ETH zürich

FS 2020

Software Requirements Specification

for

CAS

Version 0.2.1

Prepared by L.Autunno, L.Knirsch, N.Baumann

Group K

16.03.2020

Table of Contents

Revision History

Name	Date	Release Description	Version
Ioannis Athanasiadis	03/10/20	Adaptation of Karl E. Wiegers template for Software Engineering Course in ETHZ.	0.2
L. Knirsch L. Autunno N. Baumann	03/16/20	Writing first Requirements	0.2.1
L. Knirsch L. Autunno N. Baumann	03/30/20	Fixing Requirements from feedback	0.2.2

1. Introduction

The purpose of this document is to describe the software CAS, developed for the course “Software Engineering” at ETH Zurich.

1.1 Document Conventions

This document is based on the SRS template given in the lecture “Software Engineering”, which is based on the IEEE template. Knowledge of basic mathematical rules and functions is assumed.

1.2 Intended Audience and Reading Suggestions

This document is intended for

- the group designing the program (Group K),
- the group responsible for giving feedback and implementation (Group F)
- for developers, which want to further develop CAS.

Chapter 2 gives a general explanation of the product, chapters 3 to 6 add more detailed explanations.

1.3 Product Scope

The product is a Computer Algebra System (CAS), which allows users to calculate simple mathematical expressions. It is for general purpose calculations and targeted at console users which have to perform quick calculations

1.4 References

The material of the lecture “Software Engineering” can be found at *http://lec.inf.ethz.ch/se/2020/project/CAS/CAS.html* .

2. Overall Description

2.1 User Classes and Characteristics

The users of the product will be mathematicians, physicians, engineers and all kinds of scientists that might need to solve complicated calculations that cannot be solved without a computer. Those users are supposed to already know the mathematical vocabulary and they are also expected to know the way of writing down equations on a computer. They will be able to access the program over a console that allows them to write equations or other types of mathematical computations.

2.2 Operating Environment

A modern Linux-OS and C++11

2.3 User Documentation

A user can find quick help when typing “help [argument]”.
An argument can be any function (print, delete, sin). If left empty, “help” will return an overview of all available functions. (similar to the output when typing help on a Linux bash)
Some short usage examples will be provided in the man-page.

2.4 Assumptions and Dependencies

For convenience, the implementation of the parsing algorithm is already provided (see http://lec.inf.ethz.ch/se/2020/project/CAS/CAS.html).

3. External Interfaces and Requirements

3.1 User Interface

A simple Unix based console program. Upon pressing enter:

- The software parses the input,
- tries to perform the requested operation(s),
- returns the result, the requested data or an error message.

The program will accept input until the user enters “exit” or kills the program (e.g. ctrl+d on Linux).

3.2 Software Interfaces

CAS is developed in C++ and requires a modern C++ Compiler like gcc 7.4.0 or newer.
CAS runs as a simple console application.

3.3 Communications Interfaces

No communication interfaces are needed, since the program is for offline usage only.

4. System Requirements

The System must have a user interface, allowing to enter and display mathematical formulas.

4.1 Functional Requirements

4.1.1 FREQ-1: Parsing expressions

Description: The program parses given strings and performs the requested operations if possible.
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

4.1.2 FREQ-2: Basic arithmetic operations

Description: It supports the basic arithmetic operations: addition, multiplication, subtraction, division & modulo. Expressions with more than one operator are resolved correctly (precedence, from left to right).
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

4.1.3 FREQ-3: Store variables

Description: It supports entering variables and storing them by a given symbol, e.g. "a=3" stores the value 3 into a. Often used variables (e.g. pi, e) are already stored and cannot be overwritten. A variable can be used for further computations. Variables can be printed by using e.g. print(a)
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

4.1.4 FREQ-4: Delete variables

Description: The program supports deletion of variables. E.g. delete(a) deletes the value stored at a (if a was defined prior).
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

4.1.5 FREQ-5: Nested expression using parentheses

Description: Nested expressions with more than just one operation, separated with parenthesis, should be handled by the programm.
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

4.1.6 FREQ-6: Must known basic functions & constants

Description: The functions: sin(x), cos(x), tan(x), exp(x), ln(x) & pow(x, y) are implemented. Aswell constants: PI, e.
User Priority(x/5): 3/5
Technical Priority(y/5): 1/5

4.1.7 FREQ-7: Error feedback

Description: Upon entering illegal requests of any kind, the user is informed about the problem with a reasonable message.
User Priority(x/5): 3/5
Technical Priority(y/5): 1/5

4.1.8 FREQ-8: Help-Menu

Description: When entering “help [arguments]”, the user gets information according to the argument specified.
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

