



**University of  
Zurich<sup>UZH</sup>**

Software Requirement Specification for

# <YODA>

## YOUR OUTRIGHT DOCUMENT ASSEMBLER

---



*Daniela Flüeli  
Joel Barmettler  
Marius Högger  
Spasen Trendafilov*

Supervision:  
Prof. Bertrand Meyer

Software Engineering – Department of Informatics  
University of Zurich

December 17, 2017 | Zurich, Switzerland

---

## Table of Contents

List of Figures.....	IV
List of Tables.....	IV
Revision History.....	1
1 Introduction.....	1
1.1 Purpose.....	1
1.2 Scope .....	2
1.3 Definitions, Acronyms and Abbreviations .....	3
1.4 References .....	5
1.4.2 Documents.....	5
1.4.2 Websites .....	5
1.5 Overview.....	6
1.5.1 Section 2. Overall Description .....	6
1.5.2 Section 3. Specific Requirements .....	6
1.5.3 Section 4. Supporting Information .....	6
2 Overall Description .....	7
2.1 Current Solution .....	7
2.2 Product Perspective.....	7
2.3 Product Functions.....	8
2.3.1 Supported Functions .....	8
2.3.2 Unsupported Functions.....	8
2.4 User Characteristics.....	9
2.5 Constraints.....	9
2.6 Assumptions and dependencies.....	10
3 Specific Requirements .....	10
3.1 Functional Requirements .....	10
3.1.1 Project Related Requirements.....	11
3.1.2 Document Related Requirements .....	12
3.1.3 Element Related Requirements.....	13
3.2 Usability .....	18
3.2.1 Documentation.....	18
3.2.2 Training.....	18
3.2.3 Task Times .....	18
3.2.4 Language.....	18
3.3 Reliability .....	19
3.3.1 Availability .....	19
3.3.2 Error rate .....	19
3.3.3 Error handling.....	19
3.3.4 Security .....	19

---

3.4	Performance Requirements .....	19
3.4.1	Response Time.....	20
3.5	Maintainability .....	20
3.5.1	Corrective Maintenance .....	20
3.5.2	Adaptive Maintenance .....	21
3.5.3	Perfective Maintenance .....	21
3.5.4	Preventive Maintenance .....	23
3.6	Design Constraints.....	24
3.6.1	Software language and Coding.....	24
3.6.2	Development tools .....	24
3.6.3	Architectural and Design Constraints.....	25
3.7	External Interfaces.....	25
3.7.1	User Interfaces .....	25
3.7.2	Software Interfaces .....	26
3.7.3	Communications Interfaces.....	26
4	Supporting Information .....	i
4.1	Requirement index .....	i
4.2	Naming Convention .....	ii
4.3	Comment Convention .....	iii

## List of Figures

FIGURE 1: USAGE OF YODA: PROJECTS GROUP DOCUMENTS TOGETHER, DOCUMENTS THEMSELVES ARE FORMED OUT OF MULTIPLE ELEMENTS. DOCUMENTS FROM THE SAME PROJECT ARE RENDERED INTO A TEMPLATE AND LINKED TOGETHER (IF WISHED), THEN OUTPUTTED TO ONE OF THE SUPPORTED OUTPUT TYPES, LIKE HTML. ....	2
FIGURE 2: ILLUSTRATION OF HOW YODA CREATES ABSTRACT REPRESENTATION OF ELEMENTS AND FEATURES FROM ALL SUPPORTED OUTPUT LANGUAGE, AND THEN ALLOWS TO RENDER THIS ABSTRACTION TO ANY WISHED OUTPUT TYPE EQUALLY. ....	8
FIGURE 3: HOW YODA ALLOWS SIMPLE NESTING INSIDE LISTS, TABLES AND DECORATORS. ....	17

## List of Tables

TABLE 1: LIST OF ALL THE MEETINGS AND REVISION OF THE YODA TEAM.....	1
TABLE 2: DEFINITIONS OF ALL USED TERMS THAT DO NOT BELONG TO COMMON KNOWLEDGE .....	3
TABLE 3: LIST OF ALL DOCUMENTS USED AS REFERENCES IN THE DEFINITIONS .....	5
TABLE 4: LIST OF ALL WEBSITES USED AS REFERENCES IN THE DEFINITIONS .....	5

## Revision History

TABLE 1: LIST OF ALL THE MEETINGS AND REVISION OF THE YODA TEAM

Date 2017	Description	Author(s)
27.9	Kickoff Meeting, Problem identification	Team-Meeting
29.9	UML Class Diagram Prototype	Team-Meeting
1.10	Requirement Analyzation, Description Notes	Team-Meeting
4.10	Diagrams, Descriptions and Tables, References	Team-Meeting
8.10	Reviewing SRS, Dealing with Inconsistency	Team-Meeting
13.10	Reviewing SRS, Final Touches	Team-Meeting
27.10	Reviewing SRS, Overhaul due to major project changes	Team-Meeting
22.11	Reviewing Requirements, adjusting them to the Test Plan	Team-Meeting
1.12	Finalizing SRS for Final Deadline	Team-Meeting

## 1 Introduction

This Introduction provides an overview of the entire Software Requirements Specification (SRS) for the Markup generator YODA (*Your Outright Document Assembler*). First, we specify the purpose of this SRS, highlighting the importance of a complete and comprehensive SRS in terms of saving time and money. Thereafter we provide a brief description of the scope as well as the boundaries of YODA. The next two subsection represents on the one hand a table with definitions, acronyms and abbreviations needed to properly interpret the SRS and on the other hand a complete list of all documents and websites consulted during creating this SRS. The last part of this introduction gives an overview of the further content, structure and organization of the SRS.

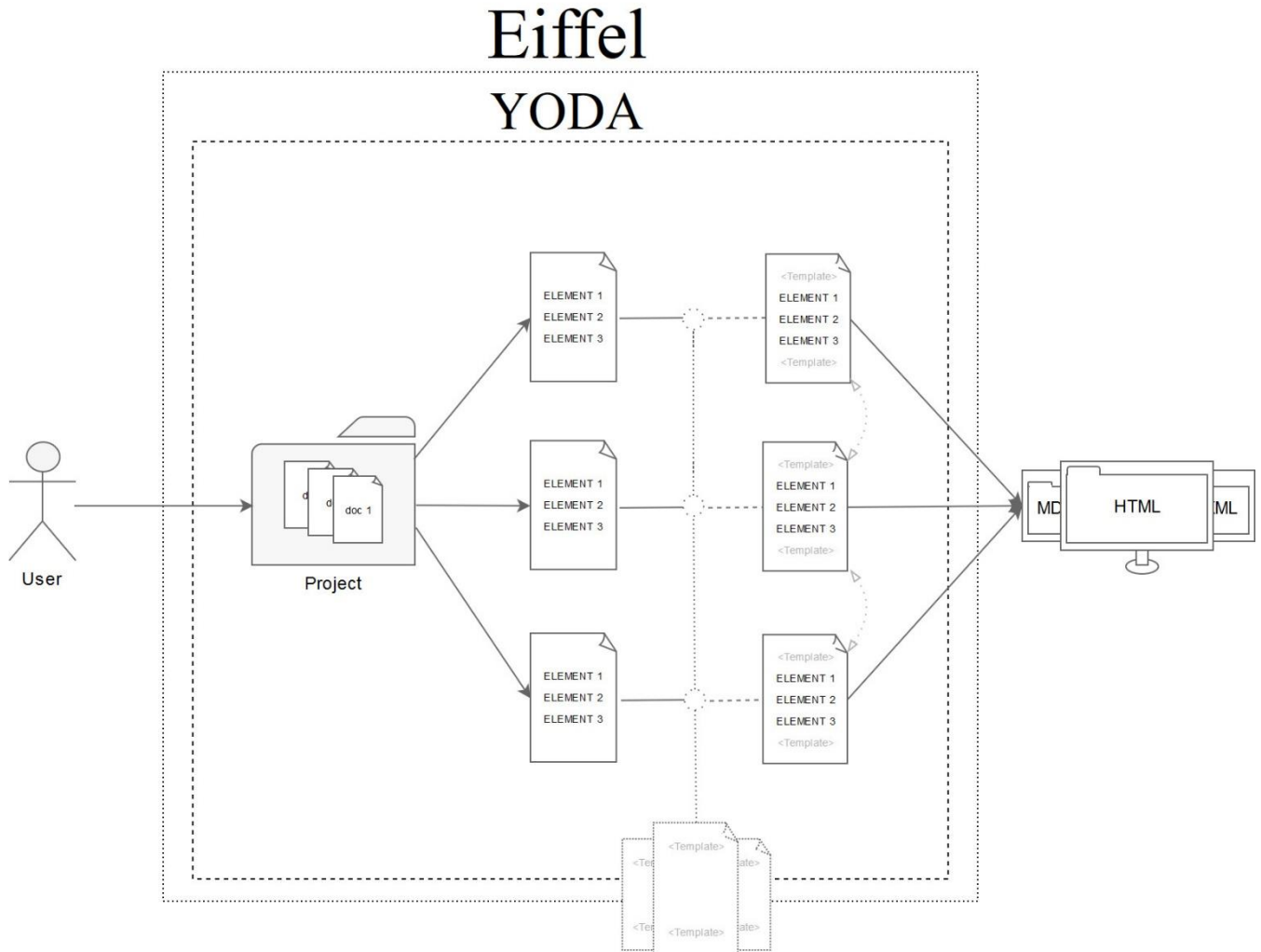
### 1.1 Purpose

Good specifications are essential for a software project to be successful in regard of task execution, teamwork and costs. Since we have the mandate to build a Markup generator named YODA, we wrote this SRS. There are several different stakeholders with different needs involved in the development of our software project. The process of writing a SRS connects these different stakeholders like customers and engineers to establish a common understanding of what YODA should be able to do. With respect to implementation the SRS facilitates the division of labour by providing a reference work. All involved engineers are obliged to consult the SRS to ensure doing the right thing. The SRS may serve to justify towards our stakeholders that the defined goals are met. Explicitly quantified requirements adhering the quality goals of verifiability and traceability help to verify the correctness of the implementation and increase the probability of being achieved. This SRS gives a complete and comprehensive description of YODA including (non-)functional requirements and design constraints. It defines precisely what we are going to build in order to save time, costs and prevent from dissatisfied stakeholders that leads to damage of our image.

