**Contents**

[1 Introduction 2](#_Toc499808838)

[1.1 Purpose 2](#_Toc499808839)

[1.2 Scope 2](#_Toc499808840)

[1.3 Abbreviations 2](#_Toc499808841)

[1.4 Glossary 2](#_Toc499808842)

[1.5 References 3](#_Toc499808843)

[1.6 Overview 3](#_Toc499808844)

[2 Overall Description 3](#_Toc499808845)

[2.1 Product Perspective 3](#_Toc499808846)

[2.2 Product Functions 3](#_Toc499808847)

[2.3 User Characteristics 3](#_Toc499808848)

[2.4 Constraints 4](#_Toc499808849)

[2.5 Assumptions and Dependencies 4](#_Toc499808850)

[3 Specific Requirements 4](#_Toc499808851)

[3.1 External Interfaces 5](#_Toc499808852)

[3.2 Functional Requirements 5](#_Toc499808853)

[3.3 Non-Functional Requirements 9](#_Toc499808854)

[3.4 Performance Requirements 10](#_Toc499808855)

[3.5 Maintainability 10](#_Toc499808856)

[3.6 Design Constraints 11](#_Toc499808857)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author(s)** |
| 2017-10-05 | 0.1 | Requirements added | Benjamin Solenthaler |
| 2017-10-05 | 0.2 | Requirements added | Elfat Esati |
| 2017-10-05 | 0.3 | Revision | Benjamin Solenthaler |
| 2017-10-05 | 0.4 | Revision | Elfat Esati |
| 2017-10-06 | 0.5 | Part ‘Overall Description’ added | Gian Raphael Prinz |
| 2017-10-06 | 0.6 | Chapter ‘Introduction’ added | Severin Plüss |
| 2017-10-08 | 0.7 | Revision | Gian Raphael Prinz |
| 2017-10-10 | 1.0 | Revision | Severin Plüss |
| 2017-10-13 | 1.1 | Revision | Gian Raphael Prinz |
| 2017-10-15 | 1.2 | Revision | Severin Plüss |
| 2017-11-24 | 1.3 | Revision | Elfat Esati |
| 2017-11-30 | 1.4 | Revision | Gian Raphael Prinz |

# **Introduction**

## **Purpose**

This document delineates the Software Requirements Specification (SRS) of an Eiffel library called «Markup Generator». The inherent project was initiated by a software construction course held at University of Zurich. The document not only describes requirements but also design constraints, system interfaces and the performance. It strongly follows the structure defined by the IEEE 830‐1998 standard [1] for specifying software requirements.

## **Scope**

The «Markup Generator» can be used by software developers to create documents in markup languages, especially HTML. It relieves programmers from the burden to create reports manually. This SRS document focuses on the requirements of the «Markup Generator».. Therefore, external systems are not described within this document.

## **Abbreviations**

|  |  |
| --- | --- |
| **Term** | **Explanation** |
| SRS | Software Requirements Specification |
| HTML | Hypertext Markup Language |
| API | Application Programming Interface |
| URL | Uniform Resource Locator |

## **Glossary**

|  |  |
| --- | --- |
| **Term** | **Explanation** |
| Markup  Generator | Name of the library to be created. |
| User | Is the person who will be using the «Markup Generator» which in this case is the developer himself. |
| Stakeholder | Person with interest or concern in the «Markup Generator». |
| Library | Collection of modules that can be accessed by external programs. |
| API | Interface allowing the communication between different software components. |
| Markdown | Markup language for annotating documents. |
| HTML | Markup language on which websites are based. HTML 5 is the most recent version of HTML language. With new elements, attributes and behavior. |

## **References**

[1] IEEE Computer Society, Software Engineering Technology Committee, Committee, Institute of Electrical, and Electronics Engineers, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. Institute of Electrical and Electronics Engineers, 1998.

