
Software Requirements Specification

for

Simple Programming

Version 1.0 approved

Prepared by Graham Mix

11/02/2020

Table of Contents

Table of Contents

1. Introduction	1
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Project Scope	1
1.5 References.....	1
2. Overall Description.....	2
2.1 Product Perspective	2
2.2 Product Features.....	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment.....	2
2.5 Design and Implementation Constraints	2
2.6 User Documentation	3
2.7 Assumptions and Dependencies	3
3. System Features.....	3
3.1 System Feature 1	3
3.2 System Feature 2 (and so on)	4
4. External Interface Requirements.....	4
4.1 User Interfaces.....	4
4.2 Hardware Interfaces	4
4.3 Software Interfaces.....	4
4.4 Communications Interfaces	4
5. Other Nonfunctional Requirements	5
5.1 Performance Requirements.....	5
5.2 Safety Requirements.....	5
5.3 Security Requirements	5
5.4 Software Quality Attributes	5
6. Other Requirements	5

Revision History

Name	Date	Reason For Changes	Version
Graham Mix	11/02/2020	Initial Additions	0.1

1. Introduction

1.1 Purpose

The product specified in this document is a programming language targeted towards teaching kids how to program syntactically.

1.2 Document Conventions

Each requirement statement is to have its own specified priority.

1.3 Intended Audience and Reading Suggestions

This document is meant to be read by other program developers, implementors of this language, or those seeking to create their own programming language. This document should be read in order, top to bottom, starting from Chapter 1. However, the main features of this language are detailed in Chapter 3. Inside that section include what the feature is, its priority level, and its importance and purpose for being in this project.

1.4 Project Scope

The main purpose of this software is to provide a way for students at a middle-school level to learn about programming with syntax, variables, and functions. The goal of this language is to be applied in a classroom setting where a teacher can provide lessons about programming. The language itself will have easy to read documentation that anyone, both teacher and student, can gain important information from. Whether this program will be provided for literal schoolwork is undetermined as of November 2020.

1.5 References

Website: Crafting Interpreters

Nystrom, Robert. *Crafting Interpreters*, 2015, craftinginterpreters.com/.

2. Overall Description

2.1 Product Perspective

This product was an idea for my Senior Project course at Eastern Washington University. After proposing it to my classmates, it was swiftly ignored. Now, I have picked up on this project and hope to complete it within the next few years. This product is new, self-contained, and an original idea.

2.2 Product Features

<Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in Section 3, so only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.>

Major features of this project include:

- Basic program syntax such as variables, functions, and operators
- More advanced program syntax such as loops, if statements, and basic structures
- Allow for user input and printing to the screen
- Mathematical operations including addition, multiplication, and modulo
- User defined types, such as enums

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

3.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

3.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

3.2 System Feature 2 (and so on)

4. External Interface Requirements

4.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

4.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

4.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

4.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>