Software Requirement Specification for Agnolearn

Yiding Li

March 22, 2024

Revision History

Date	**Version**	**Notes**
2024-01-22	0.1	Initial draft
2024-02-02	0.2	Draft for reviewing
2024-03-16	0.3	Transcribe to edit
2024-03-22	1.0	Complete rewrite

Contents

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Not applicable

1.2 Table of Symbols

Not applicable

1.3 Abbreviations and Acronyms

Symbol	Description
EA	Evolutionary algorithm
SRS	Software requirements specification
SC	System constraint
GS	Goal Statement
BFG	Best fitness of generation
A	Assumption
IM	Instance model
FR	Functional requirement
NFR	Nonfunctional requirement
LC	Likely changes
ULC	Unlikely changes

1.4 Mathematical Notation

Symbol	Description
AnyType	Any type
Real	Type of real numbers
Real^n	Type of real vectors
$A \to B$	Type of functions from type A to type B

2 Introduction

black-box optimization

EA

Black-box optimization problems deal with objective functions that are unknown, unexploitable, or non-existent (Eiben and Smith 2003). **Evolutionary algorithms** (**EA**) lend well to optimizing against black-box objective functions (Eiben and Smith 2003).

The development of evolutionary algorithms require tooling support. Agnolearn seeks to provide such support.

2.1 Purpose of Document

SRS

The **software requirement specification** (**SRS**) document records requirement specifications of the system. The specification captures the problem, then derive requirements that instruct solutions to the problem. As such, requirement specifications are completed at the beginning of development, then continues to serve as a roadmap during the development process.

"fake" rational process

While the SRS exists in the beginning of the waterfall model (Ghezzi, Jazayeri and Mandrioli 1991), it may incorporate knowledge gained during later stages. Changes in the SRS then propagates to subsequent stages. This departure from linearity follows the "fake" rational process proposed by Parnas and Clemens (Parnas and Clements 1986).

2.2 Scope of Requirements

Not applicable

2.3 Characteristics of Intended Readers

Understanding this document requires proficiency in at least high school level mathematics. In addition, readers should be familiar with the following concepts:

- **Optimization**: Objective functions, hyper-parameters, and iterative processes
- **Evolutionary learning** : (a) evaluator, (b) variator, and (c) selectors

2.4 Organization of Document

The rest of this document is organised as follows:

- 3 General System Description provides information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics, and lists system constraints.

- 4 Specific System Description begins with the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.
- 5 Requirements provides functional requirements – business tasks that the software is expected to complete, and nonfunctional requirements – qualities that the software SHOULD exhibit.
- 6 Likely Changes and 7 Unlikely Changes document anticipated and unanticipated changes. This information instructs the design of modules, which SHOULD accommodate such changes.
- 8 Traceability Matrices and Graphs: This section connects requirement items in this document. Requirement items include (a) assumptions, (b) goal statements, (c) instance models, (d) functional requirements, (e) nonfunctional requirements, (f) likely changes, and (g) unlikely changes.

3 General System Description

The general system description gives an overview of the system. It (a) identifies the interface between the system and its environment, (b) describes characteristics of users, and (c) lists system constraints that are decided prior to requirements.

This section includes the following subsections:

- 3.1 System Context: Abstract interface and responsibilities of the software
- 3.2 User Characteristics: Characteristics of expected users of the software

3.1 System Context

Agnolearn implements a research framework that facilitates research. In particular, the framework SHOULD facilitate the implementation and analysis of evolutionary algorithms.

The following diagram illustrates two typical use cases. In the first use case, the user interacts with the system to implement an algorithm. In the second case, the user uses the system to analyse an algorithm.

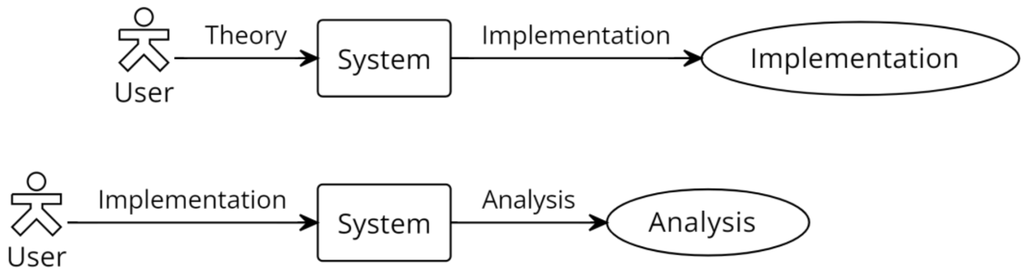

Figure 1: A typical use case

3.2 User Characteristics

The system is designed for two types of users: researchers and learners. The system assumes certain capabilities of each user type.

