Module Guide for AgnoLearn

Yiding Li
March 15, 2024

1 Revision History

Date	Version	Notes
2024-03-15	1.0	Initial draft
2024-03-22	1.1	Complete rework
2024-04-05	1.2	Apply fixes per suggestions

Contents

2 Reference Material

This section records information for easy reference.

Table 1: Abbreviations and Acronyms

Symbol	Description
MG	Module guide
SRS	Software requirement specification
AC	Anticipated change
UC	Unlikely change
HHM	Hardware-hiding modules
BHM	Behaviour-hiding modules
SHM	Software-hiding modules
ADT	Abstract data type

3 Introduction

module

A **module** is a work assignment for a programmer or programming team (Parnas, Clement and Weiss 1984). Decomposing a system into modules promotes abstraction information hiding (D. L. Parnas 1972) and supports design for change.

secret

A module presents an abstract interface that hides secrets. **Secrets** are system details that are likely to change; hiding secrets, therefore, allows such changes to take place without affecting the interface (Parnas, Clement and Weiss 1984).

design for change

Design for change is valuable in scientific computing, where modifications are frequent, especially during initial development as the solution space is explored.

The design of Agnolearn follows the following rules laid out by Parnas et al. (Parnas, Clement and Weiss 1984):

- System details that change independency SHOULD be the secrets of separate modules.
- Each data structure SHOULD be implemented in only one module
- Program MUST only be able to obtain information stored in other modules by calling access programs belonging to that module.

3.1 Purpose of Document

MG
SRS

The **module guide** (**MG**) follows the **software requirement specification** (**SRS**) (Parnas, Clement and Weiss 1984). The MG specifies the modular structure of the system, which allows designers and maintainers to easily identify parts of the software.

3.2 Characteristics of Intended Readers

Potential readers of this document are as follows:

- **New project members**: This document guides new project members to easily understand the overall structure and identify modules that relate to their responsibilities.
- **Maintainers**: This document describes the hierarchical structure of modules. This information is useful to maintains who make changes to the system. Maintains MUST update relevant sections of this document after changes have been made.

- **Designers**: This document provides an abstract view of modules. Using this information, designers can verify the system in various ways, such as (a) consistency among modules, (b) feasibility of the decomposition, and (c) flexibility of the design.

3.3 Document Structure

The rest of this document is organised as follows:

- Anticipated and Unlikely Changes lists the anticipate and unlikely changes of the software requirements.
- Module Hierarchy summarizes the module decomposition that was constructed according to the unlikely changes.
- Traceability specifies the connections between the software requirements and the modules.
- Module Decompositions gives a detailed description of the modules
- Module detail 19: FP

Type	ADT
Secrets	–
Services	Implementation of a floating-point EA
Implemented By	The system

Module detail 20: GP

Type	**ADT**
Secrets	–
Services	Implementation of a genetic programming EA
Implemented By	The system

Module detail 21: LP

Type	**ADT**
Secrets	–
Services	Implementation of a linear programming EA
Implemented By	The system

Module detail 22: TPG

Type	**ADT**
Secrets	–
Services	Implementation of tangled program graph
Implemented By	The system

- Traceability Matrix includes two traceability matrices: one checks the completeness of the design against the requirements provided in the SRS; the other shows the relation between anticipated changes and the modules.
- Use Hierarchy describes the use relation between modules.

4 Anticipated and Unlikely Changes

AC
UC

Changes classify into two categories: (a) **anticipated changes** (**ACs**) are listed in section 4.1; (b) **unlikely changes (UC)** are listed in section 4.2.

4.1.1 Anticipated Changes

Anticipated changes are the source of secrets, which MUST hide inside modules. Changing an anticipated change SHOULD only require changing one module which hides that decision.

AC 1: Implementations of evolutionary operators

4.1.2 Unlikely Changes

Unlikely changes are fixed design decisions. These decisions lead to public-facing interfaces. Changing an unlikely change leads to modifications in many parts of the design.

UC 1: Independence between operators

5 Module Hierarchy

HHM

Modules classify into a hierarchy of (a) hardware-hiding modules, (b) behaviour-hiding modules, and (c) software-hiding modules.

BHM

SHM

(A) **Hardware-hiding modules** (**HHM**) implement virtual hardware that is used by the rest of the software.

(B) **Behaviour-hiding modules** (**BHM**) implement behaviours that are specified in the requirement document.

(C) **Software-hiding modules** (**SHM**) capture design decisions that are based upon (a) mathematical theorems, (b) physical facts, and (c) programming considerations such as algorithmic efficiency and accuracy.

The following Table 2 summarizes modules by the aforementioned hierarchy.