## 1.2 Scope

This section includes a brief description of the scope as well as the boundaries of YODA, both described in more detail in section 2.3 Product Functions.

YODA is a Markup-Content generator library, written in Eiffel. YODA relieves the client from the burden of generating syntactically correct Markup-Code from data all on his own.



**FIGURE 1: USAGE OF YODA: PROJECTS GROUP DOCUMENTS TOGETHER, DOCUMENTS THEMSELVES ARE FORMED OUT OF MULTIPLE ELEMENTS. DOCUMENTS FROM THE SAME PROJECT ARE RENDERED INTO A TEMPLATE AND LINKED TOGETHER (IF WISHED), THEN OUTPUTTED TO ONE OF THE SUPPORTED OUTPUT TYPES, LIKE HTML.**

First, a programmer imports YODA to its development environment. Then the client can facilely generate documents in Markup format by using YODA. For this purpose, the client can create an arbitrary number of YODA-Projects and adds one or several YODA-Documents to the YODA-Projects. To add elements as for instance text, lists, tables or images to the YODA-Documents, the client is advised to use YODA's own wrapper-functions to spare himself creating instances of YODA-Elements manually, even though this is a practicable approach, too. The wrapper-functions may be concatenated into each other in order to create different concatenation levels corresponding to the nested structure of Markup-languages. Code snippets already written in a Markup-language can be easily included in the document, too. However, YODA does not do any kind of correctness-check to these imported code snippets, instead the client must ensure that they are of the same Markup-language as the YODA-Project type as well as that they are syntactically correct.

In addition, YODA provides simple styling of text elements such as making part of the text bold, italic or underline. Furthermore, YODA comes with methods to print out all names of YODA-Documents contained in a YODA-Project

as well as all names of YODA-Elements contained in a YODA-Document to the console as a matter of survey. YODA offers the client for some Markup-Languages moreover the ability to add links to YODA-Documents, both external to URLs in the world wide web and internal to other YODA-Documents in the same YODA-Project. When the client has finished adding YODA-Elements to YODA-Documents and YODA-Documents to YODA-Projects, the client can render whole YODA-Projects or single YODA-Documents to one of the supported Markup-languages which will return the Markup-code in string-format to the console. In addition to just rendering the projects and documents, the client is also given the opportunity to save the Markup-code into a file.

However, at time of first release YODA supports the generation of HTML documents only. But its architecture allows high extensibility with respect to output documents written in other Markup-languages as XML or Markdown.

### 1.3 Definitions, Acronyms and Abbreviations

To properly understand the terminology of this SRS, this table shows all titles with the matching descriptions. The reference numbers WX (Websource + and DX) correspond to the following references in the next table later on.

TABLE 2: DEFINITIONS OF ALL USED TERMS THAT DO NOT BELONG TO COMMON KNOWLEDGE

#References	Title	Description
W17	<u>Adaptive Maintenance</u>	An aspect of <u>maintainability</u> , describing the facility of adapting a system prompt and easily to changes in the environment.
	<u>Administrator Rights</u>	Includes the power of decision over the incorporation of commits the composition of the <u>YODA-Administrators</u> the declaration of reported bugs as proven and list them in the according list on the <u>YODA-Bug-Board</u> .
D1	<u>Class</u>	Program-code- <u>template</u> to create objects
W18	<u>Class Diagram</u>	UML <u>class diagram</u> that shows the programmatic structure
	<u>Client</u>	<u>Stakeholder</u> that in some way directly interact with <u>YODA</u>
	<u>Console</u>	Terminal which allows the Developer to Input data into a software and directly receive output.
	<u>Container</u>	Abstract object that contains a certain amount of specific data.
W17	<u>Corrective Maintenance</u>	An aspect of <u>maintainability</u> , describing the facility of bug detection and bug fixing, in case of a malfunction or breakdown of the system.
D3	<u>correctness</u>	<u>Correctness</u> is defined as the software's ability to perform according to its specification
W8	<u>Eiffel</u>	Object oriented programming language
W13	<u>EiffelStudio</u>	Integrated Development Environment for <u>Eiffel</u>
W19	<u>GitHub</u>	Web Based File Hosting-Service used for sharing Software Code
W11	<u>GNU License 2.0</u>	License out of the <u>GNU Library</u> General public license from 1991.
W1	<u>HTML</u>	Hypertext <u>Markup</u> Language used for building Websites
W24	<u>Instance</u>	An <u>instance</u> of a <u>Class</u> is an object, following the architecture defined in the <u>class</u> .
W25	<u>Interface-Segregation Principle</u>	Large interfaces should be split into smaller and more specific ones so that <u>clients</u> will only have to know about the <u>methods</u> that are of interest to them.
W9	<u>Library</u>	A software <u>library</u> is a suite of data and programming code that is used to develop software programs and applications. It is designed to assist both

		the programmer and the programming language compiler in building and executing software.
W26	<u>Maintainability</u>	A design consideration concerning the ease with which <u>YODA</u> can be maintained once it is released and running.
W5	<u>Markdown</u>	A lightweight <u>Markup</u> language with plain text formatting syntax
W20	<u>Markup</u>	A File written in a language following certain rules and using certain keywords that specify behaviour and layouts of the text.
W20	<u>Markup-language</u>	A language that annotates text so that the computer can manipulate that text
	<u>Nesting (Layer)</u>	<u>Nesting</u> describes a recursive structure where objects contain other similar objects. The number of <u>nesting</u> layers describe how many such objects are encapsulated in each other.
W10	<u>Open Source Software</u>	Software that follows certain defined criteria like having accessible source code and allowing giving away software parts through 3. Parties.
	<u>Open-Closed Principle</u>	Modules should be open and closed.
	<u>Output type</u>	The file-type that corresponds to the chosen <u>Markup-language</u> , like “.html”.
	<u>Parsing</u>	Checking input on context <u>correctness</u> and possible causes of failure.
W21	<u>Perfective Maintenance</u>	An aspect of <u>maintainability</u> , describing the extendibility of a system in order to respond to changed <u>stakeholder</u> requirements, both in terms of function and efficiency.
W6	<u>Placeholder Tag</u>	Special Marker in the <u>template</u> that will be recognized and replaced by generated Content.
W21	<u>Preventative Maintenance</u>	An aspect of <u>maintainability</u> , describing the facility of anticipating risks as well as the understandability of a system achievable through consistent coding style and comprehensive <u>documentation</u> .
	<u>Print (to Console)</u>	Commanding the program to output certain information to the <u>console</u> to make it readable for the <u>client</u> .
W22	<u>RAM</u>	Random Access Memory, volatile data storages used in computers
D3	<u>reliability</u>	general term for <u>correctness</u> and <u>robustness</u>
	<u>Rendering</u>	Process of forming abstract data into a visible, usable result.
D3	<u>robustness</u>	<u>Robustness</u> is defined as its ability to react to cases not included in the specification
D3	<u>Single Responsibility Principle</u>	Every module has responsibility over a single part of the functionality provided by the software, so that it has only one reason to change.
D3	<u>Single-Choice-Principle</u>	Whenever a software system must support a set of alternatives, one and only one module in the system should know their exhaustive list.
W3	<u>Snippets</u>	An independent, self-contained piece of text in a <u>Markup-</u> or programming language
	<u>Software Project</u>	A general Project that a Programmer is working on and tries to fulfil.
W16	<u>StackOverflow</u>	Website to learn, share and improve code
W7	<u>Stakeholder</u>	All the people that in any way deal with <u>YODA</u>
W6	<u>Template</u>	Pre-layouted file with pre-existing content and specific <u>Placeholders</u> .
D3	<u>Uniform Access principle</u>	Facilities managed by a module are accessible to its <u>clients</u> in the same way whether implemented by computation or by storage.
	<u>factories</u>	Function that autonomously creates and places objects without forcing the <u>client</u> to deal with <u>instances</u> and creations.



W23	<u>XML</u>	Extensible <u>Markup</u> Language used for structure and format data and information
Name	<u>YODA</u>	<u>Markup</u> generator <u>library</u>
	<u>YODA-Administrators</u>	A subset of the <u>YODA-Community</u> , including testers, managers and the support team with <u>administrator rights</u> and maintenance responsibilities.
	<u>YODA-Bug-Board</u>	A board on the <u>YODA-GitHub-Repository</u> with the objective of bug detecting and communicating. It consists of two lists, one contains supposed bugs reported by the <u>YODA-Community</u> , the other bugs proven by the <u>YODA-Administrators</u> .
	<u>YODA-Community</u>	The collectivity of all <u>YODA-Programmers</u> .
	<u>YODA-Documentation</u>	Readable Text File that ships with <u>YODA</u> and explains all the functionalities and usages of <u>YODA</u> for the identified <u>stakeholders</u> .
UML Diagram	<u>YODA-Element</u>	An <u>instance</u> of the Element <u>Class</u> , which is part of the <u>YODA-Library</u> .
UML Diagram	<u>YODA-Document</u>	An <u>instance</u> of the File <u>Class</u> , which is part of the <u>YODA-Library</u> .
	<u>YODA-Main-Functionality</u>	Guarantees the ability to generate documents in at least one <u>Markup</u> format consisting of at least the most important <u>YODA-Elements</u> as well as the fundamentals of working with <u>YODA-Projects</u> , <u>YODA-Documents</u> and <u>YODA-Elements</u> .
	<u>YODA-Methods</u>	All <u>methods</u> provided by <u>YODA</u> .
	<u>YODA-Programmers</u>	The Programmers working on the <u>open source code</u> of <u>YODA</u> . Forming in a body the <u>YODA-Community</u> .
UML Diagram	<u>YODA-Project</u>	An <u>instance</u> of the Project <u>Class</u> , which is part of the <u>YODA-Library</u> .
	<u>YODA-Requirements-Checklist</u>	Table to facilitate keeping congruence between source code and requirements. One row for each requirement and one column for every update.