## **Overview**

This document consists of three sections. The first chapter introduced the project whereas the second section describes the functions and limitations of the «Markup Generator». It is followed by the third chapter specifying specific requirements (functional as well as non-functional) and describing different system interfaces.

# **Overall Description**

## **Product Perspective**

The «Markup Generator» is an Eiffel library for generating reports written in markup languages. It contains several subsystems to satisfy all requirements. But there are not only interfaces between the subcomponents but also between the library and the user. This user interface will be realized in the form of an API that allows other Eiffel programs to take advantage of the library’s functionalities.

## **Product Functions**

This section gives a general overview on the functionalities provided by the «Markup Generator». A detailed description can be found in the third section (Specific Requirements).

The «Markup Generator»

* allows to create static reports.
* creates reports that are based on markup languages with focus on HTML.
* allows to include existing markup snippets in a report.
* allows to create multipage documents.
* supports a wide range of markup elements.
* is based on the newest markup language specifications.
* has an API.

The «Markup Generator» does not

* support import functionalities.
* allow to embed additional languages except the pure markup code (e.g. JavaScript or CSS).
* provide a graphical user interface.

## **User Characteristics**

The «Markup Generator» is used by people with a strong background in software construction and experience in programming with Eiffel. The library is not designed for people without advanced informatics skills.

## **Constraints**

The SRS document is based on a fictional project description and does not represent real conditions. Thus, it is rather based on assumptions of the project team than requirements elicitation. The same applies to the working schedule. It is not given by real conditions but rather by a project plan created by the teaching assistants and the professor of the software construction course. Other factors like the budget or personal resources would also play an important role in a real project since they influence the entire work process. But in this project, they don’t matter.

The «Markup Generator» will only run on Eiffel platforms. Furthermore, hardware requirements are given.

## **Assumptions and Dependencies**

It is assumed that the user is familiar with the Eiffel programming language, the principle of markup languages on the concept of an API. It is also assumed that the user satisfies minimal hardware and software requirements to run the software that allows viewing the generated reports. The library depends on the Eiffel programming language and its functionalities. Changes in Eiffel can affect the behavior of the report generator.

If user-specific markup code is entered, it is assumed to be valid. Thus, no validation and correction will be performed by the library.

# **Specific Requirements**

Under this section both functional and non-functional requirements will be exposed. To provide clear and simple explanation, the requirements follow the structure described below.

|  |  |
| --- | --- |
| Requirement ID | ID that allows identifying each requirement uniquely. |
| Title | Describes the requirement concise. |
| Description | Defines the requirement in detail. |
| Priority | Shows the order in which requirements should be implemented. Priorities are classified in 3 groups (highest to lowest): 1, 2, and 3. Requirements of   * **priority 1** are mandatory for the first Implementation. * **priority 2** are mandatory for the final Implementation. * a **priority greater or equal than 3** represent optional features. |
| Risk | Specifies the risk of not implementing the requirement. It tells how critical the requirement is to the system as a whole. The following risk levels are defined over the impact of not being implemented correctly.   * **Critical** (C): It will break the main functionality of the system. The system cannot be used if this requirement is not implemented. * **High** (H): It will impact the main functionality of the system. Some function of the system could be inaccessible, but the system can be generally used. * **Medium** (M): It will impact some system features, but not the main functionality. The system can still be used with some limitation. * **Low** (L): The system can be used without limitation, but with some workarounds. |
| References | The IDs of any requirement that is relevant in this context is listed here. |

## **External Interfaces**

All the functions provided by the «Markup Generator» should be available to its users in form of an API: It’s the users’ task to generate reports and to determine what information they retrieve form the documents.

