Software Requirements Specification

**<COMP401 Note: You can remove this page and add your own cover including your team name (logo), product details, and the members of you team>**

**<COMP401 Note: Please leave all section in the document. If you don’t/can’t fill any section, leave it there, empty and shortly explain why you did not filled it>**

*This document outline is based on the IEEE Standard 830-1993 for Software Requirements Specifications.*

*This document was created in part by Steve Mattingly (smattingly@computer.org).*

*This document should specify what functions are to be performed on what data to produce what results at what location for whom.*

*A properly written SRS limits the range of valid designs, but does not specify any particular design.*

*A good SRS is*

* *Correct (accurately captures the “real” requirements)*
* *Unambiguous (all statements have exactly one interpretation)*
* *Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)*
* *Consistent*
* *Ranked for importance and/or stability*
* *Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms specify measurable quantities)*
* *Modifiable (evolve the SRS only via a formal change process, preserving a complete audit trail of changes)*
* *Traceable (cross reference with source documents and spawned documents)*

*The paragraphs written in the “Comment” style are for the benefit of the person writing the document and should be removed before the document is finalized.*

**Version: Draft**

**Project KIWI**

**Team Tango**

**September 11, 1998, Changes: August 2013**

**Revision Chart**

*This chart contains a history of this document’s revisions. The entries below are provided solely for purposes of illustration. Entries should be deleted until the revision they refer to has actually been created.*

*The document itself should be stored in revision control, and a brief description of each version should be entered in the revision control system. That brief description can be repeated in this section. Revisions do not need to be described elsewhere in the document except inasmuch as they explain the development plan itself.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| Draft | Andreas | Initial draft created for distribution and review comments | 18/3/2020 |
| Preliminary | TBD | Second draft incorporating initial review comments, distributed for final review | TBD |
| Final | TBD | First complete draft, which is placed under change control | TBD |
| Revision 1 | TBD | Revised draft, revised according to the change control process and maintained under change control | TBD |
| Revision 2 | TBD | Revised draft, revised according to the change control process and maintained under change control | TBD |
| etc. | TBD | TBD | TBD |

**Contents**

**1.** **Introduction**

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

**2.** **Overall Description**

2.1 Product Perspective

*2.1.1* *System Interfaces*

*2.1.2* *User Interfaces*

*2.1.3* *Hardware Interfaces*

*2.1.4* *Software Interfaces*

*2.1.5* *Communications Interfaces*

*2.1.6* *Memory Constraints*

*2.1.7* *Site Adaptation Requirements*

2.2 User Characteristics

2.3 Constraints

2.4 Assumptions and Dependencies

**3.** **Specific Requirements**

3.1 External Interface Requirements

*3.1.1* *User Interfaces*

*3.1.2* *Hardware Interfaces*

*3.1.3* *Software Interfaces*

*3.1.4* *Communications Interfaces*

3.2 Software Product Features

*3.2.1* *Feature 1*

3.3 Performance Requirements

3.4 Software System Attributes

*3.4.1* *Reliability*

*3.4.2* *Availability*

*3.4.3* *Security*

*3.4.4* *Maintainability*

*3.4.5* *Portability*

3.5 Logical Database Requirements

3.6 Other Requirements

**4.** **Appendices**

# Introduction

When it comes to developer tools, feedback time can make a drastic difference in both development speed, and for the development experience itself. Tools that can make a measurable difference here can save valuable development time. They are also easily marketable because one off cost of the tool can quickly be recouped in the form of improved development efficiency.

## Purpose

The purpose of the Kiwi project is first and foremost to create a tool that facilitates both development efficiency as well as enjoyment in the realm of frontend Javascript unit testing. The purpose of the project foster this, and while doing so, contribute to the ecosystem of the “Kakoune” editor specifically.

## Scope

The project scope is deliberately quite conservative. The first iteration of the testing tool will

only support a highly opinionated setup. As stated in the purpose, the editor plugin will only support the Kakoune code editor. Also, the tool will work specifically with the “Webpack” build tool and with the “chai” assertion framework. The core feature of the project is to run unit tests and print results next to the relevant code snippet in the editor. Additional features such as coverage reports will be added depending on how fast development of the project progresses.

## Definitions, Acronyms, and Abbreviations

Kiwi – An interactive Javascript unit test runner for the Kakoune Editor

Kakoune – A Vim clone text/code editor essentially

Webpack – The most popular Javascript build tool

Chai – An assertion library

## References

https://github.com/stoand/kiwi

github.com/stoand/kiwi-webpack-plugin

# Overall Description

## Product Perspective

### System Interfaces

The product is in the form of a webpack plugin that locates running Kakoune instances and applies highlighters to them according the results of tests that it runs.

### User Interfaces

The end user interacts with the system through two main ways. First, the user configures the plugin in their webpack config. Second, they invoke actions within the Kakoune editor that are defined by the plugin.

