securityandtrust.lu

FAQAS Framework
RB - Requirements Baseline
SSS - Software System Specification

O. Cornejo, F. Pastore

Interdisciplinary Centre for Security, Reliability and Trust

University of Luxembourg

ITT-1-9873-ESA-FAQAS-SSS

Issue 4, Rev. 1

November 23, 2021

UNIVERSITÉ DU LUXEMBOURG

Revisions

Issue Number	Date	Authors	Description
ITT-1-9873-ESA-FAQAS-SSS Issue 1 Rev. 1	November 6th, 2020	Fabrizio Pastore, Oscar Cornejo	Initial release.
ITT-1-9873-ESA-FAQAS-SSS Issue 2 Rev. 1	March 31th, 2021	Oscar Cornejo, Fabrizio Pastore	Second release.
ITT-1-9873-ESA-FAQAS-SSS Issue 2 Rev. 2	April 14th, 2021	Fabrizio Pastore, Oscar Cornejo	ESA comments addressed.
ITT-1-9873-ESA-FAQAS-SSS Issue 2 Rev. 3	April 14th, 2021	Fabrizio Pastore, Oscar Cornejo	Added requirements discussed during review meeting.
ITT-1-9873-ESA-FAQAS-SSS Issue 3 Rev. 1	September 20th, 2021	Fabrizio Pastore, Oscar Cornejo, Enrico Viganò	Added Section 4.1.4, .
ITT-1-9873-ESA-FAQAS-SSS Issue 3 Rev. 2	October 7th, 2021	Fabrizio Pastore, Oscar Cornejo, Enrico Viganò	Revised document.
ITT-1-9873-ESA-FAQAS-SSS Issue 4 Rev. 1	October 29th, 2021	Fabrizio Pastore, Oscar Cornejo, Enrico Viganò	Formatting.

Contents

Chapter 1

Introduction

This document is the deliverable SSS of the ESA activity ITT-1-9873-ESA. It concerns requirements specification for the *FAQAS framework* to be delivered by ITT-1-9873-ESA. Following the structure described in the SoW *AO9873-ws00pe_SOW.pdf*, it provides the structured requirements baseline for the FAQAS framework according to ECSS-E-ST-40C Annex B.

1.1 Applicable and reference documents

- D1 - Mutation testing survey
- D2 - Study of mutation testing applicability to space software

Chapter 2

Terms, definitions and abbreviated terms

- FAQAS: activity ITT-1-9873-ESA
- FAQAS-framework: software system to be released at the end of WP4 of FAQAS
- D2: Deliverable D2 of FAQAS, *Study of mutation testing applicability to space software*
- HPC: High Performance Computing. Computing infrastructure where end-users can execute tasks in parallel on multiple nodes.
- KLEE: Third party test generation tool, details are provided in D2.
- Mutation analysis: the process of assessing a test suite through code mutation. In D2 it is also called code-driven test suite evaluation and code-driven test suite assessment.
- SUM: Software User Manual, as per definition in ECSS-E-ST-40C.
- SUT: Software under test, i.e, the software that should be mutated by means of mutation testing.
- Test suite augmentation: the process of generating one or more test cases to kill one or more mutants. A test case is said to kill a mutant if it fails when executed with the mutated software. Test suite augmentation can be automated or semi-automated.
- WP: Work package

Chapter 3

Software Overview

3.1 Function and Purpose

The FAQAS activity concerns the investigation of mutation testing as a method to evaluate the quality of software test suites and mutation testing as a method to derive new software test cases.

The FAQAS-framework shall support a code-driven and data-driven mutation testing methodology that integrates and extends state-of-art solutions. The reference methodology has been described in D2. The methodology enables a scalable and accurate mutation testing process even when the software under test is large and characterized by test suites requiring long execution time.

The end-user of the FAQAS-framework is a software engineer (i.e., a professional with a master degree in informatics or related fields) who aims to evaluate the quality of the test suite developed for a flight software component or system. Hereafter, the end-user of FAQAS-framework is referred to as *the engineer*.

Since FAQAS-framework implements two distinct features, code-driven mutation testing and data-driven mutation testing, this document contains two separate sections, each one concerning one of the two features: Section 4.1.1 concerns code-driven mutation testing, Section 4.1.3 concerns data-driven mutation testing.

The requirements defined in this chapter are univocally identified by the paragraph id appearing on the left.

3.2 General Capabilities

FAQAS-SSS-REQ-001 FAQAS-framework shall support test suite evaluation based on code mutation.

FAQAS-SSS-REQ-002 FAQAS-framework shall support test suite augmentation based on code mutation.

FAQAS-SSS-REQ-003 FAQAS-framework shall support test suite evaluation based on data mutation.

FAQAS-SSS-REQ-004 FAQAS-framework should support test suite augmentation based on data mutation.

FAQAS-SSS-REQ-005 FAQAS-framework shall implement the code-driven test suite evaluation and test suite augmentation steps depicted in Figure 3.1 and Figure 3.2. Detailed explanation is provided in D2 Chapter 1.

FAQAS-SSS-REQ-006 FAQAS-framework shall implement the data-driven test suite evaluation and test suite augmentation steps depicted in Figure 3.3 and Figure 3.4. Detailed explanation is provided in D2 Chapter 2.