Researchers extend the system to implement evolutionary operators. A researcher understands the following subjects:

Subject	**Reference**
Types of values and functions	/
Calculus	(Thomas, et al. 2005)
Linear algebra	(Anton 2010)
Optimization and evolutionary operators	(Eiben and Smith 2003)
Linear genetic programming	(Brameier and Banzhaf 2010)
Emergent tangled program graph	(Kelly and Heywood 2018)
Inheritance, encapsulation, and polymorphism	(Martin 2008)

Learners study the system for an abstract view of evolutionary learning. A learner understands the following subjects:

Subject	**Reference**
Calculus	(Thomas, et al. 2005)
Linear algebra	(Anton 2010)

3.3 System Constraints

system constraint

System constraints are restrictions imposed by practical limitations, best practices, and stakeholder requirements. Legal, regulatory, ethical, and moral considerations fall into this category.

SC 1: The system MUST follow an object-oriented design, which includes patterns such as encapsulation, inheritance, and polymorphism.

- Rationale: Feasibility study shows that most frameworks that implement evolutionary learning follow object-oriented designs. This decision .

4 Specific System Description

The specific system description describes the problem, then offers an abstract description of how the system can solve the problem.

This section includes the following subsections:

- 4.1 Problem Description describes the problem, then refines the problem into goals.
- 4.2 Solution Characteristics Specification describes a model of the problem.

4.1 Problem Description

Empirical evidence supports the analysis of evolutionary algorithms. As such, the system MUST support the implementation and empirical analysis of evolutionary algorithms.

As a research software, the system SHOULD be independent to third-party modules, so as to keep the system transparent and avoid dependency problems.

To facilitate development and encourage best practices, the system MUST provide an intuitive interface and thorough instructions.

4.1.1 Terminology and Definitions

The following sections use non-standard terminologies to describe exact requirements. These terminologies are as follows:

- **Operator suite**: a set of operator implementations that include: (a) one genome, (b) one evaluator, (b) one variator, and (c) one selector.
- **Genome suite**: A set of operator implementations that include: (a) one genome, (b) one evaluator, and (c) one variator.
- **Best fitness of generation** (**BFG**): The fitness of the solution with highest fitness among the population of a generation
- **Optimizing algorithm**: An evolutionary algorithm where the BFG of each generation increases from that of the previous generation.
- **Operator interchangeability**: Operator A isinterchangeable with operator B if, for any optimizing algorithm that incorporates B, exchanging B for A results also in an optimizing algorithm.
- **External module**: A software module that is not offered as part of the language implementation

Not applicable. User characteristics assumes understanding of many terminologies, and gives a book for reference.

4.1.2 Physical System Description

Not applicable

4.1.3 Goal Statements

goal

Goals decompose the problem into independent, achievable objectives. The system MUST achieve the following goals.

GS 1 (Implement): Enable the implementation of interoperating evolutionary operators.

GS 2 (Test): Provide utilities that identify significant faults in existing operators.

GS 3 (Analysis): Analyse and compare evolutionary operators.

GS 4 (Ease of Use): Provide comprehensive documentation and instructions.

GS 5 (Independence): Minimize dependency to external modules.

GS 6 (Speed): Implement concurrency and parallelism to speed up algorithms implemented using the system.

4.2 Solution Characteristics Specification

No system can be developed with a total understanding of the problem. Attempting to do so is also unwise, because a *maximally comprehensive problem* demands equally complex solutions.

problem model

The development process must operate with a fixed **problem model** that captures the problem. The following subsections describe this model.

This section includes the following subsections:

- 4.2.2 Scope Decisions establish boundaries for the project, in terms of what will and will not be done.
- 4.2.4 Assumptions simplify development.

- 4.2.9 Instance Models describes the theoretical foundation of the system.

4.2.1 Types

Not applicable

4.2.2 Scope Decisions

scope

The **scope** of the project decides what will and will not be done. Scope decisions are made when development begins and remains unchanged throughout development, unless through a formal process.

Scope decisions of the project are as follows:

- Locality: The system deploys on a local machine and interfaces with local users. The system does not directly interact with network traffic.
- Privacy and Security: The system does not implement security measures (such as authentication) or privacy measures (such as differential privacy).
- Expert user: The user meets characteristics specified in 3.2 User Characteristics.

