state of execution.
Topological order	A specific order of elements which considers all the relevant constraints
Topological sort	An algorithm that takes as arguments the different elements and constraints and produces a topological order of those elements.
Topological sort object	A topological sort object is represented by its name and contains a list of elements and a list of constraints.

1.4 Overview

Chapter 2 defines the general product functions, intended application, constraints to be respected and the assumption made to define requirements.

Chapter 3 specifies functional[obj]level of detail enough to enable designers to design a system to satisfy these requirements. level of detail enough to enable designers to design a system to satisfy these requirements.

The structure and format of this document was chosen according to the *IEEE Std 830-1998* standard[1], as well as the template_SRS_UZH document published on OLAT.

2 Overall Description

2.1 Product Perspective

The product *AutoTasks* to specify tasks and ordering constraints between them and to produce task orderings compatible with the constraints. There is implemented to create the task orderings. topological sort implemented to create the task orderings.
The product also includes five example using the library using the library:

- One of the programs must cover the example Rosetta Code[3]..
- One of the programs should be a simple make file.
- Three of the examples are just lists of elements and constraints entered in the library. They differ in the number of elements and constraints.

2.2 Product Functions

The *AutoTasks* supports following functions:

- Entering or removing an element or a constraint
- Entering or removing lists of elements or constraints
- Display graphical representation of given elements and constraints
- Produce topological sort of given elements and constraints.
- Document possible cycles
- Number of examples

A more detailed explanation of the functionalities can be found in the section 3.

2.3 User Characteristics

The intended users of the system are the teaching assistants, professors and developers. These stakeholders have a certain amount of knowledge about this project and are familiar with the project description. This system is not designed for people outside of this project.

2.4 Constraints

The document represents a study project It gives only directions and requirement templates for creating an Eiffel..
There is also the possibility of upcoming changes in the projectcannot be considered in this document. be considered in this document.

2.5 Assumptions and dependencies

It is assumed that the developers and software design provided in the lectures or in the exercise sessions. The developers can deliver a product at the end of this project, that satisfies the professors expectations. and software design provided in the lectures or in the exercise sessions. The developers can deliver a product at the end of this project, that satisfies the professors expectations.

3 Specific Requirements

3.1 Functional Requirements

3.1.1 Library Requirements

Property	Description
ID	Gives every single requirement a unique number.
Title	The title describes the requirement.
Description	Describes what a requirement stands for in the context of the whole system.
Priority	Shows the priority of the different requirements, 1 is the highest and 3 is the lowest priority. Requirements with priority 1 and 2 are essential for the implementation. Requirements with priority 3 are optional features, which are not essential for the implementation.
Reference	Shows if there are any references in the description to other requirements.
Input	The appropriate input, so the system can work without any problems.
Output	The correct output for a certain task

ID	3.1.001
Title	Enter an element
Description	The user shall be able to add a new element to the list of elements.
Priority	1
Reference	None
Input	Single element of type string. The input can’t be a duplicate of an existing element from the list.
Output	Short confirmation message if input is valid. Error message if input got rejected.

ID	3.1.002
Title	Enter a constraint
Description	The user shall be able to add a new constraint to the list of constraints. The constraint of the form (e1, e2) follows the convention specified under 3.2.006
Priority	1
Reference	3.2.006
Input	Two elements of type string: The two elements can't be the same. Both elements must exist in the list of elements. The constraint can't already exist.
Output	Short confirmation message if input is valid. Error message if input got rejected.

ID	3.1.003
Title	Remove an element
Description	The system shall allow a user to remove a previously added element (3.1.001).
Priority	1
Reference	3.1.001
Input	A single element has to exist in the list of elements.
Output	Short message that the element has been removed successfully if input is valid. Error message if input got rejected.

ID	3.1.004
Title	Remove a constraint
Description	The system shall allow a user to remove a previously added constraint (3.1.002).
Priority	1
Reference	3.1.002
Input	Two elements which are of type string. The input has to be a duplicate of an existing constraint.
Output	Short message that the constraint has been removed successfully if input is valid. Error message if input got rejected.