## 1.4 References

This Subsection provides a complete list of all documents and websites that we used for this SRS.

### 1.4.2 Documents

TABLE 3: LIST OF ALL DOCUMENTS USED AS REFERENCES IN THE DEFINITIONS

#Doc	Title
D1	110_softcons_intro.pdf – Bertrand Meyer
D2	ex_week2_final.pdf – Tutorial about Git and Eiffel
D3	Object-Oriented Software Construction, second edition – Bertrand Meyer 1988

### 1.4.2 Websites

TABLE 4: LIST OF ALL WEBSITES USED AS REFERENCES IN THE DEFINITIONS

#Web	Title
W1	<a href="https://de.wikipedia.org/wiki/Hypertext_Markup_Language">https://de.wikipedia.org/wiki/Hypertext_Markup_Language</a>
W2	<a href="https://www.w3schools.com/css/css_intro.asp">https://www.w3schools.com/css/css_intro.asp</a>
W3	<a href="https://www.gruenderszene.de/lexikon/begriffe/snippet">https://www.gruenderszene.de/lexikon/begriffe/snippet</a>
W4	<a href="https://www.eiffel.org/doc/eiffel/ET%3A%20Inheritance">https://www.eiffel.org/doc/eiffel/ET%3A%20Inheritance</a>
W5	<a href="https://en.wikipedia.org/wiki/Markdown">https://en.wikipedia.org/wiki/Markdown</a>
W6	<a href="https://de.wikipedia.org/wiki/Webtemplate">https://de.wikipedia.org/wiki/Webtemplate</a>
W7	<a href="http://wirtschaftslexikon.gabler.de/Definition/anspruchsgruppen.html">http://wirtschaftslexikon.gabler.de/Definition/anspruchsgruppen.html</a>
W8	<a href="http://www.eiffel.org">http://www.eiffel.org</a>
W9	<a href="https://www.techopedia.com/definition/3828/software-library">https://www.techopedia.com/definition/3828/software-library</a>
W10	<a href="https://opensource.org/osd">https://opensource.org/osd</a>

---

W11	<a href="https://opensource.org/licenses/LGPL-2.0">https://opensource.org/licenses/LGPL-2.0</a>
W12	<a href="http://www.macmillandictionary.com/dictionary/british/academic_1">http://www.macmillandictionary.com/dictionary/british/academic_1</a>
W13	<a href="https://www.eiffel.com/eiffelstudio/">https://www.eiffel.com/eiffelstudio/</a>
W14	<a href="https://www.w3schools.com/html/">https://www.w3schools.com/html/</a>
W15	<a href="http://www.dictionary.com/browse/encapsulate">http://www.dictionary.com/browse/encapsulate</a>
W16	<a href="https://stackoverflow.com/company">https://stackoverflow.com/company</a>
W17	<a href="http://www.businessdictionary.com/article/599/corrective-vs-adaptive-maintenance-for-your-business/">http://www.businessdictionary.com/article/599/corrective-vs-adaptive-maintenance-for-your-business/</a>
W18	<a href="http://study.com/academy/lesson/what-is-a-uml-class-diagram-definition-symbols-examples.html">http://study.com/academy/lesson/what-is-a-uml-class-diagram-definition-symbols-examples.html</a>
W19	<a href="https://github.com/features#code-review">https://github.com/features#code-review</a>
W20	<a href="https://techterms.com/definition/markup_language">https://techterms.com/definition/markup_language</a>
W21	<a href="http://searchitoperations.techtarget.com/definition/preventive-maintenance">http://searchitoperations.techtarget.com/definition/preventive-maintenance</a>
W22	<a href="https://www.computerlexikon.com/was-ist-ram">https://www.computerlexikon.com/was-ist-ram</a>
W23	<a href="https://wiki.selfhtml.org/wiki/XML">https://wiki.selfhtml.org/wiki/XML</a>
W24	<a href="https://www.codecademy.com/en/forum_questions/558cd3fc76b8fe06280002ce">https://www.codecademy.com/en/forum_questions/558cd3fc76b8fe06280002ce</a>
W25	<a href="http://www.oodeesign.com/interface-segregation-principle.html">http://www.oodeesign.com/interface-segregation-principle.html</a>
W26	<a href="http://www.businessdictionary.com/definition/software-maintainability.html">http://www.businessdictionary.com/definition/software-maintainability.html</a>

---

## 1.5 Overview

The further content, structure and organization of our SRS is abstracted in this section.

### 1.5.1 Section 2. Overall Description

Focuses on general factors affecting YODA which must be incorporated during specifying the requirements.

First, insight into already existing Markup-Content generators is delivered. Followed by a description of the product perspective. The section Product Functions gives an overview of supported and unsupported functions. In addition, it provides a differentiation between shared and specific features of the tree Markup-Languages considered. Then a short section deals with an accurately definition of the YODA-Client. The next section is about constraints for example due to Interfaces to other applications, hardware limitations, regulatory policies or environmental limitations. Finally, some assumptions which underlie the requirements are mentioned as well as some dependencies.

### 1.5.2 Section 3. Specific Requirements

Lists all software requirements defined for YODA as much consistent, complete, precise and concise as possible.

Functional requirements are listed first, partitioned into requirements related to YODA-Projects, YODA-Documents and YODA-Elements. Followed by requirements referring to the usability and reliability of YODA. Performance Requirements as specific response times or resource limitations are described next. Afterwards requirements enhancing the maintainability of YODA are specified. In the end, requirements concerning design constraints due to mandated design decisions as the used programming language are listed.

### 1.5.3 Section 4. Supporting Information

At the very end, an index and an appendix are provided as supporting information in order to make the SRS easier to use. The index lists all requirements by its ID, providing as further information the title as well as the page on which the requirement can be found. Additionally, a summarization of the naming and comment convention described in "Object-Oriented Software Construction" is provided to serve the YODA-Programmers as an overview, but does not replace the reference book "Object Oriented Software Construction".

## 2 Overall Description

The following chapter presents an overall description of the YODA including current solution, product functions, user characteristics, constraints, assumptions and dependencies. This section does not contain any specific requirements it is meant to outline the background for those requirements.

### 2.1 Current Solution

HTML format and other Markup-languages make data better readable and thus find many applications in software projects. We explain the current solution in the case of HTML. There are multiple ways to include HTML formats into software projects. First, there's the possibility to include raw HTML in a plain string objects, which means that the HTML gets handled as normal string, this however leads to messy code and is very inconvenient to work with. Also, there's the way to exporting the data and then creating an HTML file using any external program or web based service. More convenient is to use a library which handles all the HTML syntax but holds on to the HTML functionality with its nesting ability and styling options. There are already some HTML generating libraries available on the internet and new ones will be written surely, however with YODA we want to solve the problem in our very own way.

### 2.2 Product Perspective

YODA is a library and thus does not include any kind of Graphical User Interface (GUI). When included into a software project, YODA enables functionalities to wrap data from the current software project into a well readable format such as HTML, XML or Markdown without leaving the Eiffel-editor. This allows for simultaneous data processing and data formatting within the editor and the exporting of complete Markup-Files to local repositories.

Since the interface to the library is the Eiffel Programming language, some Eiffel programming knowledge is required to use YODA.

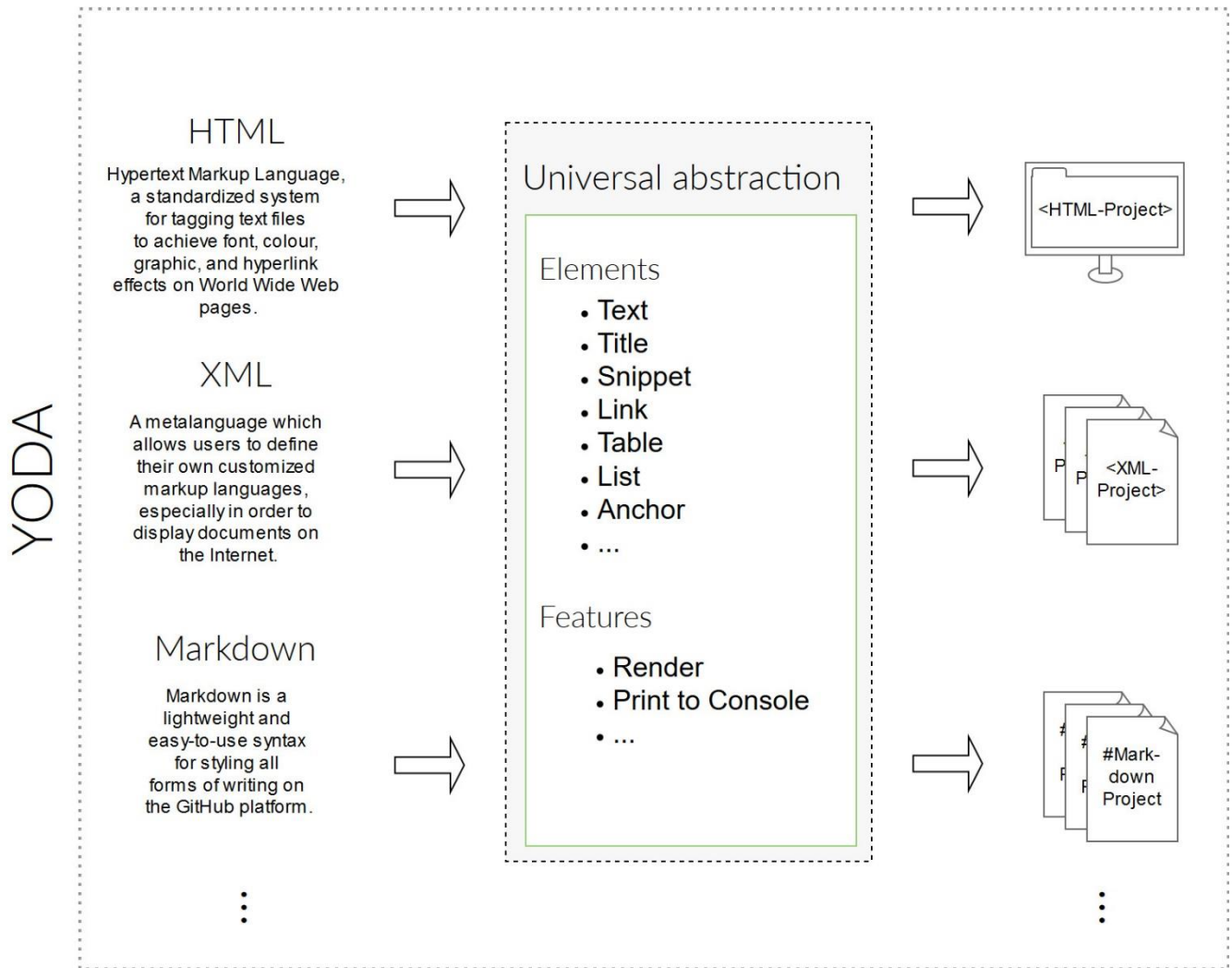
YODA is intended to support the client in formatting data into Markup-format and will generate the Markup-code. YODA emphasizes is to be client friendly and reduce the client's work of instantiation and handling with objects. However, the client shall provide some knowledge on nesting structures or read the YODA-Documentation carefully in order to use the whole potential of YODA.

