

Title

Sasja Gillissen, Martin Huijben, Martijn Terpstra

September 26, 2016

1 MARTIN Introduction

State the objectives and overview of the document at a high-level.

2 SASJA Test Goal

What is the overall goal of the testing effort, what are the final deliverables, who are the stakeholders, i.e., for whom are you doing it, applicable laws and (international) standards.

3 SASJA The Product

Identification of the SUT: What is the product (SUT – System Under Test) being tested, its version, its operation context, required platform, its interfaces, and how is it executed.

4 TERPSTRA The Specification

The full specification is included in appendixA

5 MARTIN Risks

What are the risks of the product (at a high level), of the development process, and of the test process. How are risks handled and mitigated.

6 SASJA Test Environment

What is the (controlled) environment in which experiments are performed, what is the test architecture, i.e., how are SUT and test system positioned and connected, which environment and infrastructure (hardware, software, middleware, databases, libraries, . . .) are required for testing, how to access

the SUT and its interfaces, which stubs and drivers are needed, are tests performed in a laboratory, production, or user environment.

7 MARTIN Quality Characteristics

Which quality characteristics are tested (IS 9126 or other quality model: functionality, reliability, usability, . . .),

8 MARTIN Levels and Types of Testing

Which levels and types of tests are performed: (V-model: unit, integration, module, system, acceptance, . . .), which units, components, subsystems, . . . are tested and for what, accessibility (white/black box), verification vs. validation tests, . . .

9 TERPSTRA Who will do the Testing

Who tests what, and what are the roles: developer, (independent) tester, user, alpha, certification, . . .),

10 SASJA Test Generation Techniques

As far as already known or required, e.g., by applicable standards: black-box (equivalence partitioning, boundary value analysis, error guessing, cause-effect graphing, decision tables, state transitions, use case testing, exploratory testing, . . .), white-box (path, statement, (multiple) condition, decision/branch, function, call, loop, MC/DC coverage, . . .), mutation testing, combinatorial testing, . . .

11 TERPSTRA Test Automation

As far as applicable, which parts of the testing will be automated, which test tools will be used in the various phases of the testing process (planning, preparation, test generation, test execution, completion), which tests are performed manually, what is automated, and which tools have to be obtained or developed.

12 MARTIN Exit Criteria

What are the criteria for going from one test phase to the next, when is testing finished, when is the product considered sufficiently tested, what are the (final) evaluation criteria.

13 TERPSTRA Testware

Which test products are recorded, consolidated, and kept for reuse.

14 TERPSTRA Issue Registration

How are issues (defects) registered, analysed, reported, and handled.

A Specification

A.1 Overall use of the program

the program will continuously prompt the user for input. It will continue to prompt the user for input until it receives the `exit()` function as input and then terminates.

On invalid input the program the program will show an error indicating the nature of the invalid input, but will not terminate.

Valid input is on of the following

- A function assignment
- A variables assignment
- An expression
- An equality test

A.2 Memory

The program will keep defined variables and function in memory until the program terminates.

The program will be initialized with the following variables

- `pi`, whose value is 3.141592653589793
- `e`, whose value is 2.718281828459045

The program will be initialized with the following functions

- `floor`, which rounds down
- `ceil`, which rounds up
- `round`, which rounds to the nearest whole number.
- `log`
- `ln`, which

- `sqrt`, which returns the square root of its
- `root`, which returns the Nth root of its first argument. (WARNING: does not work properly)
- `exit`, which will terminate the program

BTW floor, ceil and round work incorrectly for NEGATIVE numbers

A.3 Expression

A valid expression is defined using Context-free Grammar

Expression \rightarrow Separator Expression Separator
 Expression \rightarrow (Expression)
 Expression \rightarrow Number
 Expression \rightarrow Variable
 Expression \rightarrow PrefixFunction Separator (Expression)
 Expression \rightarrow Expression Separator InfixFunction Separator Expression
 Separator \rightarrow SPACE | ϕ | Separator Separator
 Number \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Number Number
 Variable \rightarrow Name
 PrefixFunction \rightarrow Name
 Name \rightarrow SmallLetter | CapitalLetter | NameName
 SmallLetter \rightarrow a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
 CapitalLetter \rightarrow A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
 InfixFunction \rightarrow / | * | - | + | ^
variable names have a max length

A.4 Variable assignments

A variable assignment is in the form

VARIABLENAME = EXPRESSION

Where a valid VARIABLENAME is a string of 1 or more letters and is case-sensitive.

A.5 Equality tests

An equality test is in the form

EXPRESSION1 = EXPRESSION2

Where EXPRESSION1 and EXPRESSION2 are valid expressions and EXPRESSION1 is **not** the name of a variable

A.6 Functions assignments

A function assignment is in the form

FUNCTIONNAME(ARGUMENT) := EXPRESSION

A.7 Expression parsing

Intermediate steps and calculations are show The result of the expression is shown