### Hardware Interfaces

*-*

### Software Interfaces

Webpack: WP, version 4 or 5 https://github.com/webpack/webpack

documentation: <https://webpack.js.org/concepts/>

Kakoune KA, version 1 https://github.com/mawww/kakoune

documentation: [https://github.com/mawww/kakoune#basic-interaction](https://github.com/mawww/kakoune" \l "basic-interaction)

Chai CH, version 4 https://www.chaijs.com/guide/

documentation: <https://www.chaijs.com/guide/>

### Communications Interfaces

Kakoune command evaluation by executing shell scripts

### Memory Constraints

-

### Site Adaptation Requirements

-

## User Characteristics

The target demographic is developers with at least a moderate understanding of Kakoune and Javascript.

## Constraints

* Needs to be very simple to install and integrate with a project
* Needs to have a fast response time: half a second or less
* Needs to not leave headless browsers running after exiting
* Needs to run deterministically
* Needs to support Linux and Mac platforms

## Assumptions and Dependencies

*List factors that affect the requirements. These factors are not design constraints, but areas where future changes might drive change in the requirements.*

# Specific Requirements

*This section should describe all software requirements at a sufficient level of detail for designers to design a system satisfying the requirements and testers to verity that the system satisfies requirements.*

*Every stated requirement should be externally perceivable by users, operators or other external systems.*

*At a minimum, these requirements should describe every input into the software, every output from the software, and every function performed by the software in response to an input or in support of an output.*

*All requirements should be uniquely identifiable (e.g., by number).*

*The remainder of this sample document is organized according to A.5 Template of SRS Section 3 Organized by Feature shown in the Annex of Std 830-1993. For alternative organizational schemes by system mode, user class, object, stimulus, functional hierarchy, and combinations, see the standard, which is available from IEEE Standards Office, P.O. Box 1331, Piscataway, NJ 08855-1331.*

*No text is necessary between the heading above and the heading below unless otherwise desired.*

## External Interface Requirements

*Provide a detailed description of all inputs into and outputs from the software. This section should complement the interface descriptions under section 2.1 and should not repeat information there. Include both content and format as follows:*

* *name of item*
* *description of purpose*
* *source of input or destination of output*
* *valid range, accuracy, and/or tolerance*
* *units of measure*
* *timing*
* *relationships to other inputs/outputs*
* *screen formats/organization*
* *window formats/organization*
* *data formats*
* *command formats*
* *end messages*

*These requirements may be organized in the following subsections.*

### User Interfaces

### Hardware Interfaces

### Software Interfaces

### Communications Interfaces

## Software Product Features

Please read the “Artifact” spec at [https://kiwi-spec.netlify.com](https://kiwi-spec.netlify.com/)

#### Purpose

#### Stimulus/Response Sequence

#### Associated Functional Requirements

##### Functional Requirement 1

*Repeat subsections at this level and below for each associated functional requirement.*

*Each functional requirement may be described in natural language, pseudocode, or in four subsections as follows. Functional requirements include:*

* *validity checks on inputs*
* *exact sequencing of operations*
* *responses to abnormal situations, including error handling and recovery*
* *effects of parameters*
* *relationships of inputs to outputs, including input/output sequences and formulas for input to output conversion*

###### Introduction

###### Inputs

###### Processing

###### Outputs

## Performance Requirements

*Specify static and dynamic numerical requirements placed on the software or on human interaction with the software.*

*Static numerical requirements may include the number of terminals to be supported, the number of simultaneous users to be supported, and the amount and type of information to be handled.*

*Dynamic numerical requirements may include the number of transactions and tasks and the amount of data to be processed within certain time period for both normal and peak workload conditions.*

*All of these requirements should be stated in measurable form.*

## Software System Attributes

*The following items provide a partial list of system attributes that can serve as requirements that should be objectively verified.*

### Reliability

*Specify the factors needed to establish the software’s required reliability.*

### Availability

*Specify the factors needed to guarantee a defined level of availability.*

### Security

*Specify the factors that will protect the software from accidental or malicious access, misuse, or modification. These factors may include:*

* *cryptography*
* *activity logging*
* *restrictions on intermodule communications*
* *data integrity checks*

### Maintainability

*Specify attributes of the software that relate to ease of maintenance. These requirements may relate to modularity, complexity, or interface design. Requirements should not be placed here simply because they are thought to be good design practices.*

### Portability

*Specify attributes of the software that relate to the ease of porting the software to other host machines and/or operating systems.*

## Logical Database Requirements

*Specify the requirements for any information that is to be placed into a database, including*

* *types of information used by various functions*
* *frequency of use*
* *accessing capabilities*
* *data entities and their relationships*
* *integrity constraints*
* *data retention requirements*

## Other Requirements

# Appendices

*Include supporting detail that would be too distracting to include in the main body of the document.*