The memory management is not part of YODA and is incumbent upon the client.

YODA is extensible towards other Markup-languages and new functionalities within the Markup-language.

How YODA is used is not part of this SRS but will be topic of the YODA-Documentation.

## 2.3 Product Functions



**FIGURE 2: ILLUSTRATION OF HOW YODA CREATES ABSTRACT REPRESENTATION OF ELEMENTS AND FEATURES FROM ALL SUPPORTED OUTPUT LANGUAGE, AND THEN ALLOWS TO RENDER THIS ABSTRACTION TO ANY WISHED OUTPUT TYPE EQUALLY.**

### 2.3.1 Supported Functions

YODA does provide

- creating Output for a variety of Markup-languages
- creating a YODA-Project
- creating YODA-Documents and adding them to YODA-Projects
- creating YODA-Elements and adding them to YODA-Documents
- nesting YODA-Elements, if supported by the Markup-language
- easy to use function for creating and nesting YODA-Elements (factories)
- individual YODA-Elements for the basic Markup-language-Elements
- preview of current content in a list format
- rendering a project with all its files and elements to all supported Markup-languages.
- Saving the rendered project to an output file of the according type via the use of a template
- support of HTML as a Markup-language

### 2.3.2 Unsupported Functions

YODA does **not** provide

- any kind of version control for the client,

- code management or code storage,
- parsing,
- the editing the arguments of elements after instantiation,
- saving and exporting of elements outside of the software project,
- any kind of correctness-check on imported code snippets,

## 2.4 User Characteristics

YODA is designed for clients with sophisticated expertise in software development and particularly with software projects. Hence the client shall provide a basic level of programming experience in Eiffel such as using libraries, converting data types as well as handling objects and features. People with no technical understanding and no experience in the use of computers are not the category of people YODA is designed for.

## 2.5 Constraints

Internal and external Effects that limit the usability and reliability of YODA.

### Interfaces to other applications

- YODA requires the installation of Eiffel.

### High order language limitation

- YODA is entirely written in Eiffel 17.05 and can maybe not be fully used in older Eiffel Versions as well as other languages such as C++, Python or Java. YODA's output is constrained by the general output format of the supported Markup-languages.

### Hardware limitation

- YODA is constrained by the fact that it relies on the client to provide the needed read and write permissions to the client's hard disk. In addition, a shortage of the client's RAM and disc space can limit YODA in its capabilities.

### Reliability requirement

- The client is responsible for the content of snippets.
- The speed in which YODA does operate is highly affected by the amount of data the client wants to use.

### Regulatory policies

- YODA is intended to be released as open source to the public, therefore it must fulfil the respective licensing requirement.

### Parallel operation

- YODA allows threading, as long as the individual threads don't conflict with each other, namely claim access to the same variables and files, which can happen while rendering the output files.
- The client needs to make sure all the files and content YODA requires for rendering the project are in a static folder on the local disk and are not in current use of any other program.

### Environmental limitation

- This document as well as YODA itself are part of a one-semester-project, hence time acts as a limiting factor.

### Duration of maintenance

- The current YODA-Administrators take the liberty of limiting the guaranteed maintenance to a year after the first release of YODA. All in section 3.6 defined requirements are restricted by this constraint.

## 2.6 Assumptions and dependencies

YODA's workability highly depends on correct and complete installation and integration into the software project such that all of YODA's source files are stored locally and are accessible by the editor.

YODA's workability depend on the backwards compatibility of future updates of Eiffel. Consequently, it is assumed that the client's operating system supports the use of Eiffel

It is assumed that the general structure and syntax of the supported output types does not change in a way that makes YODA's output unusable over the next 5 years. This assumption is based on the changes over the last 5 years.

All statements made in this SRS are made under the assumption that YODA's source files are not directly modified by the client.

It is assumed that the client possesses a basic knowledge of English.

## 3 Specific Requirements

### 3.1 Functional Requirements

Requirement ID   Title	<b>ID:</b> x.x.x.xx, Unique identifier for each requirement within the SRS document that serves as a group indicator. <b>Title:</b> Individual, meaningful and descriptive name for each requirement, defines group to which the requirement belongs
Reference	Shows relation to other requirements that are relevant in this context.
Description	The definition of the requirement.
Priority   Risk	<b>Priority:</b> Defines in which order the listed requirements should be implemented. Priority ranges from 1 to 3 with 1 being mandatory requirements for the first implementation, 2 being mandatory for the final submission and 3 being optional to implement at all. <b>Risk:</b> States how critical the effect of not implementing the requirement is for the System in order to work correctly and deliver the expected output. The following Risk-Levels are defined: <ul style="list-style-type: none"> <li>• <i>S (Small Risk)</i>: The validation of the requirement will only have a local effect and does not constrain the <u>YODA-Main-Functionality</u>.</li> <li>• <i>M (Medium Risk)</i>: The violation of the requirement will cause side effect to other requirements, but will not constrain the <u>YODA-Main-Functionality</u>.</li> <li>• <i>L (Large Risk)</i>: The violation of the requirement will constrain the <u>YODA-Main-Functionality</u> partially without an imminent risk of a breakdown of <u>YODA</u>.</li> <li>• <i>XL (Extra Large Risk)</i>: The violation of the requirement will constrain the <u>YODA-Main-Functionality</u> profoundly with an imminent risk of a breakdown of <u>YODA</u>.</li> </ul>

### 3.1.1 Project Related Requirements

ID   Title	<b>1.1.1.1   Supported Output Types, first release</b>
Reference	-
Description	The set of given <u>output types</u> for the first release shall consist only of one entry, namely <u>HTML</u> .
Priority   Risk	1   XL

ID   Title	<b>1.1.2.1   YODA-Project, Container of Documents and attributes</b>
Reference	R. 1.2.1.1, R. 1.1.6.1
Description	The <u>client</u> shall be able to create <u>YODA-Projects</u> that serve as a <u>container</u> of related <u>YODA-Documents</u> and project attributes. Each <u>YODA-Project</u> shall have a <u>client</u> -chosen name as an attribute.
Priority   Risk	1   XL

ID   Title	<b>1.1.3.1   Multiple Project Instances</b>
Reference	-
Description	The <u>client</u> shall be able to create an arbitrary number of <u>YODA-Project instances</u> . All <u>YODA-Project instances</u> shall be completely independent from each other.
Priority   Risk	1   L

ID   Title	<b>1.1.4.1   Add YODA-Documents to YODA-Projects</b>
Reference	R. 1.2.1.1, R. 1.1.1.1
Description	For a created <u>YODA-Document</u> , the <u>client</u> shall have the ability to add it to an arbitrary number of <u>YODA-Project instances</u> .
Priority   Risk	1   XL

ID   Title	<b>1.1.4.2   Order of YODA-Documents</b>
Reference	R. 1.1.4.1
Description	The order of the <u>YODA-Documents</u> in the final Output shall be the same as the order in which they were added to the <u>YODA-Project</u> in the program code.
Priority   Risk	1   S

ID   Title	<b>1.1.4.3   Show YODA-Documents in YODA-Project</b>
Reference	-
Description	For each <u>YODA-Project</u> , the <u>client</u> shall be able to <u>print</u> out all names of the <u>YODA-Documents</u> contained in the <u>YODA-Project</u> to the <u>console</u> .
Priority   Risk	2   S

ID   Title	<b>1.1.5.1   Render, YODA-Project, generate output</b>
Reference	-
Description	The <u>client</u> shall have the possibility to <u>render</u> a <u>YODA-Project</u> , meaning every <u>YODA-Document</u> and every <u>YODA-Element</u> , to any of the supported <u>output types</u> . All necessary output data shall be returned as a well formatted string. The string should be formatted in a readable form with correct indentation.
Priority   Risk	3   XL

ID   Title	<b>1.1.6.1   Save <i>YODA</i>-Project to files</b>
Reference	R. 1.2.5.1
Description	A <i>YODA Project</i> shall be saved into a new folder in the working directory, named according to the project name. This folder shall contain all saved <i>YODA-Documents</i> and all the resources if used
Priority   Risk	3   XL

### 3.1.2 Document Related Requirements

ID   Title	<b>1.2.1.1   <i>YODA</i>-Document, Container of <i>YODA</i>-Elements</b>
Reference	-
Description	The <i>client</i> shall be able to create new <i>YODA-Documents</i> , which serve as a <i>container</i> of <i>YODA-Elements</i> . Each <i>YODA-Document</i> shall have a <i>client</i> chosen name for identification purposes.
Priority   Risk	2   L