4.1.9 FREQ-9: Saving and loading data

Description: A user can save and load data into/from a .csv file. Saving is done with “save(filename)”, and loading with “load(filename)”.
User Priority(x/5): 2/5
Technical Priority(y/5): 3/5

4.1.10 FREQ-10: Exiting the program

Description: Upon typing “exit”, the program checks if there are stored variables. If yes, the user is asked if he wishes to save the data. If confirmed, the data is stored into a csv file.
User Priority(x/5): 2/5
Technical Priority(y/5): 2/5

5. System Scenarios

5.1 Use-case Diagrams

User steps	**System steps**
1. Starts program	
	2. Runs and requests input
3. Puts in mathematical expression	
	4. Parses users input; stores result in memory & outputs result on console; requests next input
5. Views & writes “exit”	
	6. Program stops

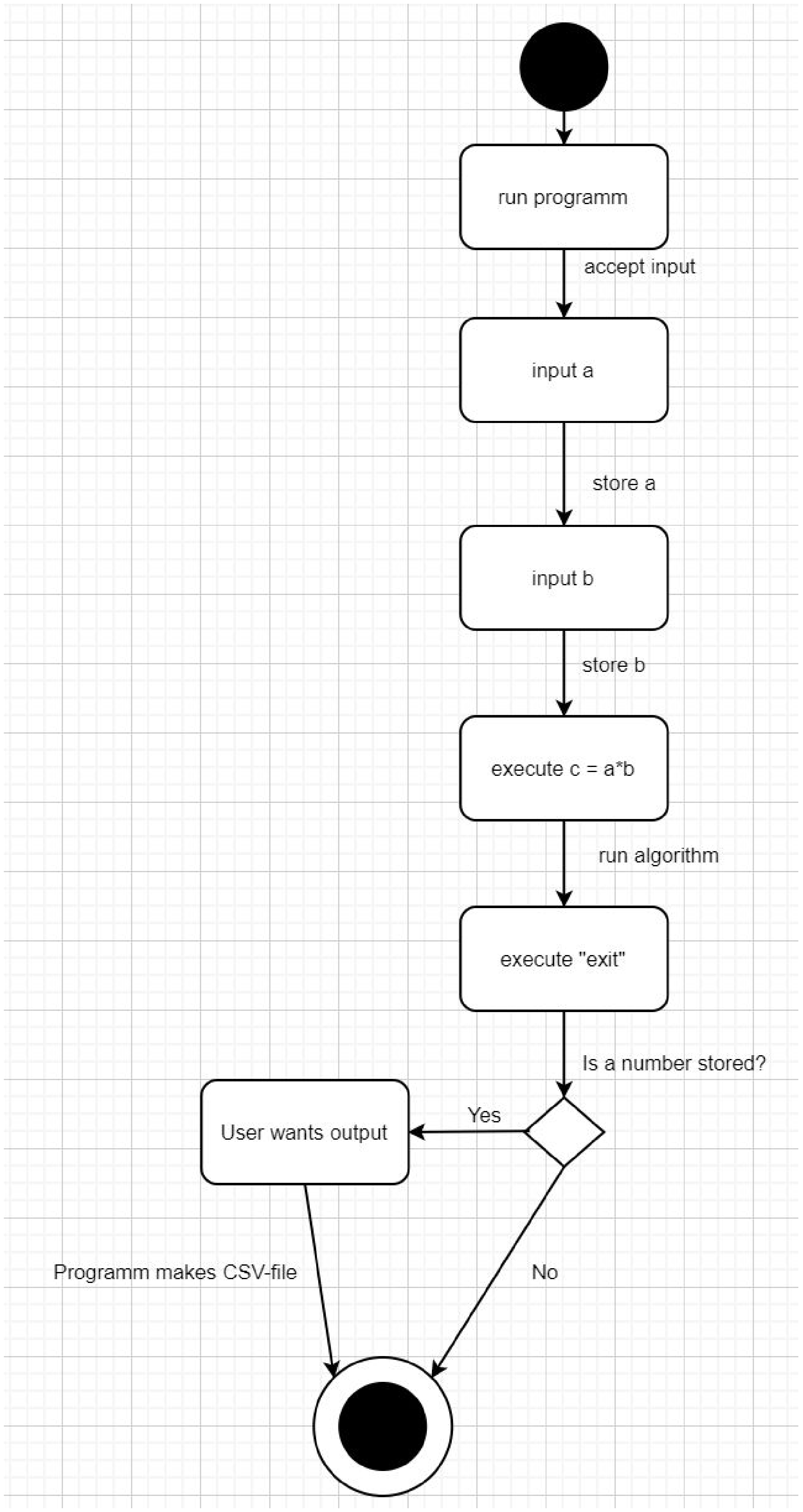
5.2 Scenarios

5.2.1 SCN-1: Store variable and multiplication

FREQ reference	1,2,3,9,10
NFREQ reference	1,2,4,5
Short Description:	The user wants to store variables and multiply them.
Activation action:	The user runs the programm.
Precondition:	The system is off.