ID	3.1.005
Title	Enter a list of elements
Description	The user shall be able to add multiple elements to the list of elements by passing a list. The system will add every single element (3.1.001) of that list, every elements of the list has to be a valid input.
Priority	1
Reference	3.1.001
Input	List of elements.
Output	Short message after input got processed.

ID	3.1.006
Title	Enter a list of constraints
Description	The user shall be able to add multiple constraints to the list of constraints by passing a list. The system will add every single constraint (3.1.002) of that list, every constraint of the list has to be a valid input.
Priority	1
Reference	3.1.002
Input	List of constraints.
Output	Short message after input got processed.

ID	3.1.007
Title	Remove a list of elements
Description	The user shall be able to remove multiple elements from the list of elements by passing a list. The system will remove every single element (3.1.003) from that list, every element of the list has to be a valid input.
Priority	1
Reference	3.1.003
Input	List of elements
Output	Short message after input got processed.

ID	3.1.008
Title	Remove a list of constraints.
Description	The user shall be able to remove multiple constraints from the list of constraints by passing a list. The system will remove every single constraint (3.1.004) from that list, every constraint has to be a valid input.
Priority	1
Reference	3.1.004
Input	The list of constraints the user wants to remove from the existing number of constraints.
Output	Short message that the list of constraints has been removed successfully.

ID	3.1.009
Title	Display graphical representation of given elements and constraints.
Description	The system shall provide a graphical representation of all the given elements ((3.1.001),(3.1.005)) and the constraints ((3.1.002),(3.1.006)) .For this purpose, the system shall work with Graphviz..
Priority	1
Reference	3.1.001, 3.1.002, 3.1.005, 3.1.006
Input	None
Output	A graphical representation of all the given elements and constraints and how they are connected.

ID	3.1.010
Title	Produce topological sort of given elements and constraints.
Description	The system shall provide a topological order using a topological sort algorithm. The system shall incorporate all given elements and all given constraints ((3.1.001), (3.1.002), (3.1.005), (3.1.006)).
Priority	1
Reference	3.1.001, 3.1.002, 3.1.005, 3.1.006
Input	None
Output	Confirmation message after sorting is done.

ID	3.1.011
Title	Document possible cycles
Description	If there are any cycles in the topological order the system shall document them.
Priority	1
Reference	None
Input	None
Output	Documentation of cycles.

ID	3.1.012
Title	If there are cycles, produce topological sort of the non-cyclical part
Description	If the system detects any cycles within the topological sort (3.1.011) there should be an output just regarding the non-cyclical part of the input.
Priority	1
Reference	3.1.011
Input	None
Output	Topological sort of the non-cyclical part

ID	3.1.013
Title	Shows all constraints
Description	The system shall allow the user to display all the existing constraints ((3.1.002), (3.1.006)).
Priority	2
Reference	3.1.002, 3.1.009
Input	None
Output	A list of all the existing constraints regarding this system.

ID	3.1.014
Title	Multiple solutions
Description	Even if there are multiple possible solutions for a certain input the system shall always produce the same result for the same elements and same constraints.
Priority	1
Reference	None
Input	None
Output	None

ID	3.1.015
Title	Create new topological sort
Description	The user shall be able to create new topological sort objects, which have their own list of elements and constraints.
Priority	3
Reference	None
Input	String, which will be the name of the new topological sort object. A name which already represents another object is considered invalid.
Output	Short message that the new object has been successfully created if input is valid. Error message if input got rejected.

ID	3.1.016
Title	Delete existing topological sort object
Description	The user shall be able to delete an existing topological sort object.
Priority	3
Reference	None
Input	Name of topological sort to delete.
Output	Short message that the object has been successfully removed if input is valid. Error message if input got rejected.