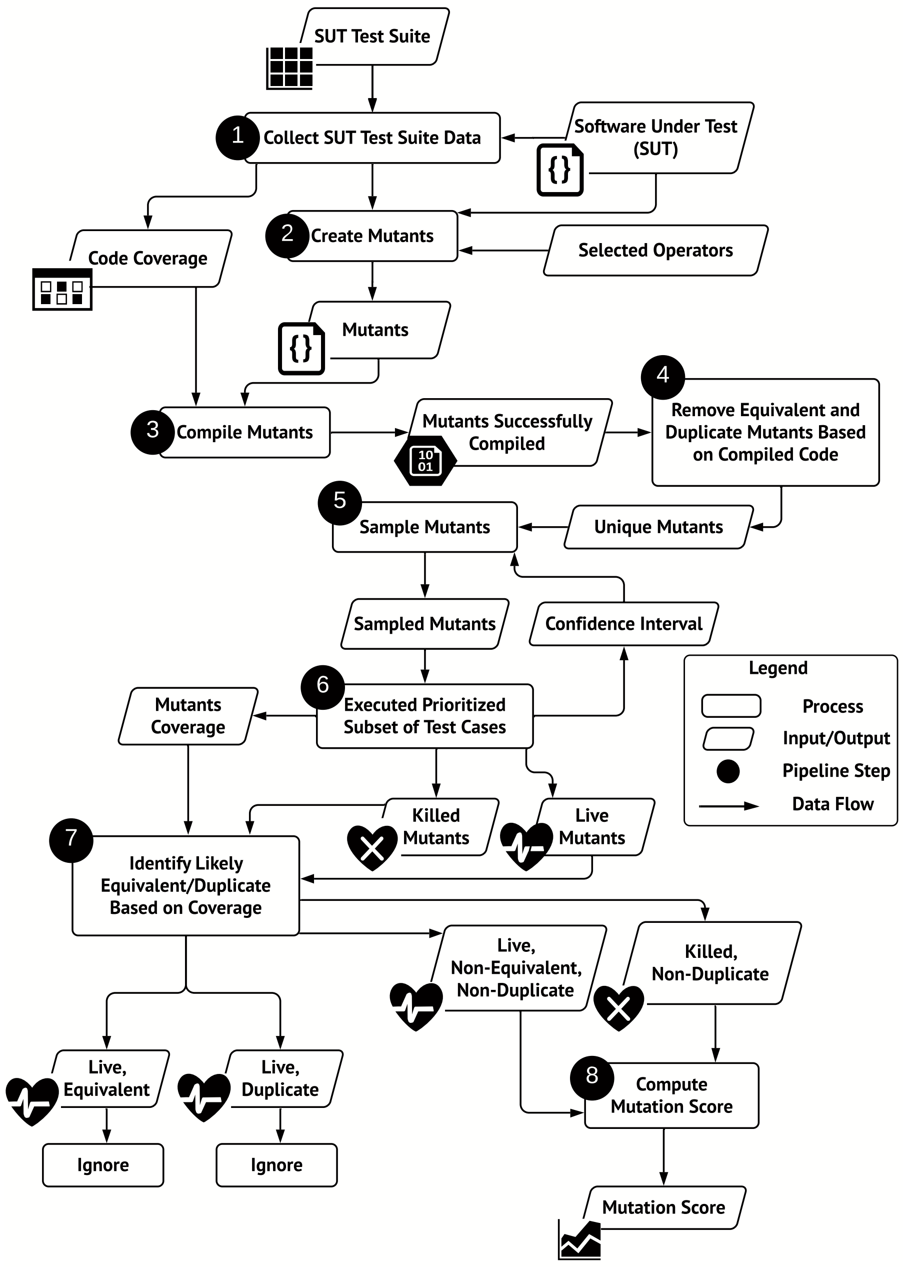

Figure 3.1: Code-driven Mutation Testing Process: Evaluation

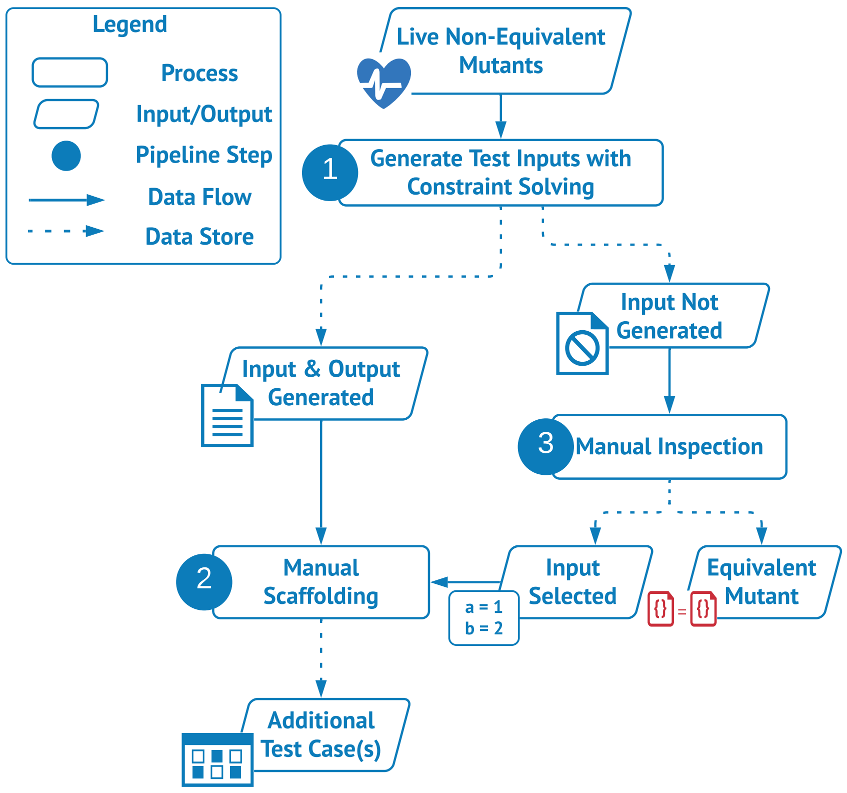

Figure 3.2: Code-driven Mutation Testing Process: Augmentation

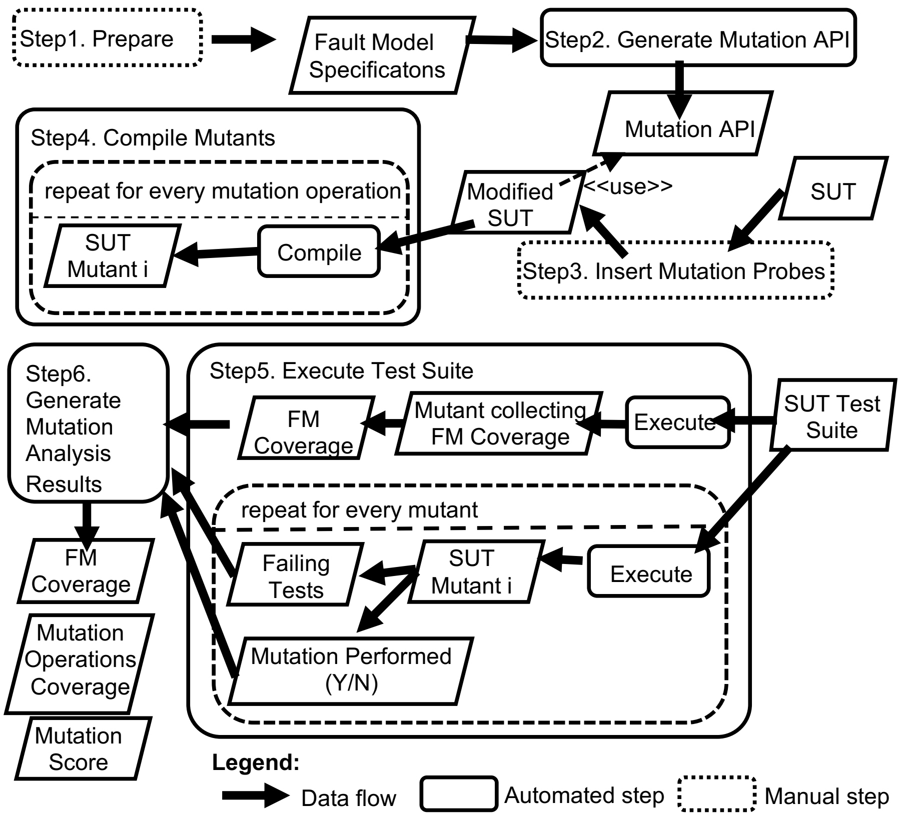

Figure 3.3: Data-driven Mutation Testing Process: Evaluation

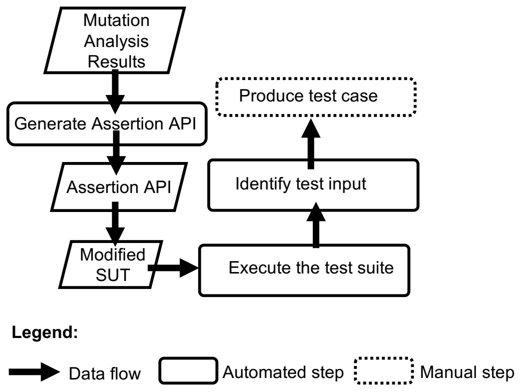

Figure 3.4: Data-driven Mutation Testing Process: Augmentation

3.3 General Constraints

FAQAS-SSS-REQ-007 Test suite evaluation for code-driven mutation testing shall be based on SRCIror[1] (detailed motivation is provided in D2).

FAQAS-SSS-REQ-008 The automated generation of test cases (i.e., the objective of test suite augmentation) shall rely on existing tools because it is an open, complex, research problem.

FAQAS-SSS-REQ-009 The automated generation of test cases for code-mutation may rely on KLEE, which is the most stable test case generation tool, based on WP2 evaluation.

3.4 Software Architecture

FAQAS-SSS-REQ-010 FAQAS-framework shall follow a modular architecture. The FAQAS Architecture is depicted in Figure 3.5. The FAQAS system shall consists of four components

- MASS (Mutation Analysis for Space Software), which implements test suite evaluation based on code-driven mutation.
- SEMuS (Symbolic Execution-based Mutation testing for Space), which implements test suite augmentation based on code-driven mutation.
- DAMAt (Data-driven Mutation Analysis), which implements test suite evaluation based on data-driven mutation.
- DAMTE (DAta-driven Mutation TEsting), which implements test suite augmentation based on data-driven mutation.