ID   Title	<b>1.1.2.1   Multiple Document Instances</b>
Reference	-
Description	The <i>client</i> shall be able to create an arbitrary number of <i>YODA-Document instances</i> . All <i>YODA-Document instances</i> shall be completely independent from each other.
Priority   Risk	1   L

ID   Title	<b>1.2.3.1   Add <i>YODA</i>-Elements to <i>YODA</i>-Document</b>
Reference	-
Description	The <i>client</i> shall have the freedom to add <i>YODA-Elements</i> to an arbitrary number of <i>YODA-Document instances</i> .
Priority   Risk	1   XL

ID   Title	<b>1.2.3.2   Allowed <i>YODA</i>-Elements in <i>YODA</i>-Documents</b>
Reference	R. 1.2.3.1
Description	An <i>YODA-Element</i> can be added to a <i>YODA-Document</i> an arbitrary number of times, at arbitrary places.
Priority   Risk	1   L

ID   Title	<b>1.2.3.3   Order of <i>YODA</i>-Elements</b>
Reference	R. 1.2.3.1
Description	The order of the <i>YODA-Elements</i> in the final <i>Output-Document</i> shall be the same as the order in which they were added to the <i>YODA-Document</i> in the program code.
Priority   Risk	1   S

ID   Title	<b>1.2.4.1   Show <i>YODA</i>-Elements in <i>YODA</i>-Document</b>
Reference	R. 1.2.3.1
Description	For each <i>YODA-Document</i> , the <i>client</i> shall be able to <i>print</i> out all names of the <i>YODA-Elements</i> contained in the <i>YODA-Document</i> to the <i>console</i> .
Priority   Risk	2   S



ID   Title	<b>1.2.5.1   Rendering YODA-Documents</b>
Reference	-
Description	Every <u>YODA-Document</u> shall offer the functionality to <u>render</u> itself, meaning to <u>render</u> all its <u>YODA-elements</u> into the <u>client</u> -chosen <u>Output-language</u> .
Priority   Risk	1   XL

ID   Title	<b>1.2.6.1   Save YODA-Document to files</b>
Reference	-
Description	A document of a project shall be saved into the project folder in the working directory. If a document is saved individually a new folder shall be created in the working directory, named according to the document name. This folder shall contain the document-file and resources if used.
Priority   Risk	3   XL

### 3.1.3 Element Related Requirements

ID   Title	<b>1.3.1.1   Text</b>
Reference	-
Description	The <u>client</u> should have the ability to create text-elements, containing text-stings.
Priority   Risk	1   L

ID   Title	<b>1.3.1.2   Text Styling</b>
Reference	-
Description	The <u>client</u> should have the ability to add multiple styling attributes in combination to just parts of a text-element.
Priority   Risk	2   L

ID   Title	<b>1.3.1.3   Text Styling - bold</b>
Reference	-
Description	The <u>client</u> should have the ability to add a styling to a text element to make the text bold.
Priority   Risk	2   L

ID   Title	<b>1.3.1.4   Text Styling - italic</b>
Reference	-
Description	The <u>client</u> should have the ability to add a styling to a text element to make the text italic.
Priority   Risk	2   L

ID   Title	<b>1.3.1.5   Text Styling - underline</b>
Reference	-
Description	The <u>client</u> should have the ability to add a styling to a text element to make the text underlined.
Priority   Risk	2   L

ID   Title	<b>1.3.1.6   Text Styling - code</b>
Reference	-
Description	The <u>client</u> should have the ability to add a styling to a text element to represent the text as a code.
Priority   Risk	2   L

ID   Title	<b>1.3.1.7   Text Styling - quote</b>
Reference	-
Description	The <u>client</u> should have the ability to add a styling to a text element to represent the text as a quote.
Priority   Risk	2   L

ID   Title	<b>1.3.1.8   Text Styling - title</b>
Reference	-
Description	The <u>client</u> should have the ability to add a styling to a text element to represent a title, the <u>client</u> shall also have the ability to add different levels of titles.
Priority   Risk	2   L

ID   Title	<b>1.3.1.9   Text, handling prohibited strings as input</b>
Reference	R. 1.3.5.1
Description	The <u>client's</u> text input shall not have an impact on the output document's look, so <u>YODA</u> shall modify input text and exclude all words that are part of the output-document-types language.
Priority   Risk	1   XL

ID   Title	<b>1.3.2.1   Code Snippet from File</b>
Reference	-
Description	The <u>client</u> shall have the ability to insert his own code into the document, he should therefore choose a file that contains a well-formatted, syntactically correct <u>code-snippet</u> , which content will then be inserted into the <u>YODA-Document</u> .
Priority   Risk	1   L

ID   Title	<b>1.3.2.2   Code Snippet from String</b>
Reference	-
Description	The <u>client</u> shall have the ability to insert his own code into <u>YODA</u> , he should therefore write a syntactically correct <u>code-snippet</u> into <u>EiffelStudio</u> .
Priority   Risk	1   L

ID   Title	<b>1.3.2.3   Code Snippet – not parsed</b>
Reference	R. 1.3.7.1
Description	The <u>snippet</u> shall not be <u>parsed</u> or processed to prevent errors.
Priority   Risk	1   XL

ID   Title	<b>1.3.3.1   Anchor element</b>
Reference	
Description	The <u>client</u> shall be able to place anchor elements, which can be linked to by the anchor link. The anchor element has a <u>client</u> given id.
Priority   Risk	2   M

ID   Title	<b>1.3.4.1   Link</b>
Reference	-
Description	A link shall be placed around another <u>YODA-Element</u> which then becomes clickable.
Priority   Risk	2   M

ID   Title	<b>1.3.4.2   Link - extern</b>
Reference	R. 1.3.4.1
Description	The <u>client</u> shall have the ability to add links to an external URL in the world wide web.
Priority   Risk	2   M

ID   Title	<b>1.3.4.3   Link - intern</b>
Reference	R. 1.3.4.1
Description	The <u>client</u> shall have the ability to add links to his <u>YODA-Documents</u> .
Priority   Risk	2   M

ID   Title	<b>1.3.4.4   email</b>
Reference	R. 1.3.4.1
Description	The <u>client</u> shall have the ability to add links to an mailto of a given email address.
Priority   Risk	2   M

ID   Title	<b>1.3.4.5   Anchor link</b>
Reference	R. 1.3.3.1
Description	The <u>client</u> shall have the ability to link to an Anchor element in the same document. When the anchor-link is clicked the current view shall jump to the anchor point.
Priority   Risk	2   M

ID   Title	<b>1.3.5.1   Image, extern resource</b>
Reference	-
Description	The <u>client</u> shall be able to add an image to his document that is stored externally on the web using a static URL.
Priority   Risk	1   L

ID   Title	<b>1.3.5.2   Image, local resource</b>
Reference	-
Description	The <u>client</u> shall be able to add an image to his document that is stored locally.
Priority   Risk	1   L

ID   Title	<b>1.3.5.3   Image, storage in resources folder</b>
Reference	-
Description	For an image that is stored locally <u>YODA</u> shall create a copy of the image-file and store it in the resource folder. For the usage of the images that new file in the resource folder shall be used.
Priority   Risk	3   XL

ID   Title	<b>1.3.6.1   List</b>
Reference	-
Description	The <u>client</u> shall be able to add lists to his files. Lists are either numbered or bullet pointed, but never both at the same time.
Priority   Risk	2   L

ID   Title	<b>1.3.7.1   Table</b>
Reference	-
Description	The <u>client</u> shall be able to insert tables with freely choosable content into his <u>YODA-Documents</u> . The <u>client</u> provides two-dimensional data containing <u>YODA-Elements</u> , which will then be displayed in the individual cells of the table.
Priority   Risk	2   L

ID   Title	<b>1.3.8.1   YODA-Element validation</b>
Reference	-
Description	For each composition of <u>YODA-Elements</u> , <u>YODA</u> shall validate whether it is valid for all supported output languages. A <u>nested</u> composition of <u>YODA-Elements</u> is valid for an output language when it can be directly represented by that output language or when the <u>YODA Renderer</u> for that language is able to create an alternative representation for it.
Priority   Risk	1   XL

ID   Title	<b>1.3.9.1   Not shared YODA-Element representation</b>
Reference	R. 1.3.8.1
Description	For the most important, not shared features of the output-document languages, there shall exist a corresponding <u>YODA-Element-Interpretation</u> representing that feature for the output languages that do not directly support that element. Such Interpretations can be combinations of supported Elements, own representations that imitate the original representation that is supported by another output language or in rare occasions that representation can be empty.
Priority   Risk	1   XL

ID   Title	<b>1.3.10.1   Render YODA-Elements</b>
Reference	-
Description	Each <u>YODA-Element</u> shall offer the functionality of being <u>rendered</u> , meaning to be outputted as a proper string-based representation in the chosen output language. Whenever a certain <u>nested</u> element composition is not directly supported by the chosen output language, <u>YODA</u> shall <u>render</u> the element composition in an alternative, acceptable way.
Priority   Risk	1   XL