TABLE 2: MODULE HIERARCHY

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	`Controller, Genome, Population, ParentPool, Evaluator, Selector, Variator`
Software-Decision Module	`TestEvaluator, TestSelector, TestVariator, TestController, TestAlgorithm, AnalyseEvaluator, AnalyseSelector, AnalyseVariator. AnalyseAlgorithm, OneMax, FP, GP, LP, TPG`

6 Traceability

The design of Agnolearn MUST satisfy requirements developed in the SRS. In particular, every module must derive from at least one requirement. The following table and graph connects modules to requirements.

Table 3: Trace Between Requirements and Modules

Requirement	Modules
FR 1 FR 2 FR 3	`Controller, Genome, Population, ParentPool, Evaluator, Selector, Variator`
FR 4	`TestEvaluator, TestSelector, TestVariator, TestController`
FR 5	`TestAlgorithm`
FR 6	`AnalyseEvaluator, AnalyseSelector, AnalyseVariator`
FR 7	`AnalyseAlgorithm`
FR 8, FR 13	`OneMax`
FR 9, FR 13	`FP`
FR 10, FR 13	`GP`
FR 11, FR 13	`LP`
FR 12, FR 13	`TPG`

7 Module Decompositions

module decomposition

The principle of information hiding (Parnas, Clement and Weiss 1984) instructs **module decomposition**. Each item in following sections consists of these fields:

- *Secrets* briefly states the design decisions hidden by the module.
- *Services* specifies what the module will do, without documenting how to do it.

- *Implemented By* suggests the implementing software.

Modules classify into the following types: (a) **libraries** that only implement behaviours, (a) **records** that only implement data and (c) **abstract data types** (**ADTs**) that implement states and behaviours that change data .

Table 4: Comparison of library, record, and ADT

Module Type	**Behaviour**	**Data**
Library	Implemented	None
Record	None	Implemented
ADT	Implemented	Implemented

7.1 Hardware Hiding Modules

7.1.1 Virtual Hardware

The virtual hardware hiding module hides interactions with the hardware.

Module detail 1: Virtual hardware

Secrets	The data structure and algorithm used to implement the virtual hardware.
Services	Serves as a virtual hardware used by the rest of the system.
Implemented By	OS

The programming language Python, in particular the CPython 3.12 implementation of Python (Python Software Foundation 2024), hides interactions with the virtual hardware.

Module detail 2: Python

Secrets	Dependency to the virtual hardware
Services	Programming language
Implemented By	The Python Software Foundation

7.2 Behaviour-Hiding Module

The Genome module describes a genome. In particular, the module represents the capabilities of genomes to store a score and be deep copied.

Module detail 3: Genome

Type	Generic ADT
Secrets	Genotype
Services	Represent a single genome and its capabilities to store a score and be deep copied
Implemented By	The system

The Population represents a collection of genomes. This module hides the data structure that implements the collection.

Module detail 4: Population

Type	Generic ADT
Secrets	Data structure that collects genomes
Services	Represent a collection of genomes
Implemented By	The system

The ParentPool represents a collection of tuples of solutions. It captures the state of the population after parent selection and before variation, where tuples of parents are prepared but not used. This module hides the data structure that implements the collection.

Module detail 5: ParentPool

Type	Generic ADT
Secrets	Data structure of tuples of Genomes
Services	Represent a collection of tuples of genomes
Implemented By	The system

The modules Evaluator, Variator, and Selector define common interfaces of operator implementations.

Module detail 6: Evaluator

Type	Interface
Secrets	–
Services	Abstract interface for evaluators
Implemented By	The system

Module detail 7: Variator

Type	Interface
Secrets	–
Services	Abstract interface for variators
Implemented By	The system

Module detail 8: Selector

Type	Interface
Secrets	–
Services	Abstract interface for selectors
Implemented By	The system

The module Controller defines a common interface for controllers, environments where operators interact with each other.

Module detail 9: Controller

Type	Interface
Secrets	–
Services	Abstract interface for controllers
Implemented By	The system

7.3 Software Decision Module

The modules TestEvaluator, TestSelector, and TestVariator detect significant defects in operator implementations. The module TestAlgorithm detects significant defects in an operator suite.

These defects include, for example, mismatching arities.

Module detail 10: TestEvaluator

Type	ADT
Secrets	Process of testing
Services	Detect defects in evaluator implementations
Implemented By	The system

Module detail 11: TestSelector

Type	ADT
Secrets	Process of testing
Services	Detect defects in selector implementations
Implemented By	The system

Module detail 12: TestVariator

Type	ADT
Secrets	Process of testing
Services	Detect defects in variator implementations
Implemented By	The system

Module detail 13: TestController

Type	ADT
Secrets	Process of testing
Services	Detect defects in controller implementations
Implemented By	The system

The modules AnalyseEvaluator, AnalyseSelector, and AnalyseVariator collect and display the empirical performance of operator implementations. The module AnalyseAlgorithm does so for an operator suite.