4.2.3 Modelling Decisions

Not applicable

4.2.4 Assumptions

assumption

Assumptions simplify the problem model. Assumptions are assumed to be true during development, with the goal of reducing complexity and streamlining the development process.

Not applicable

4.2.5 Theoretical Models

Not used

4.2.6 General Definitions

Not used

4.2.7 Data Definition

Not used

4.2.8 Data Types

Not used

4.2.9 Instance Models

The project operates with the instance model of optimization.

Number	IM 1: Optimization
Label	Optimize input
Input	$x : \text{AnyType}$ $\text{eval} : \text{AnyType} \to \text{Real}$
Output	$x : \text{AnyType}$
Description	The system receives an input x and an evaluator eval that accepts x as input. The system then emits y for which $$\text{eval}(y) \geq \text{eval}(x).$$
Goal	GS1

GS 1 › IM 1

4.2.10 Input Data Constrains

Not used

4.2.11 Properties of a Correct Solution

Not used

5 Requirements

Requirements refine goals and derive from goals. In particular, requirements MUST be (a) actionable and (b) verifiable:

- Actionable requirements instruct design and implementation. All design decisions must contribute to the **fulfilment** of requirements.
- Requirements must be verifiable to instruct verification and validation. Checking against a requirement indirectly examines if the related goal is achieved and, subsequently, if the problem is solved.

This section includes the following subsections:

- 5.1 Functional Requirements: Requirements that make explicit tasks and behaviours the software should complete
- 5.2 Nonfunctional Requirements: Requirements that facilitate the satisfaction of functional requirements
- 5.3 Rationale: Rationale for scope decisions, modelling decision, and assumptions

5.1 Functional Requirements

Functional requirements describe tasks that the system MUST complete.

GS 1 ›

FR 1: (extensibility): Provides ways to implement operators

FR 2: (variator interchangeability): Correctly implemented variators of the same genome type are interchangeable.

FR 3 (selector interchangeability): Correctly implement selectors are interchangeable

GS 2 ›

FR 4 (test evaluator): Empirically, test if an evolutionary operator exhibits expected behaviours.

FR 5 (test algorithm): Empirically, test if an operator suite exhibits expected behaviours.

GS 3 ›

FR 6 (analyse evaluator): Empirically, analyse then display the performance of an evolutionary operator.

FR 7 (analyse algorithm): Empirically, analyse then display the performance of an operator suite.

GS 4 ›

FR 8 (reference implementation – OneMax): Provide an operator suite that solves the OneMax problem

FR 9 (reference implementation – floating point): Provide a genome suite for floating-point representations

FR 10 (reference implementation – genetic programming): Provide a genome suite for genetic programming

FR 11 (reference implementation – linear programming): Provide a genome suite for linear programming

FR 12 (reference implementation – tangled program graph): Provide a genome suite for tangled program graphs

5.2 Nonfunctional Requirements

Nonfunctional requirements describe qualities the system MUST exhibit.

GS 4 ›

NFR 1 (intuitive interface): The interface is used and deemed intuitive by a domain expert

NFR 2 (thorough documentation): Every public object and public function is documented.

NFR 3 (comprehensive instructions): The system offers step-by-step instructions that lead to the success implementation of an operator suite

GS 5 ›

NFR 4 (self sufficiency): Implementation of operator suites using the system does not, directly or indirectly, require external modules.

NFR 5 (platform independence): Implementation of operator suites using the system does not directly depend on the underlying operating system

GS 6 ›

NFR 6 (parallelization): The system can run multiple instances of the same operator at the same time.

5.3 Rationale

Requirements for research software are difficult to interrogate, because these requirements must anticipate novel algorithms that are yet to exist. Any explicit requirement, therefore, risk the system's ability to implement novel algorithms. Many requirements, in particular those relating to GS 1, follow feedback from an domain expert.

A software can assist research in many ways. Requirements for the system only cover the

The assumption of reader and user characteristics greatly simplifies the development process.

6 Likely Changes

Not applicable

7 Unlikely Changes

Not applicable

8 Traceability Matrices and Graphs

Requirements derive from goals. In particular, the system achieves a goal if it satisfies all related requirements. Traceability matrices and graphs explicitly connects requirements to goals.

VnV

MG

This information also instructs the propagation of changes. For example, if a goal changes, then only requirements following that goal needs to change. Subsequent development artefacts, such as the **verification and validation (VnV) plan** and **module guide (MG)**, MUST similarly depend on requirements.