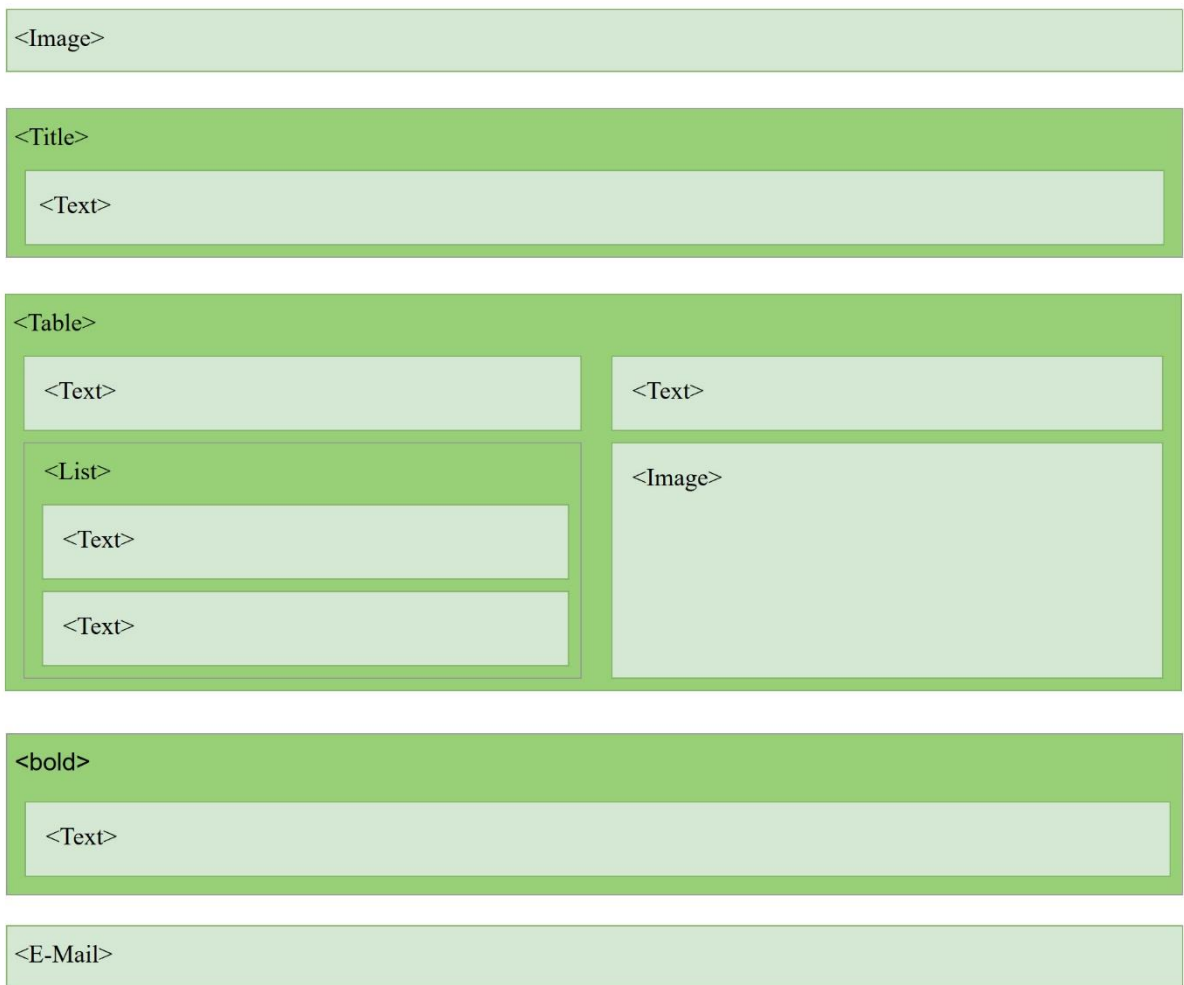
ID   Title	<b>1.3.11.1   Nesting of YODA-Elements</b>
Reference	-
Description	Some <u>YODA-Elements</u> such as table or list shall support multiple layer deep <u>nesting</u> .
Priority   Risk	1   XL

ID   Title	<b>1.3.12.1   Factories</b>
Reference	-
Description	The <u>client</u> should not have to deal with manually creating <u>instances</u> of <u>YODA-Elements</u> , like he has to when creating <u>YODA-Documents</u> and <u>YODA-Projects</u> . Instead, the <u>client</u> should have the ability to use <u>factories</u> that create the <u>YODA-Element</u> and return its <u>instance</u> .
Priority   Risk	2   M

ID   Title	<b>1.3.12.2   Concatenating Factories</b>
Reference	R. 1.3.12.1
Description	The <u>factories</u> should be able to directly be <u>concatenated</u> into each other in order to create different <u>concatenation</u> levels.
Priority   Risk	2   M

# YODA-Project

## YODA-Document



**FIGURE 3: HOW YODA ALLOWS SIMPLE NESTING INSIDE LISTS, TABLES AND DECORATORS.**

The diagram above shall illustrate how nesting in YODA shall work.

## 3.2 Usability

This section includes requirements affecting usability like training time, task times or the language used.

### 3.2.1 Documentation

ID   Title	<b>2.1.1.1   Documentation</b>
Reference	-
Description	The <u>YODA-Documentation</u> should provide an easy start in creating convention-following output files without previous knowledge of the output-file structure. The interface view of the <u>Eiffel classes</u> an act as a basic documentation.
Priority   Risk	3   -

### 3.2.2 Training

ID   Title	<b>2.2.1.1   Client Training</b>
Reference	-
Description	<p>The following <u>client</u>-types must be able to use the system productively for their respective everyday work-life:</p> <ul style="list-style-type: none"> <li>• <u>Client</u>, experienced in <u>Eiffel</u> &amp; <u>HTML</u> or <u>Markup</u> - after 0.5 days of training</li> <li>• <u>Client</u>, experienced in <u>Eiffel</u>, no experience in <u>HTML</u> or <u>Markup</u> - after 1 day of training</li> <li>• <u>Client</u>, no experience in <u>Eiffel</u> &amp; <u>HTML</u> or <u>Markup</u> - after 3 days of training</li> </ul>
Priority   Risk	1   -

### 3.2.3 Task Times

ID   Title	<b>2.3.1.1   Task Times</b>
Reference	-
Description	<ul style="list-style-type: none"> <li>• For a trained <u>client</u>, creating a <u>YODA-Project</u> with one <u>YODA-Document</u> containing no <u>nested YODA-Elements</u> and at most 3 <u>not-nested YODA-Elements</u> with no styling at all, shall take no longer than 30 minutes.</li> <li>• For a trained <u>client</u>, creating a single <u>YODA-Project</u> with not more than 3 not interlinked <u>YODA-Documents</u> each not containing more than 3 <u>YODA-Elements</u> with one <u>nesting layer</u> at most and no styling included, shall take no more than 2 hours.</li> <li>• In general task times can differ in great extent depending on the complexity of the <u>YODA-Project</u> e.g. <u>nesting layers</u>, styling and interlinking of the <u>YODA-Documents</u>.</li> </ul>
Priority   Risk	1   -

### 3.2.4 Language

ID   Title	<b>2.4.1.1   Language</b>
Reference	-
Description	The used language is English.
Priority   Risk	1   -

### 3.3 Reliability

The Eiffel mechanisms such as static typing, assertions, automatic memory management and disciplined exception handling, enabling the YODA-Programmers to state correctness and robustness requirements, and enabling tools to detect inconsistencies before they lead to defects are the key factors that allow reliability.

#### 3.3.1 Availability

ID   Title	<b>3.1.1.1   Availability</b>
Reference	-
Description	<u>YODA</u> is used as a <u>library</u> without needing an internet connection in order to use it although it is recommended. The system shall be available for use at 24 hours a day, every day of the week. Since <u>YODA</u> shall be <u>open source</u> it shall be possible to add useful and correct functionality via <u>GitHub</u> . <u>Clients</u> can pull from this repository in order to add more functions.
Priority   Risk	1   -

#### 3.3.2 Error rate

ID   Title	<b>3.2.1.1   Error rate</b>
Reference	-
Description	The quality of the input by the <u>client</u> is out of scope for this document. The first published version of the <u>library</u> shall have sufficient quality. This is defined by: <ol style="list-style-type: none"> <li>1. Not more than 2 errors per week</li> <li>2. Not more than 2 patch severity errors per two weeks</li> <li>3. Not more than 3 medium and low severity errors are per three weeks</li> </ol>
Priority   Risk	2   -

#### 3.3.3 Error handling

ID   Title	<b>3.3.1.1   Error handling</b>
Reference	3.2.1.1
Description	Error rate shall be as low as possible through using design by contract in <u>Eiffel</u> . In case of errors there shall be detailed error messages. Certain errors shall be handled automatically.
Priority   Risk	2   -

#### 3.3.4 Security

ID   Title	<b>3.4.1.1   Security</b>
Reference	-
Description	<u>YODA</u> will not collect nor store any data of any <u>clients</u> .
Priority   Risk	1   -

### 3.4 Performance Requirements

The golden mean between performance and efficiency like *Dr. Abstract* and *Mr. Microsecond* is hard to meet. But in order to be expandable the focus shall be on the side of architecture and abstraction to easily add other *features* and functionality.

### 3.4.1 Response Time

ID   Title	<b>4.1.1.1   Response Time</b>
Reference	-
Description	<ul style="list-style-type: none"> <li>• Compiling a <u>YODA-Project</u> with one <u>YODA-Document</u> containing no <u>nested YODA-Elements</u> and at most three <u>not-nested YODA-Elements</u> with no styling at all, shall take no longer than one second. The execution time shall not be longer than two seconds.</li> <li>• Compiling a single <u>YODA-Project</u> with not more than three not interlinked <u>YODA-Documents</u> each not containing more than three <u>YODA-Elements</u> with one <u>nesting layer</u> at most and no styling included, shall take no longer than two second. The execution time shall not be longer than three seconds.</li> </ul>
Priority   Risk	3   -

## 3.5 Maintainability

Maintainability is a design consideration concerning the ease with which YODA can be maintained once it is released and running. One aspect of Maintainability describes the facility of bug detection and bug fixing, in case of a malfunction or breakdown of YODA, referred to as corrective Maintenance. Another aspect is the capability of adapting YODA prompt and easily to changes in the environment such as a release of a new Eiffel version, known as adaptive maintenance. Moreover, maintainability involves the handling of perfective maintenance. The main focus lies on a highly extendable software architecture in order to respond to changed stakeholder requirements, both in terms of function and efficiency. A fourth aspect of maintainability deals with the preventative maintenance. This includes infrastructure to anticipate risks as well as the criterion of understandability achievable through consistent coding style and comprehensive documentation.