ID	3.1.017
Title	List all existing topological sort objects
Description	The user shall be able to list all existing topological sort objects.
Priority	3
Reference	None
Input	None
Output	List of all existing topological sort objects.

3.1.2 Examples Requirements

ID	3.1.018
Title	Number of examples
Description	There should be 5 pre-defined examples. They test whether the system works properly or not.
Priority	2
Reference	None
Input	None
Output	None

ID	3.1.019
Title	Example from Rosetta Code
Description	This example is from the rosetta code website and checks if the system works properly and produces the correct output for the given input.
Priority	2
Reference	Rosetta Code[3]
Input	All the elements and constraints given on the defined website
Output	All the element ordered in topological order

ID	3.1.020
Title	Example with 10 constraints
Description	In this example are 4 elements with 10 constraints. This example should test if the system works with a relatively small number of elements and constraints as input.
Priority	2
Reference	None
Input	4 elements, 10 constraints
Output	4 elements ordered in topological order

ID	3.1.021
Title	Example with 1’000 constraints
Description	In this example are 200 elements with 1’000 constraints. This example should test if the system works with a medium amount of elements and constraints as input.
Priority	2
Reference	None
Input	200 elements, 1’000 constraints
Output	200 elements ordered in topological order

ID	3.1.022
Title	Example with 100’000 constraints
Description	In this example are 2^000 elements with 100’000 constraints. This example should test if the system works with a relatively big number of elements and constraints as input.
Priority	2
Reference	None
Input	2’000 elements, 100’000 constraints
Output	2’000 elements ordered in topological order

ID	3.1.023
Title	Same convention
Description	This example should be a very simple make file. The inputs are different lines in form of: • $elem_0$: $elem_1$, $elem_2$, $elem_n$ This line means that $elem_0$ depends on $elem_1$, $elem_2$, $elem_n$. The output should contain all elements in the input in such an order that all the constraints are considered and the output is in correct topological order.
Priority	2
Reference	None
Input	A certain number of lines as described in the description.
Output	The files ordered in topological order.

ID	3.1.024
Title	Convention of Notation
Description	All the examples in 3.2.003, 3.2.004, 3.2.005 should have the same convention regarding the notation of the constraints. i.e. • (23,25) which means that 23 has to occur before 25 occurs.
Priority	2
Reference	3.2.002, 3.2.003, 3.2.004, 3.2.005
Input	None
Output	None

3.2 Performance Requirements

ID	3.3.001
Title	Capacity
Description	The library must be able to hold at least 100’000 constraints and 2000 elements.
Priority	2
Reference	3.1.022
Input	None
Output	None

ID	3.3.002
Title	Running Time
Description	The implementation of the topological sort *O(n)*..
Priority	2
Reference	None
Input	None
Output	None

3.3 Maintainability

ID	3.4.001
Title	VariableLabeling
Description	Each variable should be named in a suggestive manner.
Priority	2
Reference	None
Input	None
Output	None

ID	3.4.002
Title	ClassLabeling
Description	Each name given to a class or a feature has to clearly identify its meaning and suggest its behaviour. Comments should be present to clarify meanings when names do not suffice. All comments shall be written in English.
Priority	2
Reference	None
Input	None
Output	None

3.4 Design Constraints

ID	3.5.001
Title	Implementation Language
Description	The library should be implemented in the programming language Eiffel.
Priority	1
Reference	None
Input	None
Output	None

ID	3.5.002
Title	Design Criteria
Description	All code follow the style guidelines.
Priority	2
Reference	3.5.001 Eiffel[5][5]
Input	None
Output	None

4 Supporting Information

4.1 References

[1] Project_description.pdf on Olat
[2] IEEE Std 830-1998, Recommended Practice for Software Requirementsfications, revision of IEEE Std 830-1993.cations, revision of IEEE Std 830-1993.
[3] https://rosettacode.org/wiki/Topological_sort
[4] http://graphviz.org/
[5] https://www.eiffel.org/doc/eiffel/Style_Guidelines

Index