## **Functional Requirements**

* + 1. **General**

|  |  |
| --- | --- |
| Requirement ID | R3.2.1.001 |
| Title | Inexplicit Usage of Functions |
| Description | The functions within the software library should be used within the program body, without defining them explicitly. |
| Priority | 1 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.1.002 |
| Title | System |
| Description | The library should be usable from within Eiffel projects. |
| Priority | 1 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.1.003 |
| Title | Creating Documents |
| Description | The user should be able to generate a document. |
| Priority | 1 |
| Risk | C |
| References |  |

* + 1. **Presentation Semantic**

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.001 |
| Title | Internal Links |
| Description | The library should be able to generate links that can be placed in the document and are referencing to other parts of the same document. The user can specify the link text. |
| Priority | 1 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.002 |
| Title | Linking Multiple Reports |
| Description | The library should be able to generate links that can be placed in the document and are referencing to external documents on the same computing device. The user can specify the link text. |
| Priority | 2 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.003 |
| Title | Linking Documents / Web Pages Accessible through the Web |
| Description | The library should be able to generate links that can be placed in the document and are referencing to documents and web pages that are accessible through the web. The user can specify the link text. |
| Priority | 2 |
| Risk | M |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.004 |
| Title | Referencing Images by URL |
| Description | It should be possible to embed images in the report by using a remote URL. |
| Priority | 2 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.005 |
| Title | Generating Images from Local Files |
| Description | An image should be generated by using a local file reference. It should also be possible to specify an alternative text that is displayed if the image cannot be shown. |
| Priority | 2 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.006 |
| Title | Creating Bullet Lists |
| Description | The user can create a bullet list of entries inside the document. At least unordered lists should be supported. The implementation of ordered lists and other types is not a mandatory requirement. |
| Priority | 2 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.007 |
| Title | Creating Tables |
| Description | A document should contain data displayed in a two-dimensional table. Both, normal as well as header cells are supported. |
| Priority | 2 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.008 |
| Title | Text |
| Description | A document should be able to include unformatted text. |
| Priority | 1 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.009 |
| Title | Headers |
| Description | It is possible to define headers of different size to mark sections. At least three levels of headers should be supported. |
| Priority | 2 |
| Risk | H |
| References | R3.2.1.008 |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.010 |
| Title | Including Existing Code |
| Description | A document should be able to include existing code. |
| Priority | 2 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.2.011 |
| Title | Reading from File |
| Description | The system should be able to read content from a file to display its content as unformatted text in the report. |
| Priority | 3 |
| Risk | L |
| References |  |

* + 1. **Document Structure**

|  |  |
| --- | --- |
| Requirement ID | R3.2.3.001 |
| Title | Document Name |
| Description | A document should be able to have a name that can be determined by the user. |
| Priority | 3 |
| Risk | L |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.3.002 |
| Title | Indentation |
| Description | Standard indentation should be used to make the text readable, neat and pretty to the user (formatter will keep spaces and tabs between content elements). |
| Priority | 2 |
| Risk | M |
| References |  |

* + 1. **Output**

|  |  |
| --- | --- |
| Requirement ID | R3.2.4.001 |
| Title | Output to HTML (Clear Text) |
| Description | A document should be presented to the user in clear text that’s adhering to the HTML5-Standard. |
| Priority | 2 |
| Risk | H |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.3.002 |
| Title | HTML Output to Local File |
| Description | The library should be able to output the report’s content in form of HTML files that are saved locally. The main file should be named index whereas all other files should get the name of its title. |
| Priority | 3 |
| Risk | L |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.4.003 |
| Title | Output to Markdown (Clear Text) |
| Description | A document should be presented to the user in clear text that’s adhering to the newest Markdown standard. |
| Priority | 3 |
| Risk | L |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.4.004 |
| Title | Markdown Output to Local File |
| Description | The library should be able to output the report’s content in form of Markdown files that are saved locally. The main file should be named index whereas all other files should get the name of its title. |
| Priority | 3 |
| Risk | L |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.2.4.006 |
| Title | Correctly Indented HTML Output |
| Description | A document should have an indented output where children are indented relative to its parent by a globally defined space. |
| Priority | 2 |
| Risk | H |
| References |  |