FAQAS-SSS-REQ-011 At high-level, the four FAQAS-framework components shall work as follows. All the components receive as input the software under test (SUT), the test suite to evaluate (SUT Test Suite), and a set of configurations. MASS generates a set of code-driven mutants, the mutation score, and a list of live mutants. SEMuS receive as input the list of live mutants and generate test cases that kill them. DAMAt generates a set of data-driven mutants, the mutation score, and a list of live mutants. DAMTE receive as input the list of live data-driven mutants and generate test cases that kill them.

[1] https://github.com/TestingResearchIllinois/srciror

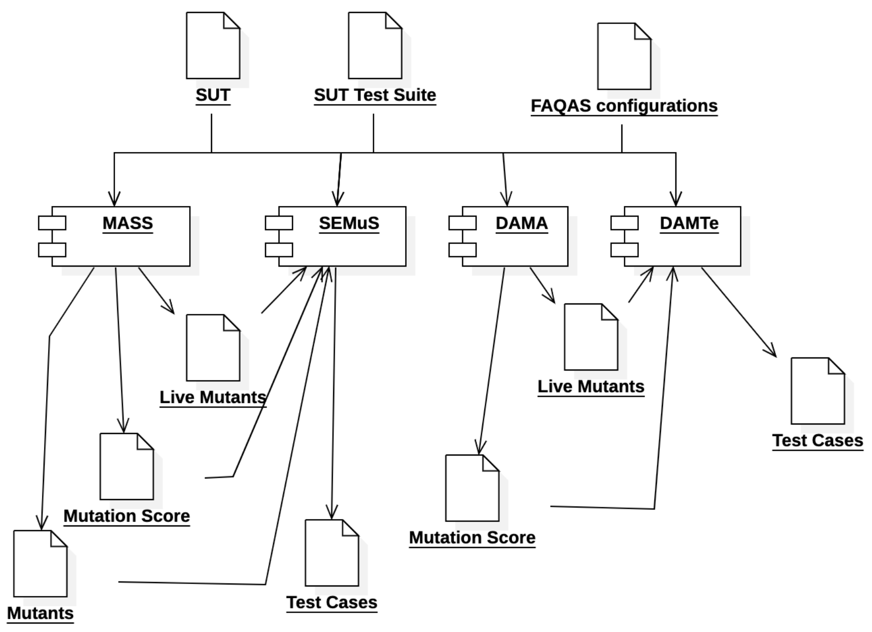

Figure 3.5: FAQAS Architecture. Arrows represent data-flow.

3.5 Operational Environment

FAQAS-SSS-REQ-012 The FAQAS-framework shall run on any operating system provided that a bash shell and Python 3 are installed on it.

FAQAS-SSS-REQ-013 The FAQAS-framework shall provide support to work on HPC infrastructure.

FAQAS-SSS-REQ-014 The FAQAS-framework shall be executed on the following minimum hardware specifications: 4 GB RAM and 10 GB of free disk space.

FAQAS-SSS-REQ-015 The FAQAS-framework shall be tested to work on the following Linux distributions: (i) Debian 9, (ii) Ubuntu 16, and (iii) CentOS 7.

3.6 Assumptions and dependencies

FAQAS-SSS-REQ-016 The FAQAS-framework shall process SUT built using either GCC Make [2] or WAF[3].

FAQAS-SSS-REQ-017 The FAQAS-framework shall process SUT compiled with GCC [4] versions above 4.

FAQAS-SSS-REQ-018 The FAQAS-framework shall process SUT test suites implemented as scripts (hereafter, *test suite script*) that trigger the execution of test cases (each test case is executed in a separate process). Test suite scripts could be implemented either as Makefiles or a Bash scripts. Each test case is executed through the invocation of a command in the script.