Basic flow: Storage of 2 variables and multiplication		
Step	**User action**	**System response**
1	Starts the program.	Displays “Enter mathematical expression: ”.
2	Writes “a = 2.5” in the command line.	Stores the given input in the variable “a”, i.e. 2.5
		Displays “Enter mathematical expression: ”.
3	Writes “b = 3” in the command line.	Stores the given input in the variable “b”, i.e. 3
		Displays “Enter mathematical expression: ”.
4	Writes “c = a * b” in the command line.	Computes a*b and stores the results in the variable “c”, i.e. 7.5
		Displays “Enter mathematical expression: ”.
5	Writes “exit” in the command line.	Asks if the stored variables should be put into a CSV-file
6	Writes “yes”	Write CSV-file: variables.csv
		Displays “Enter mathematical expression: ”.
Post-condition:	The system is waiting for an input from the user, i.e. a mathematical expression. The values of a, b and c are stored and can be used for the next computations	

Scenario Diagram for SCN-1 Store variable and multiplication


run programm
accept input
input a
store a
input b
store b
execute c = a*b
run algorithm
execute "exit"
Is a number stored?
Yes
User wants output
No
Programm makes CSV-file

5.2.2 SCN-2: Nested expressions and exit

FREQ reference	1, 2, 4, 5, 10
NFREQ reference	2,4,5
Short Description:	The user wants to know the results of nested expressions with parentheses.
Activation action:	The user runs the programm.
Precondition:	The system is off.

Basic flow: Nested expressions		
Step	**User action**	**System response**
1	Starts the program.	Displays "Enter mathematical expression: ".
2	Writes "-4*(5 - 2/(3 * 2 - 3.3))" in the command line.	Calculates and displays the result over the console (because no "=" is detected). Output: "-1.7037037037…"
		Displays "Enter mathematical expression: ".
4	Sees the result and writes "exit" in the command line.	The program cannot find any stored variables and shuts down.
Post-condition:	The system is off.	

5.2.3 SCN-3: Trigonometric expression

FREQ reference	1, 6
NFREQ reference	2, 4
Short Description:	The user wants to know the result of a trigonometric expression.
Activation action:	The user writes a mathematical expression in the command line in the form of a trigonometric equation.
Precondition:	The system is already on and waiting for an input.

Basic flow: Trigonometric equation		
Step	**User action**	**System response**
1	Writes "sin(PI)" in the command line.	Calculates and displays the result over the console. Output: "0"
		Displays "Enter mathematical expression: ".
2	Sees the result and writes "exit" in the command line.	The program cannot find any stored variables and shuts down.
Post-condition:	The system is off.	

5.2.4 SCN-4: Error message and variable storage

FREQ reference	1, 3, 7
NFREQ reference	1, 2, 3, 4, 5
Short Description:	The user enters an illegal expression which leads to an error
Activation action:	The user runs the programm.
Precondition:	The system is off.

Basic flow: Error message		
Step	**User action**	**System response**
1	Starts program.	Displays “Enter mathematical expression: ”.
2	Writes “x = ln(0)” in the command line.	Displays “ERROR: cannot compute ln(0)! No operations were performed. ”
3		Displays “Enter mathematical expression: ”.
4	Writes “x = ln(1)” in the command line.	Computes “x” and stores the results in the variable “x”, i.e. 0
Post-condition:	The system is on, the value 0 is stored in x.	

6. System Constraints

6.1 Important Nonfunctional Requirements

6.1.1 NFREQ-1: Storage size

Description: At least 20 different variables can be stored at the same time. Upon reaching this limit, the system warns the user and asks to delete some variables.
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

6.1.2 NFREQ-2: Accuracy

Description: The error between the real value and the calculated value should be less than 2^-10 (~9*10^-4 single precision). Output should be rounded by up to 10 decimals.
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

6.1.3 NFREQ-3: Error handling

Description: There are no inputs which lead to a crash of the program.
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

6.1.4 NFREQ-4: Double values

Description: For all computations, C-doubles are used.
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5

6.1.5 NFREQ-5: Whitespaces in input

Description: The system should be flexible with whitespaces in the input and handle them properly.
E.g. "   1 *   15 " returns "15".
User Priority(x/5): 5/5
Technical Priority(y/5): 5/5