### 3.5.1 Corrective Maintenance

ID   Title	<b>5.1.1.1   Bug classification</b>
Reference	-
Description	<p>The <u>YODA-Administrators</u> classify bugs</p> <ul style="list-style-type: none"> <li>- that are local, without constraining the <u>YODA-Main-Functionality</u> into S (<i>Small Risk</i>)</li> <li>- that cause side-effects, without constraining the <u>YODA-Main-Functionality</u> into M (<i>Medium Risk</i>)</li> <li>- constraining the <u>YODA-Main-Functionality</u> partially without an imminent risk of a breakdown of <u>YODA</u> into L (<i>Large Risk</i>)</li> <li>- constraining the <u>YODA-Main-Functionality</u> profoundly with an imminent risk of a breakdown of <u>YODA</u>. into XL (<i>Extra Large Risk</i>)</li> </ul>
Priority   Risk	1   XL

ID   Title	<b>5.1.1.2   Bug detection, Test coverage, Test disposability</b>
Reference	R. 5.4.1.1
Description	<p>The <u>YODA-Administrators</u> should</p> <ol style="list-style-type: none"> <li>1. check the <u>YODA-Bug-Board</u> on bugs reported by the <u>YODA-Community</u> once a week.</li> </ol>



	<ol style="list-style-type: none"> <li>undertake the annual scheduled checks</li> </ol> <p>The <u>YODA-Administrators</u> should</p> <ol style="list-style-type: none"> <li>write tests that covers 80% of the code.</li> <li>should put the whole testing environment at the disposal of the <u>community</u> once the <u>library</u> is declared as <u>open source</u>.</li> </ol>
Priority   Risk	2   XL

ID   Title	<b>5.1.1.3   Bug fixing</b>
Reference	R. 5.1.1.1
Description	<p>The <u>YODA-Administrators</u> should fix bugs</p> <ol style="list-style-type: none"> <li>of type XL within 1</li> <li>of type L within 2</li> <li>of type M within 3</li> <li>of type S within 5</li> </ol> <p>week(s) after reported on <u>GitHub</u> by the <u>YODA-Community</u>.</p>
Priority   Risk	1   XL

ID   Title	<b>5.1.1.4   Bug communicating</b>
Reference	-
Description	<p>The <u>YODA-Administrators</u> should set a <u>YODA-Bug-Board</u> up on their <u>GitHub</u> site. The <u>YODA-Bug-Board</u> consists of two lists, one contains supposed bugs reported by the <u>YODA-Community</u>, the other bugs proven by the <u>YODA-Administrators</u>.</p> <ol style="list-style-type: none"> <li>The entries of the first list should consist of a bug index, a bug title, a short bug description referring to the observed limitations, the bug detection date.</li> <li>The entries of the second list should consist of a bug index, a bug title, a short bug description referring to the caused limitations, the bug detection date, the bug classification and a bug fix duration prediction. After the fixing a bug fixing report should be added to the entry, which describes the conducted changes and the affected <u>YODA-Methods</u>.</li> </ol>
Priority   Risk	1   XL

### 3.5.2 Adaptive Maintenance

ID   Title	<b>5.2.1.1   Compatibility with Eiffel</b>
Reference	-
Description	<p>The <u>YODA-Community</u> should adapt the source code of the <u>library</u>, while keeping the same functionality, to new <u>Eiffel</u> versions that are not compatible with the current one within three months after the <u>Eiffel</u> versions release.</p>
Priority   Risk	1   XL

### 3.5.3 Perfective Maintenance

ID   Title	<b>5.3.1.1   Commit incorporation</b>
Reference	-
Description	<p>The <u>YODA-Administrators</u> should decide over the incorporation of committed code into the <u>library</u> within at most 31 days after commitment.</p>

	<p>An incorporation requires</p> <ol style="list-style-type: none"> <li>1. the new code being conform with the reviewed SRS</li> <li>2. writing tests covering the new code</li> </ol> <p>An incorporation goes along with</p> <ol style="list-style-type: none"> <li>1. incorporating the committed code into the source code</li> <li>2. making the correspondent tests accessible to the <u>community</u> by uploading them to <u>GitHub</u>.</li> </ol>
Priority   Risk	3   S

ID   Title	<b>5.3.2.1   YODA-Design-Document</b>
Reference	-
Description	The <u>YODA-Programmer</u> shall note down design decision and used design pattern in a design document as for <u>documentation</u> reason and for better understanding of the program structure
Priority   Risk	1   M

ID   Title	<b>5.3.3.1   Software architecture, Continuity</b>
Reference	-
Description	The <u>YODA-Programmer</u> should apply extendibility to <u>YODAs</u> underlying architecture ensuring that small changes in the SRS induce only small changes in architecture. For this purpose, the <u>Principle of Uniform Access</u> should be taken into consideration.
Priority   Risk	2   M

ID   Title	<b>5.3.3.2   Software architecture, Single Choice Principle</b>
Reference	-
Description	The <u>YODA-Programmer</u> should apply the <u>Single-Choice-Principle</u> in respect of <u>YODA-Projects</u> , <u>YODA-Documents</u> and <u>YODA-Elements</u> .
Priority   Risk	2   M

ID   Title	<b>5.3.3.3   Software architecture, <u>Open-Closed</u> Principle</b>
Reference	-
Description	The <u>YODA-Programmers</u> should structure the <u>library</u> in such a way that it is extensible for other <u>Markup-language</u> output formats, as well as additional functionality within an already existing <u>Markup-language</u> , by adding new <u>classes</u> , attributes and <u>methods</u> to the source code with heavy usage of already existing components, instead of changing or copy-pasting current code.
Priority   Risk	2   M

ID   Title	<b>5.3.3.4   Software architecture, <u>Single Responsibility</u> Principle</b>
Reference	-
Description	The <u>YODA-Programmers</u> should create every module in such a way, that it has responsibility over a single part of the functionality provided by the software, so that it has only one reason to change.
Priority   Risk	2   M

ID   Title	<b>5.3.3.5   Software architecture, Interface-Segregation Principle</b>
Reference	-

Description	The <u>YODA-Programmers</u> should split interfaces that are very large into smaller and more specific ones so that <u>clients</u> will only have to know about the <u>methods</u> that are of interest to them.
Priority   Risk	2   S

### 3.5.4 Preventive Maintenance

ID   Title	<b>5.4.1.1   Scheduled checks</b>
Reference	-
Description	The <u>YODA-Administrators</u> check <u>YODA</u> once a year by <ol style="list-style-type: none"> <li>1. running the entire test set over the source code in order to detect latent faults</li> <li>2. studying the <u>Class Diagram</u> regarding to the architectural requirements defined in the SRS in order to detect out-of-date due to extensions and induce adaption</li> <li>3. analysing the effects of the newest <u>Eiffel</u> version on <u>YODA</u>.</li> </ol>
Priority   Risk	2   S

ID   Title	<b>5.4.2.1   Consistent Coding Style</b>
Reference	D3
Description	All <u>YODA-Programmers</u> in the team should apply the same coding style with respect to <ol style="list-style-type: none"> <li>1. Layout</li> <li>2. Names</li> <li>3. Comments</li> </ol> generating a consistent source code. The coding style should abide the <u>Eiffel</u> conventions described in “ <i>Object-Oriented Software Construction</i> ”, see sections 4.2.1 and 4.2.2.
Priority   Risk	2   S

ID   Title	<b>5.4.2.2   Consistent Layout</b>
Reference	D3
Description	All <u>YODA-Programmers</u> should apply the same layout with respect to <ol style="list-style-type: none"> <li>1. Height and width</li> <li>2. Indenting details</li> <li>3. Spaces</li> <li>4. Precedence and parentheses</li> <li>5. Semicolons</li> <li>6. Assertions</li> </ol> generating a consistent source code.
Priority   Risk	1   S

ID   Title	<b>5.4.3.1   Support</b>
Reference	-
Description	The <u>YODA-Administrators</u> should provide <ol style="list-style-type: none"> <li>1. an <u>open source</u> code with comments conform to the SRS on <u>GitHub</u></li> <li>2. a <u>YODA-Documentation</u> on <u>GitHub</u></li> <li>3. answers within a month to unanswered questions on <u>StackOverflow</u> regarding the use of the <u>library</u></li> </ol>

	4. an e-mail address for direct contact, with a response rate of at least 60% within 20 days for topics regarding bugs, extensions and unanswered problems that were posted on <u>StackOverflow</u> before.
Priority   Risk	2   M

### 3.6 Design Constraints

Design constraints describe decisions made before building the system which have to be addressed during the building phase.

ID   Title	<b>6.0.1.1   Non-interfering</b>
Reference	-
Description	<u>YODA</u> shall not interfere negatively with any other system- or operation.
Priority   Risk	1   XL

ID   Title	<b>6.1.0.0   Output extensibility, Markdown or <u>XML</u></b>
Reference	R. 1.1.1.1
Description	The software architecture shall allow easy extensibility to support more <u>output types</u> in the future, namely <u>XML</u> or <u>Markdown</u> .
Priority   Risk	2   XL

#### 3.6.1 Software language and Coding

ID   Title	<b>6.1.1.1   Eiffel - Coding</b>
Reference	-
Description	All coding shall be done in <u>Eiffel</u> . At every point in time, all code shall be on the same version of <u>Eiffel</u> . The base Version used for this project is <u>Eiffel</u> 17.05, which is also the version the system shall be tested with.
Priority   Risk	1

ID   Title	<b>6.1.1.2   External Libraries</b>
Reference	-
Description	<u>YODA</u> shall run on the <u>client's</u> system without installing or buying any additional <u>libraries</u> except the ones that standardly ship with <u>Eiffel</u> 17.05.
Priority   Risk	1   -

ID   Title	<b>6.1.1.3   Operating Systems used for Development</b>
Reference	-
Description	The development and all testing of <u>YODA</u> will happen on Windows and Mac OS.
Priority   Risk	1   XL

#### 3.6.2 Development tools

ID   Title	<b>6.2.1.1   EiffelStudio</b>
Reference	-
Description	For the development of <u>YODA</u> , the tool to be used shall be <u>EiffelStudio</u> 17.05.
Priority   Risk	1   S