[2]https://gcc.gnu.org/onlinedocs/gccint/Makefile.html
[3]https://waf.io/
[4]https://gcc.gnu.org

Chapter 4

Requirements

4.1 Functional requirements

4.1.1 Test Suite Evaluation Based on Code-driven Mutation

The following requirements regard the Test Suite Evaluation Based on Code-Mutation functionality of the FAQAS-framework.

FAQAS-SSS-REQ-019 The FAQAS-framework shall implement a set of optimizations to make code-driven mutation analysis feasible (see D2).

Remark: The term mutation analysis refers to the process of assessing a test suite through code-driven mutation. In D2 it is also called code-driven test suite evaluation and code-driven test suite assessment.

FAQAS-SSS-REQ-020 The FAQAS-framework shall implement the following optimization steps:

1. Generate mutants based on code coverage;
2. Generate mutants based on the sufficient set of operators;
3. Discard equivalent and redundant mutants based on trivial compiler optimizations;
4. Sample mutants based on proportional uniform sampling, proportional method-based sampling, uniform fixed-size sampling, and uniform FSCI sampling;
5. Execute test suites prioritized and reduced based on code coverage;
6. Discard equivalent mutants based on code coverage.

FAQAS-SSS-REQ-021 The FAQAS-framework shall provide means to enable engineers to run code-driven mutation analysis on the SUT.

FAQAS-SSS-REQ-022 The FAQAS-framework shall provide means to enable engineers to select the optimization steps to be carried on the SUT.

FAQAS-SSS-REQ-023 The FAQAS-framework shall receive as input code coverage information of the SUT.

FAQAS-SSS-REQ-023-a The FAQAS-framework shall provide means to enable engineers to execute code-driven mutation analysis both when code coverage information is available and not available.

FAQAS-SSS-REQ-024 The FAQAS-framework shall support test case execution following the practice for the SUT (e.g., running the command `make test`).

FAQAS-SSS-REQ-025 The FAQAS-framework shall mutate the source code of the SUT.

FAQAS-SSS-REQ-026 The FAQAS-framework shall support C-coded software.

FAQAS-SSS-REQ-027 The FAQAS-framework shall mutate the SUT by applying a set of mutation operators that can be selected by the engineer.

FAQAS-SSS-REQ-028 The FAQAS-framework shall implement the set of operators listed in Table 4.1.

Table 4.1: Implemented set of mutation operators.

	Operator	**Description***
Sufficient Set	ABS	$\{(v, -v)\}$
	AOR	$\{(op_1, op_2) \mid op_1, op_2 \in \{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{/}, \texttt{\%}\} \wedge op_1 \neq op_2\}$ $\{(op_1, op_2) \mid op_1, op_2 \in \{\texttt{+=}, \texttt{-=}, \texttt{*=}, \texttt{/=}, \texttt{\%=}\} \wedge op_1 \neq op_2\}$
	ICR	$\{i, x) \mid x \in \{1, -1, 0, i+1, i-1, -i\}\}$
	LCR	$\{(op_1, op_2) \mid op_1, op_2 \in \{\&\&, \Vert\} \wedge op_1 \neq op_2\}$ $\{(op_1, op_2) \mid op_1, op_2 \in \{\&=, \vert=, \&=\} \wedge op_1 \neq op_2\}$ $\{(op_1, op_2) \mid op_1, op_2 \in \{\&, \vert, \&\&\} \wedge op_1 \neq op_2\}$
	ROR	$\{(op_1, op_2) \mid op_1, op_2 \in \{\texttt{>}, \texttt{>=}, \texttt{<}, \texttt{<=}, \texttt{==}, \texttt{!=}\}\}$ $\{(e, !(e)) \mid e \in \{\texttt{if(e)}, \texttt{while(e)}\}\}$
	SDL	$\{(s, \texttt{remove}(s))\}$
	UOI	$\{(v, \texttt{-}v), (v, v\texttt{-}), (v, \texttt{++}v), (v, v\texttt{++})\}$
OODL	AOD	$\{((t_1\, op\, t_2), t_1), ((t_1\, op\, t_2), t_2) \mid op \in \{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{/}, \texttt{\%}\}$
	LOD	$\{((t_1\, op\, t_2), t_1), ((t_1\, op\, t_2), t_2) \mid op \in \{\}\}$
	ROD	$\{((t_1\, op\, t_2), t_1), ((t_1\, op\, t_2), t_2) \mid op \in \{\texttt{>}, \texttt{>=}, \texttt{<}, \texttt{<=}, \texttt{==}, \texttt{!=}\}\}$
	BOD	$\{((t_1\, op\, t_2), t_1), ((t_1\, op\, t_2), t_2) \mid op \in \{\&, \vert, \wedge\}\}$
	SOD	$\{((t_1\, op\, t_2), t_1), ((t_1\, op\, t_2), t_2) \mid op \in \{», «\}\}$
Other	LVR	$\{(l_1, l_2) \mid (l_1, l_2) \in \{(0, -1), (l_1, -l_1), (l_1, 0), (\mathit{true}, \mathit{false}), (\mathit{false}, \mathit{true})\}\}$

*Each pair in parenthesis shows how a program element is modified by the mutation operator. The left element of the pair is replaced with the right element. We follow standard syntax (see D2). Program elements are literals (l), integer literals (i), boolean expressions (e), operators (op), statements (s), variables (v), and terms (t_i, which might be either variables or literals).

FAQAS-SSS-REQ-029 The FAQAS-framework shall apply all available mutation operators in case they are not specified.

FAQAS-SSS-REQ-029–a The FAQAS-framework shall generate all the mutants defined by each operator to be applied.

FAQAS-SSS-REQ-030 The FAQAS-framework shall store every generated mutant on a directory tree that follows the structure of the source directory tree of the SUT.

Remark: Every source file is replaced by a folder; the folder has the same name of the file. The folder contains all the mutants generated for that file.

FAQAS-SSS-REQ-031 The FAQAS-framework shall generate mutants with unique name identifiers.

FAQAS-SSS-REQ-032 The FAQAS-framework shall generate mutants with names that results from the conjunction of the following information: source file name, mutated function name, mutated line, mutation operator name, mutation operation, mutated "column" (i.e., char position from the beginning of the line).

FAQAS-SSS-REQ-033 The FAQAS-framework shall compile mutants incrementally to save compilation time.

FAQAS-SSS-REQ-034 The FAQAS-framework shall disregard equivalent and redundant mutants based on trivial compiler equivalence.