Module detail 14: AnalyseEvaluator

Type	ADT
Secrets	Exact integration of
Services	Execution of evolutionary algorithm
Implemented By	The system

Module detail 15: AnalyseSelector

Type	ADT
Secrets	Exact integration of
Services	Execution of evolutionary algorithm
Implemented By	The system

Module detail 16: AnalyseVariator

Type	ADT
Secrets	Exact integration of
Services	Execution of evolutionary algorithm
Implemented By	The system

Module detail 17: AnalyseController

Type	ADT
Secrets	Exact integration of
Services	Execution of evolutionary algorithm
Implemented By	The system

The modules OneMax, FP (for floating-point representations), GP (for genetic programs), LP (for linear programs), and TPG (for tangled program graphs) provide reference implementations for users who wish to implement custom algorithms.

Module detail 18: OneMax

Type	ADT
Secrets	–
Services	Implementation of OneMax problems
Implemented By	The system

Module detail 19: FP

Type	ADT
Secrets	–
Services	Implementation of a floating-point EA
Implemented By	The system

Module detail 20: GP

Type	**ADT**
Secrets	–
Services	Implementation of a genetic programming EA
Implemented By	The system

Module detail 21: LP

Type	**ADT**
Secrets	–
Services	Implementation of a linear programming EA
Implemented By	The system

Module detail 22: TPG

Type	**ADT**
Secrets	–
Services	Implementation of tangled program graph
Implemented By	The system

8 Traceability Matrix and Graph

Table 3 details the connection between requirements and modules. The following figure visualizes these connections.

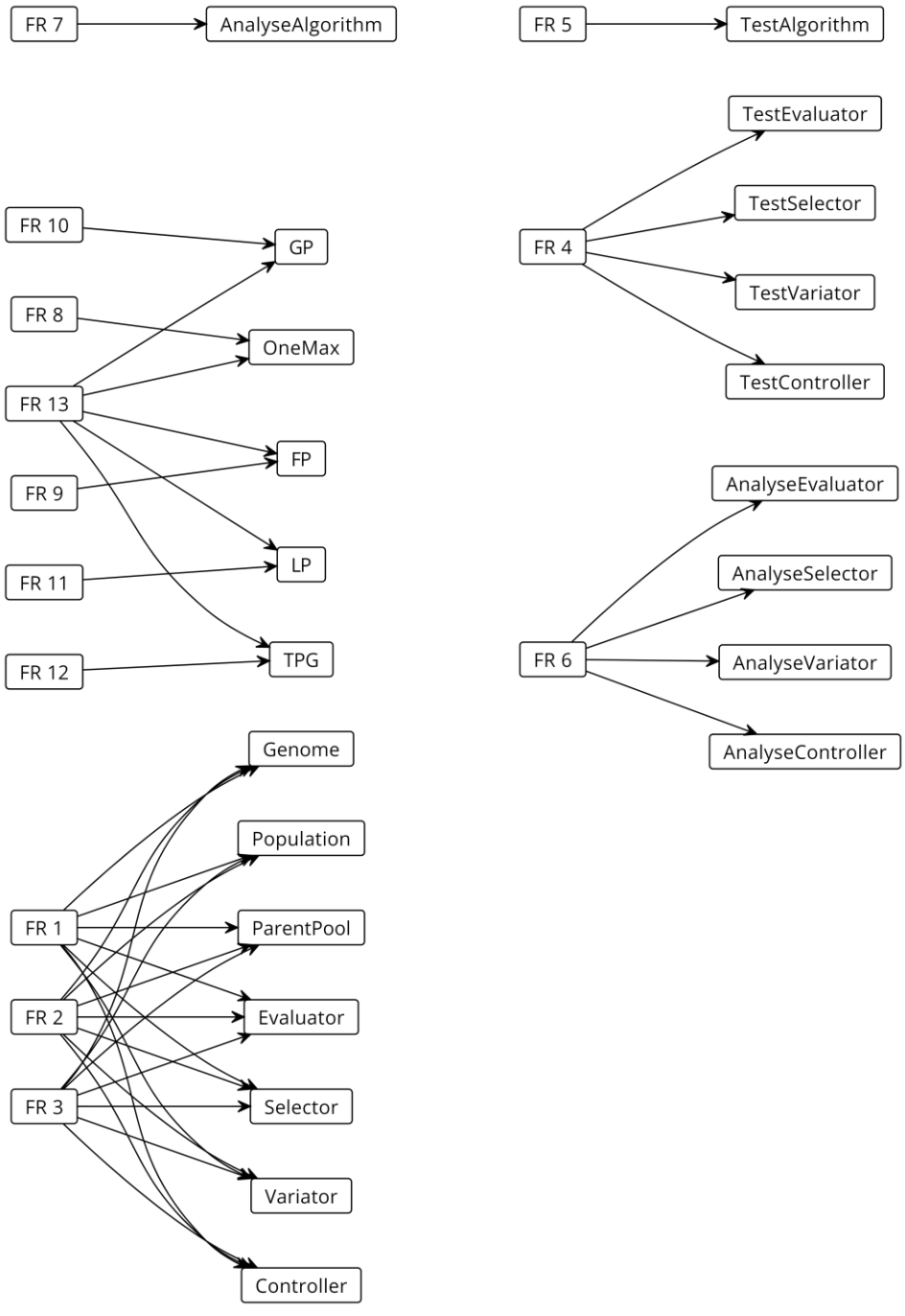

Figure 1: Traceability Graph

9 Use Hierarchy

The use hierarchy diagram describes the use relation between modules.