## **Non-Functional Requirements**

|  |  |
| --- | --- |
| Requirement ID | R3.3.0.001 |
| Title | Usability |
| Description | A developer with programming experience and little exposure to management tools should be able to use all the functions provided with minimal effort. Thus, the API should be designed as simple as possible. |
| Priority | 3 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.3.0.002 |
| Title | Reliability |
| Description | The system should have a high reliability. If an error occurs, an error will be raised. |
| Priority | 1 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.3.0.003 |
| Title | Scalability |
| Description | The «Markup Generator» should be easily extendable. This has been considered through the design process by using design patterns and other established methods for software construction. |
| Priority | 3 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.3.0.004 |
| Title | Transparency |
| Description | The Eiffel library is simple and transparent. With little effort. a developer can visualize the potential situations it might encounter. |
| Priority | 3 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.3.0.005 |
| Title | Dependency on External Libraries |
| Description | The library should not depend on external libraries. |
| Priority | 3 |
| Risk | C |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.3.0.006 |
| Title | Robustness |
| Description | The Eiffel library is built according to established methods of software construction such that changes in the library have as minimal as possible effects on other system components. |
| Priority | 3 |
| Risk | C |
| References |  |

## **Performance Requirements**

|  |  |
| --- | --- |
| Requirement ID | R3.4.0.001 |
| Title | Error Messages |
| Description | The error messages must be meaningful (if errors need to be handled) to any user that may encounter them, with inclusion of the appropriate action to be taken. |
| Priority | 2 |
| Risk | L |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.4.0.002 |
| Title | Building Time |
| Description | Compiling the library should be as short as possible. Thus, the Eiffel code should be as minimal and simple as possible. |
| Priority | 2 |
| Risk | L |
| References |  |

## **Maintainability**

|  |  |
| --- | --- |
| Requirement ID | R3.5.0.001 |
| Title | Frequent Updates |
| Description | The library must be built in a way that allows updates to be implemented very easily. This has been considered through the design process by using design patterns and other established methods for software construction. |
| Priority | 1 |
| Risk | C |
| References |  |
| Requirement ID | R3.5.0.002 |
| Title | Coding Standards |
| Description | The code should be logical in every aspect so that high quality code is produced. Standard coding conventions ought to be followed. |
| Priority | 1 |
| Risk | C |
| References |  |
| Requirement ID | R3.5.0.003 |

|  |  |
| --- | --- |
| Requirement ID | R3.5.0.003 |
| Title | Naming Conventions |
| Description | We ought to ensure the code is readable and understandable in a long term. To do that standard naming conventions must be followed. |
| Priority | 1 |
| Risk | M |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.5.0.004 |
| Title | Change of Team Members |
| Description | The library has to be built in a way that allows new members of the development team to easily get accustomed to the project and be able to contribute without a long learning phase. |
| Priority | 1 |
| Risk | C |
| References |  |

## **Design Constraints**

The system ought to generate Markup documents. To be more specific it should only produce static sites. The main limitation of the library is that it doesn’t allow producing dynamic sites. In addition to that, it is not allowed to extend the documents’ functionality with external style sheets or the use of other programming languages.

|  |  |
| --- | --- |
| Requirement ID | R3.6.0.001 |
| Title | Dynamic Markup Sites |
| Description | The «Markup Generator» library should only produce static Markup sites. It is not intended for generating dynamic Markup sites. |
| Priority |  |
| Risk | L |
| References |  |

|  |  |
| --- | --- |
| Requirement ID | R3.6.0.002 |
| Title | Programming Language |
| Description | All coding will be done in Eiffel programming language. Other programming languages should not be used. |
| Priority |  |
| Risk | L |
| References | R3.3.1.001 |
| Requirement ID | R3.6.0.003 |
| Title | No Standalone Application |
| Description | The library is a module for the user. It is a base for building further projects, not a standalone application. |
| Priority |  |
| Risk | L |
| References |  |