FAQAS-SSS-REQ-035 The FAQAS-framework shall work with compilation scripts that are modified by the engineer according to rules specific for the FAQAS-framework. The modified compilation script (i) shall not contain debugging nor coverage flags, (ii) shall contain a placeholder for the compiler optimization option, and (iii) shall contain a 'sort' command in the source dependency list to ensure that source files are always compiled in the same order.

FAQAS-SSS-REQ-036 The FAQAS-framework shall compile every mutant with the *O0*, *O1*, *O2*, *O3*, *Ofast*, and *Os* GCC compiler optimisation options[1].

FAQAS-SSS-REQ-037 The FAQAS-framework shall generate a SHA512 hash for every compiled mutant to enable mutant comparisons.

FAQAS-SSS-REQ-038 The FAQAS-framework shall disregard mutants that generate a compilation error.

FAQAS-SSS-REQ-039 The FAQAS-framework shall not disregard mutants that produce a compilation warning.

FAQAS-SSS-REQ-040 The FAQAS-framework shall a produce a list of nonequivalent and nonredundant mutants based on trivial compiler equivalence.

FAQAS-SSS-REQ-041 The FAQAS-framework shall generate a set of prioritized and reduced test suites for every mutant. The generation of such test suites should be based on the PrioritizeAndReduce Algorithm (See D2).

FAQAS-SSS-REQ-042 The FAQAS-framework shall provide an option to execute only the prioritized and reduced set of test cases when testing a mutant affecting a certain statement. When the option is disabled, the FAQAS-framework shall execute the whole test suite for every mutant. The execution of a test suite shall be stopped when a mutant is killed.

FAQAS-SSS-REQ-043 The FAQAS-framework shall support strong mutation analysis (see D2).

[1]https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

FAQAS-SSS-REQ-044 The FAQAS-framework shall sample the mutants to be executed.

FAQAS-SSS-REQ-045 The FAQAS-framework shall support the mutant selection strategy *all mutants* (see D2).

FAQAS-SSS-REQ-046 The FAQAS-framework shall support the mutant selection strategy *proportional uniform sampling* (see D2).

FAQAS-SSS-REQ-047 The FAQAS-framework shall support the mutant selection strategy *proportional method-based sampling* (see D2).

FAQAS-SSS-REQ-048 The FAQAS-framework shall enable engineers to provide a sampling rate if *proportional uniform sampling* or *proportional method-based sampling* is selected.

FAQAS-SSS-REQ-049 The FAQAS-framework shall support the mutant selection strategy *uniform FSCI sampling* (see D2).

FAQAS-SSS-REQ-050 The FAQAS-framework shall compile a mutant by running the build script of the original program.

FAQAS-SSS-REQ-051 The FAQAS-framework shall support simple runtime optimizations:

1. stopping the execution of the test suite when the mutant has been killed,
2. executing only those test cases that cover the mutated statements, and
3. rely on timeouts to automatically detect infinite loops introduced by mutation.

FAQAS-SSS-REQ-052 The FAQAS-framework shall compile and execute mutants till a termination criterion is met. The termination criterion depends on the mutants selection strategy (see D2 for details):

- *all mutants*: all mutants have been executed
- *proportional uniform sampling*: a number of mutants matching the selected percentage has been executed
- *proportional method-based sampling*: a number of mutants matching the selected percentage for all methods in the SUT has been executed
- *uniform fixed-size sampling*: a number of mutants matching the selected value has been executed
- *uniform FSCI sampling*: the confidence interval computed is smaller than 10%.

FAQAS-SSS-REQ-053 The FAQAS-framework shall identify equivalent mutants based on code coverage information using the distance criterion D_C (see D2).

FAQAS-SSS-REQ-054 The FAQAS-framework shall compute the mutation score of the SUT based on mutation results. Equivalent mutants shall not be considered for the computation of the mutation score (see D2).

FAQAS-SSS-REQ-055 The FAQAS-framework shall produce two prioritized lists (list A and list B) of live and nonequivalent mutants. List A shall contain live mutants that are different from the SUT (i.e., nonequivalent) and different from each other. Ideally, each mutant in list A should represent a distinct limitation of the SUT. List A contains mutants that have a distance D_C (strictly) greater than zero; in other words, the distance between each pair of mutants appearing in list A should be greater than zero. List B contains all the remaining live, nonequivalent mutants. The two lists are sorted in decreasing order, based on the distance D_C from the original program.

Remark: Equivalent mutants, which are discarded, have a distance D_C from the original program equal to zero. The generation of the list proceeds as follows. First, a list (list X) containing all the mutants, sorted by distance D_C, is created. Iteratively, each mutant in list X is compared with each mutant in list A. If the distance is greater than zero, the mutant is added to list A, otherwise it is added to list B.

FAQAS-SSS-REQ-056 The FAQAS-framework shall report a summary of the results obtained in every step of the mutation analysis process. The generated reports shall include information about the total number of mutants generated, the total number of mutants filtered by compiler optimisations, the type of mutants sampling employed, the total number of mutants executed, the total number of killed mutants, the total number of live mutants, the mutation score, the statement coverage, and the location of the two prioritized lists of nonequivalent mutants.

4.1.2 Test Suite Augmentation Based on Code-driven Mutation

The following requirements regard the Test Suite Augmentation functionality aiming to generate test cases that kill mutants generated with the Code-driven Mutation functionality of the FAQAS-framework.

FAQAS-SSS-REQ-057 The FAQAS-framework shall support test suite augmentation based on code mutation.

FAQAS-SSS-REQ-058 The FAQAS-framework shall rely on the KLEE test case generation tool to support test suite augmentation.

FAQAS-SSS-REQ-059 For each live mutant, the FAQAS-framework should generate test scaffoldings that are processed by KLEE.

FAQAS-SSS-REQ-060 The FAQAS-framework shall provide the engineer with the ability to configure the generated test scaffoldings.

Remark: For example, the FAQAS-framework shall provide the engineer with the ability to refine the generated assertions. Since assertions concern output and state variables, it is necessary to verify that all the necessary output/state variables had been referred in assertion. Indeed, in C, with pointers and pointers to pointers, it is not possible to have a precise automated identification of output/state variables.

FAQAS-SSS-REQ-061 The FAQAS-framework shall generate a tentative unit test case (i.e., a source file in C) that kills the mutants.

Remark: The FAQAS-framework may generate test cases consisting of (i) an invocation of the function under test (i.e., the function targeted by the mutation), (ii) its assigned arguments, and (iii) an assertion that verifies results.