Table 1: Derivation from goals to functional requirements

G	FR											
	1	2	3	4	5	6	7	8	9	10	11	12
1	✓	✓	✓									
2				✓	✓							
3						✓	✓					
4								✓	✓	✓	✓	✓

Table 2: Derivation from goals to nonfunctional requirements

G	NFR					
	1	2	3	4	5	6
4	✓	✓	✓			
5				✓	✓	
6						✓

The traceability graph visualizes the traceability matrix. Arrows in the graph represent dependency: the component at the tail of an arrow depends on the component at the head of that arrow. Therefore, if a component is changed, then all components that depend on that component MUST also change.

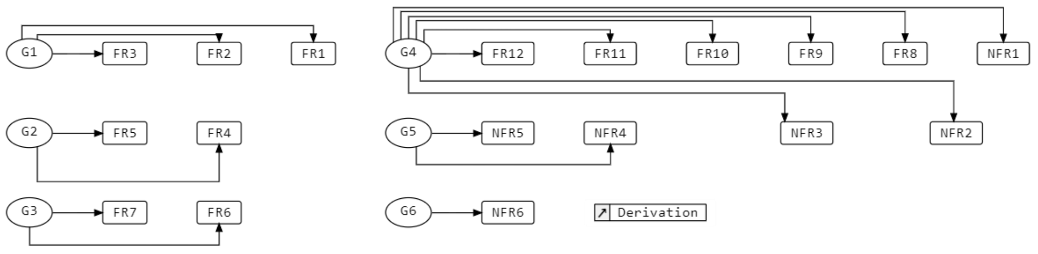

Figure 2: Traceability graph from goals to requirements

9 Development Plan

Not applicable

10 Style References

This document is developed from an existing template (Thomas, et al. 2005) (Smith, Lai and Khedri, Requirements Analysis for Engineering Computation: A Systematic Approach for Improving Software Reliability 2007) (Smith and Koothoor, A Document-Driven Method for Certifying Scientific Computing Software for Use in Nuclear Safety Analysis 2016).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (Bradner 1997).

11 Works Cited

Anton, H. 2010. *Elementary Linear Algebra.* 10. John Wiley & Sons.

Bradner, S. 1997. *RFC2119: Key words for use in RFCs to Indicate Requirement Levels.* Edited by RFC Editor. doi:10.17487/RFC2119.

Brameier, Markus F., and Wolfgang Banzhaf. 2010. *Linear Genetic Programming.* 1. Springer Publishing Company, Incorporated. doi:10.5555/1951880.

Eiben, A, and Jim Smith. 2003. *Introduction To Evolutionary Computing.* Vol. 45. Springer. doi:10.1007/978-3-662-05094-1.

Ghezzi, Carlo, Mehdi Jazayeri, and Dino Mandrioli. 1991. *Fundamentals of software engineering.* 2. Prentice-Hall, Inc. doi:10.5555/102710.

Kelly, Stephen, and Malcolm I. Heywood. 2018. "Emergent Tangled Program Graphs in Multi-Task Learning." *International Joint Conference on Artificial Intelligence.*

Martin, Robert C. 2008. *Clean Code: A Handbook of Agile Software Craftsmanship.* 1. Prentice Hall PTR. doi:10.5555/1388398.

Parnas, David L., and P.C. Clements. 1986. "A Rational Design Process: How and Why to Fake it." *IEEE Transactions on Software Engineering* 12: 251-257.

Ralyte, Jolita, Pär J. Ågerfalk, and Naoufel Kraiem. 2005. "A New Requirements Template for Scientific Computing." *Proceedings of the First International Workshop on Situational Requirements Engineering Processes -- Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05* (In conjunction with 13th IEEE International Requirements Engineering Conference) 107--121.

Smith, W. Spencer, and Nirmitha Koothoor. 2016. “A Document-Driven Method for Certifying Scientific Computing Software for Use in Nuclear Safety Analysis.” *Nuclear Engineering and Technology* 48 (2): 404--418.

Smith, W. Spencer, Lei Lai, and Ridha Khedri. 2007. “Requirements Analysis for Engineering Computation: A Systematic Approach for Improving Software Reliability.” *Reliable Computing, Special Issue on Reliable Engineering Computation* 13: 83--107.

Thomas, George B., Maurice D. Weir, Joel R. Hass, and Frank R. Giordano. 2005. *Thomas' Calculus Early Transcendentals.* 11. Addison-Wesley Longman Publishing Co., Inc. doi:10.5555/1202398.