Chapter 1

Software requirement specification

1.1 Introduction

1.1.1 Purpose

The software requirement specification should provide all needed information to develop the context extraction framework and define all delivery objects. All interfaces to external components, input and output data, deployment considerations and quality attribute should be well defined within this document.

1.1.2 Scope

The context extraction framework will perform automated text extraction on a set of HTML test data with two to three different text extraction algorithms. The performance of each algorithm is measured and an output file with the measured results is generated.

1.2 General description

1.2.1 Operating Environment

1.2.1.1 Local environment

JDK	1.7.X
Gradle	1.1
Eclipse Keppler	2.X
git	1.9.X
python	2.7.X

1.2.1.2 Continuous Integration Environment

Open JDK	1.6.X
Open JDK	1.7.X
Oracle JDK	1.7.X
Oracle JDK	1.8.X
Gradle	1.1
Travis CI	

1.2.2 Design and Implementation Constraints

1.2.2.1 User interface

As parts of the text extraction framework may be implemented in a server environment and a user interface is not gewünscht from the client so there will be no graphical user interface. The application is built, deployed and started by gradle. While the application is running, no interaction with the user is needed.

1.3 System Features

This section specifies all system features. Each feature is specified more close with multiple user stories but all important information such as external dependencies and output files are defined in this chapter. The related user stories are located in the planning section.

Name	Read configuration
Feature id	f1
Description	The text extraction framework is configurable with an external text file. The configuration file will contain following items: • Path to folder with html files • Path to folder with text files • Path to folder with output files • Configuration for algorithms • etc. The configuration file location is defined as a relative path to the source directory and structured in a key value list: `key:value;` `key:value;` `key:value;`
Relevance	needed
Related stories	tbd

Name	Create test
Feature id	f2
Description	A test contains two input files which are a html file and a text file. They are located in the defined directories by the configuration. As soon as the test framework finds a html and a text file with the same name, a new test is created and the files are read.
Relevance	needed
Related stories	tbd

Name	Run test
Feature id	f3
Description	A test is run as defined in the configuration file. The configuration file defines which algorithms are tested. The output of a test is a text file which contains the results.
Relevance	needed
Related stories	tbd

Name	Integrate Justext algorithm
Feature id	f4
Description	The justext interface is providing a method with an html file as parameter and an extracted text as return value.
Relevance	needed
Related stories	tbd

Name	Integrate Boilerpipe algorithm
Feature id	f5
Description	The Boilerplate interface is providing a method with an html file as parameter and an extracted text as return value.
Relevance	needed
Related stories	tbd

Name	Integrate RSS feed algorithm
Feature id	f6
Description	The RSS feed algorithm interface is providing a method with an html file as parameter and an extracted text as return value.
Relevance	nice to have
Related stories	tbd

Name	Comparison of extracted text files
Feature id	f7
Description	Each output file from the different algorithm needs to be compared to the text file with the actual content. • Split HTML document into blocks separated by HTML tags • Define which blocks are content and which are boilerplate based on the text file which defines the content • Define which blocks are content and which are boilerplate based on the output file of each text extraction algorithm • Compare the results and categorize all blocks as true negative or false positive • Put the results into an output text file (structure output file: tbd)
Relevance	nice to have
Related stories	tbd

Name	Analyze data
Feature id	f8
Description	From the results of the comparison several further values can be calculated. Some possible values are: • Presicion • Recall/True positive rate (TPR) • false positive rate (FPR) • F-measure • Reciever Operation Characteristics (ROC)
Relevance	needed
Related stories	tbd

Name	Visualize data
Feature id	f9
Description	The calculated values from feature f8 are visualized in diagrams. (tbd: which tool)
Relevance	tbd
Related stories	tbd

1.4 External Interface Requirements

1.4.1 Boilerpipe

The boilerpipe algorithm is already implemented in Java so it is easy to integrate. The API can be found under following link. https://code.google.com/p/boilerpipe/

Other useful links:

Getting started: http://code.google.com/p/boilerpipe/wiki/QuickStart

javadoc extractor: http://boilerpipe.googlecode.com/svn/trunk/boilerpipe-core/javadoc/1.0/de/l3s/boilerpipe/extractors/ExtractorBase.html

1.4.2 justext

The justext algorithem is implemented in python and it is not yet defined how it will be integrated into the text extraction framework. See risk analysis for further information. The documentation can be found under following link:

https://code.google.com/p/justext/

jython: http://www.jython.org/

Chapter 2

Risk analysis

2.1 Introduction

2.1.1 Purpose

This document evaluates and weights all possible risks and defines actions to minimize them as good as possible. blublub

2.2 Risk evaluation

2.2.1 Unclear requirements

Requirements are somehow vague at the beginning of each project and if they are not well defined as soon as possible, they stay vague trough out the whole project and this can lead to a disaster.

2.2.2 New technologies

The new technologies which are present in this project are:

- Gradle
- Travis CI
- Python

Each of them brings his own risk.

2.2.3 Interface Boilerpipe

The Boilerplate algorithm needs to be integrated into the text extraction framework. Every interface of an external component is a possible risk factor.

2.2.4 Interface Justext

The Justext algorithm needs to be integrated into the text extraction framework. Every interface of an external component is a possible risk factor.

2.2.5 Implementation RSS algorithm

The development and implementation of a new algorithm is predestined to generate risks.

2.3 Risk rating

Risk	**Impact**	**Probability of occurrence**	**Risk factor**
Unclear requirements	2	4	8
New technologies	3	3	9
Interface Boilerpipe	5	1	5
Interface Justext	5	5	20
Implementation RSS algorithm	1	5	5

2.4 Consequences

2.4.1 Unclear requirements

As I am working with the client each and every day, it is very easy prevent misunderstandings with asking the client at once. Even though misunderstandings can occur between student and expert. To prevent this, it is necessary to have a document to define the requirements as soon and as exact as possible. This well be done in the form of the system requirement specification in the first mile stone. Possible ambiguities can be clarified at the first mile stone meeting.

2.4.2 New technologies

It is important to do prototyping with new technologies in the first phase of the project to eliminate these risks as soon as possible.

Gralde and Travic CI are needed in the first mile stone to set up the programming environment. So if there is any problem it will occur in a very early stage of the project and a possible solution can be found.

The risks about python are related to the chapter **??**.

2.4.3 Interface Boilerpipe

This risk is rated much lower than the Justext interface becaus it's implementation is in Java and it provides a Java API. Never then less a prototype should be done as soon as possible to prevent any nasty surprises with the interface.

2.4.4 Interface Justext

This point is classified as the highest risk of all. This is because the implementation is in Python and it is not clarified yet how it will be integrated into the text extraction framework. An analysis of possible solution with prototypes needs to be done as soon as possible.

Possible solution are:

- jython (`http://www.jython.org/`)
- Implementation in Java
- Java Processor Interface

2.4.5 Implementation RSS algorithm

This risk has a very high probability of occurrence because it is very likely that a development and an implementation of a new algorithm is going to cause problems. There is no real solution to that risk. But because of this requirement is nice to have, the impact on the outcome of the project is very low. Further more Patrik Lengacher, the tutor of this project, is very experienced in this subject area and will be able to help out if any problems occur.