4.1.3 Test Suite Evaluation Based on Data-driven Mutation

The following requirements regard the Test Suite Evaluation Based on the Data-driven Mutation Testing functionality of the FAQAS-framework.

FAQAS-SSS-REQ-062 The FAQAS-framework shall work with a fault model and a data model for the SUT specified according to D2.

FAQAS-SSS-REQ-063 The FAQAS-framework shall provide a mutation analysis API (i.e., predefined functions to perform data mutation for buffers).

FAQAS-SSS-REQ-064 The FAQAS-framework shall automatically generate the data mutation probes to be integrated within the SUT.

FAQAS-SSS-REQ-065 The engineer shall manually modify the source code of the SUT to integrate
P-11 mutation probes into it.

FAQAS-SSS-REQ-066 The FAQAS-framework shall support C-coded software and C++-coded

P-11 software.

FAQAS-SSS-REQ-067 The FAQAS-framework shall support the mutation of buffers implemented

P-11 as arrays.

FAQAS-SSS-REQ-068 The FAQAS-framework shall support the mutation of C/C++ standard data

types.

FAQAS-SSS-REQ-069 The FAQAS-framework shall provide the engineer with the ability to choose to mutate all the instances of the data item, one instance of the data item, or a percentage of the
P-11 instances.

FAQAS-SSS-REQ-070 The FAQAS-framework shall implement the set of operators listed in Ta-

P-11 ble 4.2.

FAQAS-SSS-REQ-071 For every mutation operator defined in the Fault Model, the FAQAS-

P-11 framework shall generate one or more mutants performing a mutation operation.

FAQAS-SSS-REQ-072 The FAQAS-framework shall identify each mutant with a unique numeri-

P-11 cal identifier.

FAQAS-SSS-REQ-073 The FAQAS-framework will automatically generate a table containing the definition of all the generated mutants and their identifiers.

FAQAS-SSS-REQ-074 The FAQAS-framework shall work with compilation scripts that are modified by the engineer according to rules specific for the FAQAS-framework.

FAQAS-SSS-REQ-075 The FAQAS-framework shall enable the compilation of a version of the SUT that traces the data items (targeted by mutation) that are covered by each test case.

FAQAS-SSS-REQ-076 The FAQAS-framework shall derive, from logs generated during the execution of the SUT test suite, the data items exercised by each test case.

P-11

FAQAS-SSS-REQ-077 The FAQAS-framework shall provide, for each mutant, a list of all test cases covering the data item targeted by the mutant.

FAQAS-SSS-REQ-078 The FAQAS-framework shall compile a version of the SUT for every generated mutant.

FAQAS-SSS-REQ-079 For each mutation operator, the FAQAS-framework shall execute only the test cases exercising the data item targeted by the mutation operator.

P-11

FAQAS-SSS-REQ-080 The FAQAS-framework shall execute every mutant against all the test cases exercising the data item targeted by the mutation operator.

P-11

FAQAS-SSS-REQ-081 The FAQAS-framework shall support test case execution following the practice for the SUT (e.g., running the command `make test`).

FAQAS-SSS-REQ-082 For each mutation operation, the FAQAS-framework shall provide the engineer with information on the number of instances of the data item that have been mutated and not mutated by the selected mutation operation.

FAQAS-SSS-REQ-083 The FAQAS-framework shall automatically determine if the mutation operation has been performed at least once.

FAQAS-SSS-REQ-084 The FAQAS-framework shall automatically determine if a mutant has been killed.

P-11

FAQAS-SSS-REQ-085 The FAQAS-framework shall compute the fault model coverage (see D2) on the mutation results.

P-11

FAQAS-SSS-REQ-086 The FAQAS-framework shall compute the mutation operation coverage (see D2) on the mutation results.

FAQAS-SSS-REQ-087 The FAQAS-framework shall compute the mutation score (see D2) based on

P-11 the mutation results.

FAQAS-SSS-REQ-088 The FAQAS-framework shall produce a list of all the mutants and their
P-11 status.

FAQAS-SSS-REQ-089 The FAQAS-framework shall describe the status of a mutant relative to the
P-11 Fault Model Coverage as `COVERED`, if the or `NOT_COVERED`.

FAQAS-SSS-REQ-090 The FAQAS-framework shall mark all mutants targeting a data item exer-
P-11 cised by the test suite as `COVERED`.

FAQAS-SSS-REQ-091 The FAQAS-framework shall mark all mutants targeting a data item not
P-11 exercised by the test suite as `NOT_COVERED`

FAQAS-SSS-REQ-092 The FAQAS-framework shall describe the status of a mutant relative to the
P-11 Mutation Operation Coverage as `APPLIED` or `NOT_APPLIED`.

FAQAS-SSS-REQ-093 The FAQAS-framework shall mark all mutants that performed the corre-
P-11 sponding mutation operation at least once as `APPLIED`

FAQAS-SSS-REQ-094 The FAQAS-framework shall mark all mutants that did not perform the
P-11 corresponding mutation operation at least once as `NOT_APPLIED`.

FAQAS-SSS-REQ-095 The FAQAS-framework shall describe the status of a mutant relative to the
P-11 Mutation Score as `KILLED` or `LIVE`.

FAQAS-SSS-REQ-096 The FAQAS-framework shall mark all mutants that caused at least a test
P-11 case failure as `KILLED`.

FAQAS-SSS-REQ-097 The FAQAS-framework shall mark all mutants that did not cause at least a test case failure as `LIVE`.

FAQAS-SSS-REQ-098 The FAQAS-framework shall provide the engineer with the means to inspect the mutation results. The FAQAS-framework data-driven mutation results shall include the list of data types not covered by any test case, the list of data types exercised by each test case, the list of live and killed mutants, and the result of every executed test case.

4.1.4 Test Suite Augmentation Based on Data-driven Mutation

The following requirements regard the Test Suite Augmentation functionality that aims to generate test cases that kill mutants generated with the Data-driven Mutation functionality of the FAQAS-framework.

FAQAS-SSS-REQ-099 The FAQAS-framework shall generate a version of the testing API that shall contain reachability assertions that enable KLEE to generate inputs that reach the mutation.

FAQAS-SSS-REQ-0100 The FAQAS-framework shall not fully automate the Data-driven Test Suite Augmentation functionality.

Remark: The engineer shall execute KLEE, after performing the required scaffolding, if necessary. Also, the engineer shall implement a test case based on KLEE's output.