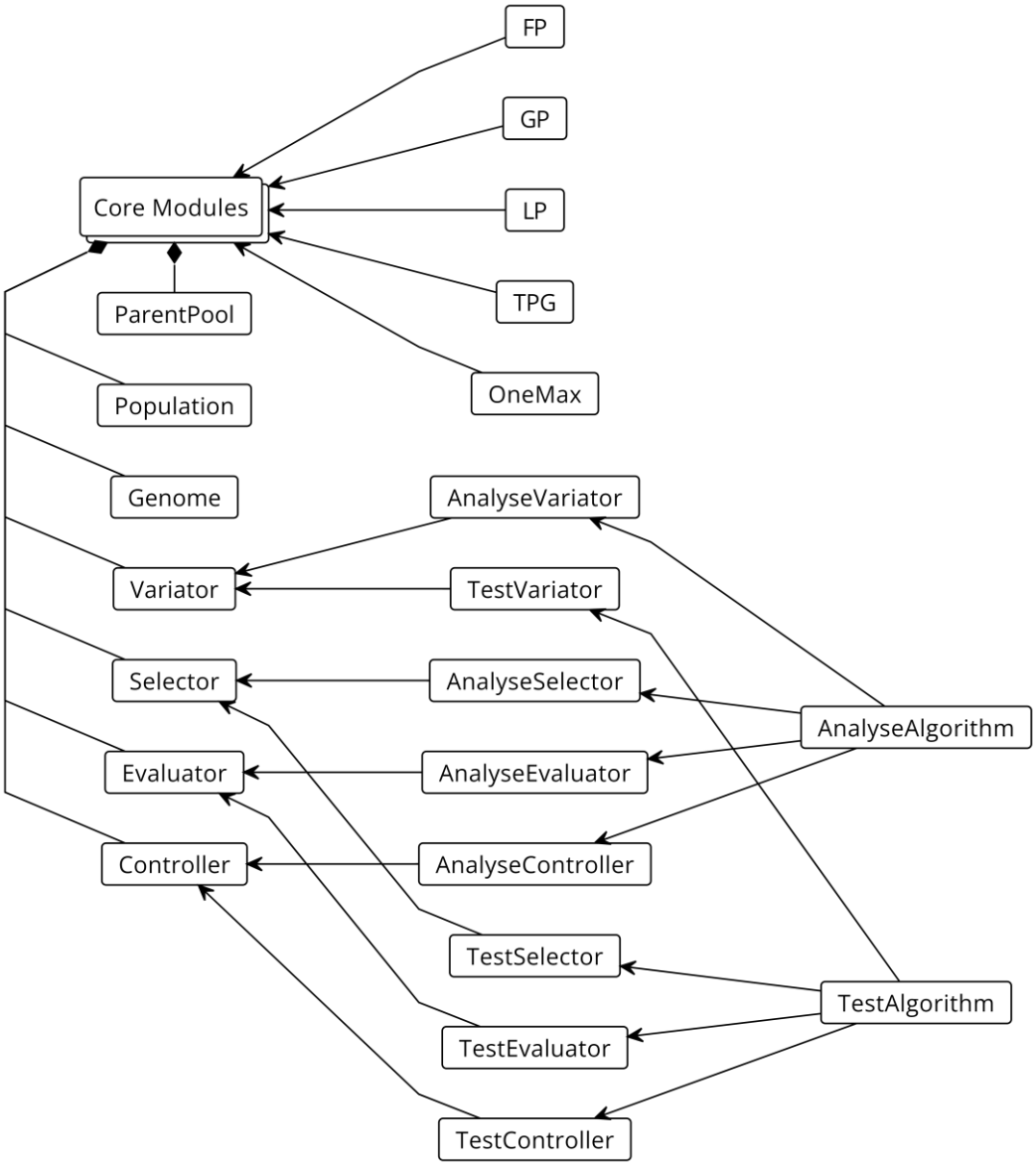

Figure 2: Use hierarchy diagram

10 Formats and References

The design of Agnolearn follows *The Modular Structure of Complex Systems* (Parnas, Clement and Weiss 1984).

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.