ID   Title	<b>6.2.1.2   Version Control - <u>GitHub</u></b>
Reference	-
Description	<u>YODA</u> 's versions shall be managed by one <u>GitHub</u> repository.
Priority   Risk	1   S

ID   Title	<b>6.2.1.3   <u>GitHub</u> – development branch</b>
Reference	-
Description	The development of <u>YODA</u> shall only happen on the development branch on <u>GitHub</u> (“dev”)
Priority   Risk	1   S

### 3.6.3 Architectural and Design Constraints

ID   Title	<b>6.3.1.1   Open Source</b>
Reference	-
Description	<u>YODA</u> shall be released under an <u>open source</u> free software licence <u>GNU Library General Public License</u> , version 2 (SPDX short identifier: LGPL-2.0)
Priority   Risk	2   S

ID   Title	<b>6.3.2.1   <u>Placeholder-Tags</u> Convention</b>
Reference	-
Description	All <u>Templates</u> shall use the same <u>placeholder-tags</u> . The set of <u>placeholder-tag</u> shall be minimal but complete, which means it shall contain one and only one tag for every different Element supported per <u>Markup-language</u> . All <u>placeholder-tags</u> shall be listed in the <u>YODA-Documentation</u>
Priority   Risk	2   XL

ID   Title	<b>6.3.3.1   Library Constraints</b>
Reference	-
Description	<u>YODA</u> is a library, thus <u>YODA</u> shall NOT provide the functionality to edit and change <u>YODA-Elements</u> after instantiation since the <u>YODA-Element</u> do not exist until <u>rendering</u> . It's the <u>client's</u> task to instantiate the <u>YODA-Element</u> in the form in which it should finally look when <u>rendering</u> .
Priority   Risk	1   XL

## 3.7 External Interfaces

EiffelStudio acts as the main interface to access all the functionalities of YODA such as creating new YODA-Project or adding content to them.

### 3.7.1 User Interfaces

The main client interface shall be EiffelStudio. All the functionalities of YODA can be accessed through EiffelStudio. Additional interfaces include templates on which the generated code can be displayed. Changes and additions to the displayed content shall be made in EiffelStudio. The Console also acts as an interface for YODA since can display the content of YODA-Projects and YODA-Documents as well as the output string.

### 3.7.2 Software Interfaces

Software interfaces include the templates provided by the YODA-Programmers but also the GitHub Repository of YODA. The client can find the saved YODA-Projects and YODA-Documents in the working directory YODA is used in, therefore the working directory also counts as a client interface.

### 3.7.3 Communications Interfaces

YODA does not contain any communication Interfaces. All communication from client to developer shall happen over GitHub and is not integrated into YODA itself. The client shall be provided with information about YODA via GitHub and the YODA-Documentation on there. This information exchange is also not integrated into YODA itself (R 3.6.1.1).

## 4 Supporting Information

### 4.1 Requirement index

#### Project Related Requirements

1.1.1.1   Supported Output Types, first release .....	11
1.1.2.1   YODA-Project, Container of Documents and attributes .....	11
1.1.3.1   Multiple Project Instances .....	11
1.1.4.1   Add YODA-Documents to YODA-Projects .....	11
1.1.4.2   Order of YODA-Documents .....	11
1.1.4.3   Show YODA-Documents in YODA-Project .....	11
1.1.5.1   Render, YODA-Project, generate output .....	11
1.1.6.1   Save YODA-Project to files .....	12
1.2.1.1   YODA-Document, Container of YODA-Elements .....	12
1.1.2.1   Multiple Document Instances .....	12
1.2.3.1   Add YODA-Elements to YODA-Document .....	12
1.2.3.2   Allowed YODA-Elements in YODA-Documents .....	12
1.2.3.3   Order of YODA-Elements .....	12
1.2.4.1   Show YODA-Elements in YODA-Document .....	12
1.2.5.1   Rendering YODA-Documents .....	13
1.2.6.1   Save YODA-Document to files .....	13
1.3.1.1   Text .....	13
1.3.1.2   Text Styling .....	13
1.3.1.3   Text Styling - bold .....	13
1.3.1.4   Text Styling - italic .....	13
1.3.1.5   Text Styling - underline .....	13
1.3.1.6   Text Styling - code .....	13
1.3.1.7   Text Styling - quote .....	14
1.3.1.8   Text Styling - title .....	14
1.3.1.9   Text, handling prohibited strings as input .....	14
1.3.2.1   Code Snippet from File .....	14
1.3.2.2   Code Snippet from String .....	14
1.3.2.3   Code Snippet – not parsed .....	14
1.3.3.1   Anchor element .....	14
1.3.4.1   Link .....	14
1.3.4.2   Link - extern .....	15
1.3.4.3   Link - intern .....	15
1.3.4.4   email .....	15
1.3.4.5   Anchor link .....	15
1.3.5.1   Image, extern resource .....	15
1.3.5.2   Image, local resource .....	15
1.3.5.3   Image, storage in resources folder .....	15
1.3.6.1   List .....	15
1.3.7.1   Table .....	16
1.3.8.1   YODA-Element validation .....	16
1.3.9.1   Not shared YODA-Element representation .....	16

1.3.10.1   Render YODA-Elements .....	16
1.3.11.1   Nesting of YODA-Elements .....	16
1.3.12.1   Factories .....	16
1.3.12.2   Concatenating Factories .....	17
2.1.1.1   Documentation .....	18
2.2.1.1   Client Training .....	18
2.3.1.1   Task Times .....	18
2.4.1.1   Language .....	18
3.1.1.1   Availability .....	19
3.2.1.1   Error rate .....	19
3.3.1.1   Error handling .....	19
3.4.1.1   Security .....	19
4.1.1.1   Response Time .....	20
5.1.1.1   Bug classification .....	20
5.1.1.2   Bug detection, Test coverage, Test disposability .....	20
5.1.1.3   Bug fixing .....	21
5.1.1.4   Bug communicating .....	21
5.2.1.1   Compatibility with Eiffel .....	21
5.3.1.1   Commit incorporation .....	21
5.3.2.1   YODA-Design-Document .....	22
5.3.3.1   Software architecture, Continuity .....	22
5.3.3.2   Software architecture, Single Choice Principle .....	22
5.3.3.3   Software architecture, Open-Closed Principle .....	22
5.3.3.4   Software architecture, Single Responsibility Principle .....	22
5.3.3.5   Software architecture, Interface-Segregation Principle .....	22
5.4.1.1   Scheduled checks .....	23
5.4.2.1   Consistent Coding Style .....	23
5.4.2.2   Consistent Layout .....	23
5.4.3.1   Support .....	23
6.0.1.1   Non-interfering .....	24
6.1.0.0   Output extendibility, Markdown or XML .....	24
6.1.1.1   Eiffel - Coding .....	24
6.1.1.2   External Libraries .....	24
6.1.1.3   Operating Systems used for Development .....	24
6.2.1.1   EiffelStudio .....	24
6.2.1.2   Version Control - GitHub .....	25
6.2.1.3   GitHub – development branch .....	25
6.3.1.1   Open Source .....	25
6.3.2.1   Placeholder-Tags Convention .....	25
6.3.3.1   Library Constraints .....	25

## 4.2 Naming Convention

This naming convention summarizes the Eiffel naming convention described in “Object Oriented Software Construction”. It should serve the YODA-Programmers as an overview, but does not replace the mentioned reference book.



Letter case of Names	<p>The programmers should write</p> <ol style="list-style-type: none"> <li>1. <u>class</u> names and formal generic parameters in all uppercase characters</li> <li>2. feature names, non-constant attributes, routines other than once functions, local entities and routine arguments in all lower-case characters</li> <li>3. constant attributes and once functions with the first letter in lowercase to make <u>class</u> texts consistent and readable.</li> </ol>
Compound words	<p>The programmers should write compound names by separating words by the underscore (" _ ") character to enhance readability.</p>
Name	<p>The programmers should choose names that are</p> <ol style="list-style-type: none"> <li>1. meaningful - to enhance clarity by indicating the intent use of the bearer of the name</li> <li>2. terse - to avoid exaggerated complexity by eliminating unneeded redundancies, including the applying of the composite feature name rule,</li> <li>3. explicit - to enhance clarity by using full words, not abbreviations.</li> </ol>
Grammatical categories	<p>The programmers should</p> <ol style="list-style-type: none"> <li>1. use nouns for <u>class</u> names, may use adjectives for <u>deferred classes</u> describing a structural property</li> <li>2. use verbs in the infinitive or imperative, possibly with complements for procedures</li> <li>3. use nouns for non-boolean query names and adjectives (maybe in <u>is_</u> form) for boolean queries, never use imperative or infinitive verbs for attributes and functions</li> </ol>

### 4.3 Comment Convention

This comment convention summarizes the Eiffel comment convention described in "Object Oriented Software Construction". It should serve the YODA-Programmers as an overview, but does not replace the mentioned reference book.

Header comments	<p>The programmers should write in the source code for every routine a header comment with a one step further indentation than the start of the routine body.</p> <p>The header comment should be</p> <ol style="list-style-type: none"> <li>1. informative by naming what a query returns, qualified noun for a non-boolean query, question form for a boolean query and imperatives or infinitives for a command</li> <li>2. terse by saying what the routine does, not that it does it, applying the Command-Query Separation principle, not paraphrasing type information or precondition's requirements</li> </ol>
-----------------	---

Feature clause header comments	The programmers should write in the source code for every feature a feature clause header comment on the same line as the keyword feature, characterizing the category the feature belongs to.
Non-header comments	<p>The programmers should write non-header comments in the source code only if they provide additional information to the pre- and postconditions, the Invariants and the other forms of comments, needed to prevent confusion and errors.</p> <p>Non-header comments should be of a level of abstraction higher than the code it documents, summarizing its effect instead of paraphrase it.</p>
Software entities	The programmers should write software entities like attributes or arguments in the source code between an opening and a closing quote.