4.2 System interface requirements

FAQAS-SSS-REQ-0101 The user interface of the FAQAS-framework is provided through the command line.

FAQAS-SSS-REQ-0102 The FAQAS-framework shall process code coverage information recorded using the format of `gcov`[2].

FAQAS-SSS-REQ-0103 The FAQAS-framework shall automatically execute gcov, when available, to generate code coverage information.

FAQAS-SSS-REQ-0104 The FAQAS-framework shall support the following commands:

- `MASS`:
  - `PrepareSUT`: command to prepare the SUT and collect information about the SUT test suite.
  - `GenerateMutants`: command to generate mutants from the SUT source code.
  - `CompileOptimizedMutants`: command to compile the mutants with the multiple optimisation levels.
  - `OptimizedPostProcessing`: command to disregard equivalent and redundant mutants based on compiler optimisations.
  - `GeneratePTS`: command to generate the prioritized and reduced test suites.
  - `ExecuteMutants`: command to execute mutants against the SUT test suite.
  - `IdentifyEquivalents`: command to identify equivalent mutants based on code coverage.
  - `MutationScore`: command to compute the mutation score and final reporting.
  - `PrepareMutants_HPC`: command to prepare the mutants workspace for the execution on HPCs.
  - `ExecuteMutants_HPC`: command to execute mutants on HPCs.
  - `PostMutation_HPC`: command to assess past mutant executions, and to decide whether more mutant executions are needed.

P-11

- `DAMAt`:
  - `DAMAt_probe_generation`: command to generate the data mutation probes.

[2]https://gcc.gnu.org/onlinedocs/gcc/Gcov.html

  - DAMAt_mutants_launcher: command to execute the mutants against the SUT. test suite.

FAQAS-SSS-REQ-0105 The FAQAS-framework shall generate the following log files:

- MASS
  - output concerning SUT compilation process
  - output concerning execution of SUT test cases
  - output concerning the execution of the CompileOptimizedMutants command
  - output concerning the execution of the OptimizedPostProcessing command
  - output concerning the execution of the GeneratePTS command
  - output concerning the execution of the ExecuteMutants command
  - output concerning the execution of the IdentifyEquivalents command
  - output concerning the execution of the MutationScore command
  - output concerning the execution of the PrepareMutants_HPC command
  - output concerning the execution of the ExecuteMutants_HPC command
  - output concerning the execution of the PostMutation_HPC command

P-11

- DAMAt
  - output concerning the execution of the DAMAt_probe_generation command
  - output concerning the execution of the DAMAt_mutants_launcher command

FAQAS-SSS-REQ-0106 The FAQAS-framework shall be configured through texts files.

FAQAS-SSS-REQ-0107 The FAQAS-framework configuration file shall be set with the following parameters:

- FAQAS-framework installation path
- FAQAS-framework workspace path
- SUT relevant paths (e.g., source code folder, test folder, coverage folder)
- SUT compilation command
- SUT test case execution command
- FAQAS-framework execution environment mode (i.e., single machine, HPC)
- FAQAS-framework trivial compiler optimisation flags
- FAQAS-framework sampling type
- FAQAS-framework sampling rate
- FAQAS-framework test suite prioritization type

FAQAS-SSS-REQ-0108 The FAQAS-framework shall provide built-in features to process the SUT test harness Google Test[3] and the Basic mathematical Library Test Suite (BLTS)[4].

FAQAS-SSS-REQ-0109 The FAQAS-framework shall provide means to configure its execution with any test harness infrastructure for C/C++ software.

FAQAS-SSS-REQ-0110 The FAQAS-framework shall provide the engineer with the ability to configure the environment variables used to tune the mutation testing process.

4.3 Adaptation and missionization requirements

FAQAS-SSS-REQ-0111 The FAQAS-framework shall not be used in flight. Precisely, it shall not be executed to test software running on a spacecraft hardware on orbit.

FAQAS-SSS-REQ-0112 The FAQAS-framework shall be used in the development environment.

4.4 Computer resource requirements

FAQAS-SSS-REQ-0113 The FAQAS-framework shall run on Linux operating systems.

4.5 Security requirements

FAQAS-SSS-REQ-0114 The FAQAS-framework shall not use network connections.

4.6 Safety requirements

FAQAS-SSS-REQ-0115 The FAQAS-framework should not be used to assess test cases that are executed with target hardware in the loop, to avoid safety hazards.

4.7 Reliability and availability requirements

FAQAS-SSS-REQ-0116 The FAQAS-framework shall work according to its functional specifications every time it is invoked.

[3]https://github.com/google/googletest
[4]https://essr.esa.int/project/mlfs-mathematical-library-for-flight-software

FAQAS-SSS-REQ-0117 The FAQAS-framework shall not provide an upper bound for mutation testing execution time.

Remark: Mutation testing execution time depends on both the number of mutants to be executed and the duration of the test suite execution, information that cannot be foreseen before execution.

4.8 Quality requirements

FAQAS-SSS-REQ-0118 Usability. Software engineers shall be able to successfully use the FAQAS-framework after reading its documentation.

FAQAS-SSS-REQ-0119 The FAQAS-framework shall be used in any environment matching the characteristics indicated in this document.

FAQAS-SSS-REQ-0120 The FAQAS-framework shall comply with software development standards. The software development process shall follow ECSS guidelines as per SoW.

4.9 Design requirements and constraints

FAQAS-SSS-REQ-0121 The FAQAS-framework shall be released with ESA Software Community Licence Permissive – v2.3, as defined at https://essr.esa.int/. Any reused component should be compatible with the abovementioned licence.

FAQAS-SSS-REQ-0122 The FAQAS-framework shall be implemented using the bash shell script language, the Python programming language, and the C/C++ programming language.

FAQAS-SSS-REQ-0123 FAQAS-framework shall mutate the source code of the SUT not the intermediate representation.

4.10 Adaptation and installation requirements

FAQAS-SSS-REQ-0124 The FAQAS-framework should be installed using an installer software.

4.11 Software operations requirements

FAQAS-SSS-REQ-0125 The FAQAS-framework shall be executed at anytime, in an environment matching the characteristics indicated in this document.

4.12 Software maintenance requirements

FAQAS-SSS-REQ-0126 One year maintenance support shall be provided as per SoW.

4.13 System and software observability requirements

FAQAS-SSS-REQ-0127 The debugging of the FAQAS-framework shall be possible if and only if all the temporary files generated by the system are kept.

Table 4.2: Data-driven mutation operators

Fault Class	Types	Parameters	Description
Value above threshold (VAT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	Replaces the current value with a value above the threshold T for a delta (Δ). It simulates a value that is out of the nominal case and shall trigger a response from the system that shall be verified by the test case (e.g., the system may continue working but an alarm shall be triggered). Not applied if the value is already above the threshold. ***Data mutation procedure:*** $v' = (T + \Delta)(\mathit{ifv} \leq T); v' = v(\mathit{otherwise});$
Value below threshold (VBT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	Replaces the current value with a value below the threshold T for a delta (Δ). It simulates a value that is out of the nominal case and shall trigger a response from the system that shall be verified by the test case (e.g., the system may continue working but an alarm shall be triggered). Not applied if the value is already below the threshold. ***Data mutation procedure:*** $v' = (T - \Delta)(\mathit{ifv} \geq T); v' = v(\mathit{otherwise})$
Value out of range (VOR)	I,L,F,D,H	MIN: minimum valid value MAX: maximum valid value Δ: delta, difference with respect to minimum/maximum valid value	Replaces the current value with a value out of the range $[MIN; MAX]$. It simulates a value that is out of the nominal range and shall trigger a response from the system that shall be verified by the test case (e.g., the system may continue working but an alarm shall be triggered). Not applied if the value is already out of range. ***Data mutation procedure 1:*** $v' = (MIN - \Delta)(\mathit{if}MIN \leq v \leq MAX); v' = v(\mathit{otherwise})$ ***Data mutation procedure 2:*** $v' = (MAX + \Delta)(\mathit{if}MIN \leq v \leq MAX); v' = v(\mathit{otherwise})$
Bit flip (BF)	B	MIN: lower bit MAX: higher bit STATE: mutate only if the bit is in the given state (i.e., 0 or 1). VALUE: number of bits to mutate	A number of bits randomly chosen in the positions between MIN and MAX (included) are flipped. If STATE is specified, the mutation is applied only if the bit is in the specified state; the value -1 indicates that any state shall be considered for mutation. Parameter VALUE specifies the number of bits to mutate. ***Data mutation procedure:*** the operator flips VALUE randomly selected bit if they are in the specified state.
Invalid numeric value (INV)	I,L,F,D,H	MIN: lower valid value MAX: higher valid value	Replace the current value with a mutated value that is legal (i.e., in the specified range) but different than current value. It simulates the exchange of data that is not consistent with the state of the system. ***Data mutation procedure:*** Replace the current value with a different value randomly sampled in the specified range.
Illegal Value (IV)	I,L,F,D,H	VALUE: illegal value that is observed	Replace the current value with a value that is equal to the parameter *VALUE*. ***Data mutation procedure:*** $v' = \mathit{VALUE}(\mathit{ifv} \neq \mathit{VALUE}); v' = v(\mathit{otherwise})$
Anomalous Signal Amplitude (ASA)	I,L,F,D,H	T: change point Δ: delta, value to add/remove VALUE: value to multiply	The mutated value is derived by amplifying the observed value by a factor *V* and by adding/removing a constant value Δ from it. It is used to either amplify or reduce a signal in a constant manner to simulate unusual signals. The parameter *T* indicates the observed value below which instead of adding we subtract . ***Data mutation procedure:*** $v' = T + ((v - T) * \mathit{VALUE}) + \Delta(\mathit{if}\ v \geq T); v' = T - ((T - v) * \mathit{VALUE}) - \Delta(\mathit{if}\ v < T);$
Signal Shift (SS)	I,L,F,D,H	Δ: delta, value by which the signal should be shifted	The mutated value is derived by adding a value Δ to the observed value. It simulates an anomalous shift in the signal. ***Data mutation procedure:*** $v' = v + \Delta$
Hold Value (HV)	I,L,F,D,H	V: number of times to repeat the same value	This operator keeps repeating an observed value for V times. It emulates a constant signal replacing a signal supposed to vary. ***Data mutation procedure:*** $v' = \mathit{previous}\ v'(\mathit{if\ counter} \leq V); v' = \mathit{votherwise}$
Fix value above threshold (FVAT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	It is the complement of VAT and implements the same mutation procedure as VBT but we named it differently because it has a different purpose. Indeed, it is used to verify that test cases exercising exceptional cases are verified correctly. In the presence of a value above the threshold, it replaces the current value with a value below the threshold T for a delta Δ. ***Data mutation procedure:*** $v' = v(\mathit{ifv} > T); V' = (T - \Delta)(\mathit{otherwise})$
Fix value below threshold (FVBT)	I,L,F,D,H	T: threshold Δ: delta, difference with respect to threshold	It is the counterpart of FVAT for the operator VBT. ***Data mutation procedure:*** $v' = v(\mathit{ifv} < T); v' = (T + \Delta)(\mathit{otherwise})$
Fix value out of range (FVOR)	I,L,F,D,H	MIN: minimum valid value MAX: maximum valid value	It is the complement of VOR and implements the same mutation procedure as INV but we named it differently because it has a different purpose. Indeed, it is used to verify that test cases exercising exceptional cases are verified correctly. ***Data mutation procedure:*** $v' = v(\mathit{if}MIN \leq v \leq MAX); v' = \mathit{random}(\mathit{MIN}, \mathit{MAX})(\mathit{otherwise})$

Legend: I: INT, L: LONG INT, F: FLOAT, D: DOUBLE, B: BIN, H: HEX

Chapter 5

Verification, validation and system integration

5.1 Verification and validation process requirements

FAQAS-SSS-REQ-0128 Every mutation operator implemented by the FAQAS-framework shall be tested by a dedicated unit test case.

FAQAS-SSS-REQ-0129 The FAQAS-framework shall enable the correct computation of the mutation score for the FAQAS case study systems indicated in deliverable D2.

FAQAS-SSS-REQ-0130 Since in the FAQAS-framework units are self-contained (i.e., they do not interact with other units but simply store results in files processed by other units), the integration of pairs of unit shall not be tested.

FAQAS-SSS-REQ-0131 The FAQAS-framework shall be tested as a whole, based on tutorial examples described in SUM.

FAQAS-SSS-REQ-0132 Unit test cases shall be executed on a x86-64 desktop PC.

FAQAS-SSS-REQ-0133 Unit test cases execution time should not exceed one day.

5.1.1 Validation approach

FAQAS-SSS-REQ-0134 SnT shall perform a preliminary validation of the delivered framework.

FAQAS-SSS-REQ-0135 FAQAS industry partners shall use the system at their premises to validate it.