Using Scientific Software engineering to implement the transcendental function $y = ab^x$.

Herve Ngomseu Fotsing (40067741)

SEON 6011 – Software Engineering Processes

Project Report

Concordia University

August 05^{th}, 2022

Repository address: CLICK HERE!!.

Keywords: agile methodologies and DevOps; software engineering; versioning and testing; transcendental function.

Contents

1 Introduction **3**

1.1 *Description of the function* 3

1.2 Context of use model 4

2 Requirement Extraction **5**

2.1 Requirements elicitation 5

2.2 Requirements specification 6

2.2.1 Functional requirements 6

2.2.2 Non-functional requirements 7

2.3 Requirements verification and validation 7

2.4 Requirements management 7

3 Source code review and description of test cases **8**

3.1 Pseudocode and description of Algorithms 8

3.2 Test cases and tool used 10

3.2.1 Advantages of Eclipse 10

3.2.2 Disadvantages of Eclipse 11

4 Conclusion **12**

1 Introduction

Software engineering best practices such as agile methodologies and DevOps are increasingly incorporated in the development and implementations of software in organisations. In scientific software engineering, for instance, scientific calculators have to support functionalities including evaluating transcendental functions. In this document we present an implementation of the function ab^x. In this section we describe the function and construct a context of use for the model, section 2 expresses the requirements of the function and assumptions made. In section 3, we describe the algorithm and its subordinate functions, and reasons behind the choices. We conclude the study in section 4.

1.1 *Description of the function*

In the function ab^x we have three parameters namely: the numbers a and b, and the variable x representing the power or exponent of the base number b. Hence, to evaluate this function we need to multiple two numbers a and b raised to the power of x. a and b are real constants with range $(-\infty, +\infty)$ while x is a variable with the same range.

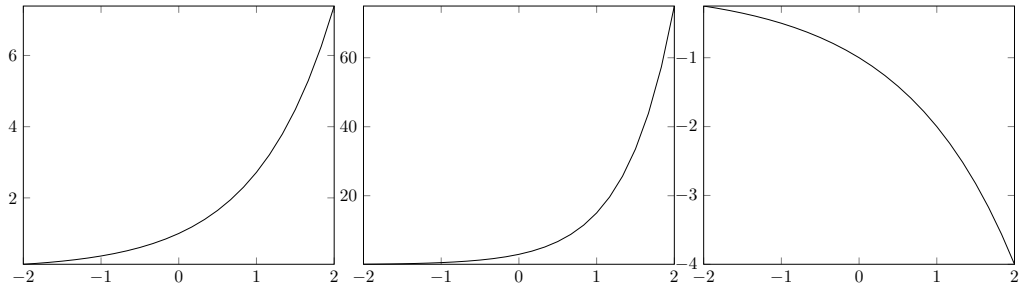

Figure 1: Illustration of examples e^x, $3(5^x)$, and -2^x respectively with domain $[-2, 2]$.

In the function $y = ab^x$, b^x represents an exponential function a is regarded as a scaling factor. The function contains a single term and also can not be simplified or reduced to an alternate form. Some examples are show in figure 1 and 2 below.

The range of y varies depending on various parameter as follows.

- When $a < 0$, $y \in (-\infty, 0)$.
- When $a > 0$, $y \in (0, +\infty)$.

- When $a = 0$, $y = 0$, if $b = 1$ then $y = a$.
- When $b = e$, $a = 1$, $y = e^x$ (The exponential function. $y \in (0, +\infty)$).

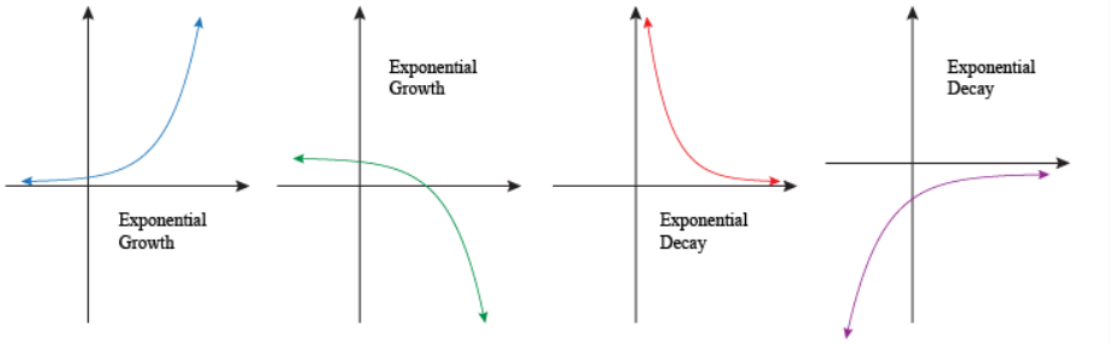

Figure 2: Typical variations of $y = ab^x$ for standard trends.

1.2 Context of use model

The calculator application as described by the function can accommodate two types of user namely. Scientists (human users) and external programs through Application Program Interface (API) calls on subordinate functions.

The main function takes as input valid double data types for the three parameters a, b and x. These values are entered by the user through the user interface and validated by the program and then passed into the main method. This method in turn calls subordinate functions to obtain the relevant results. The calculator then return the final results of calculation to the user interface. This could be a value or simply a message indicating a calculation error. This process is illustrated in figure 3 below [4].

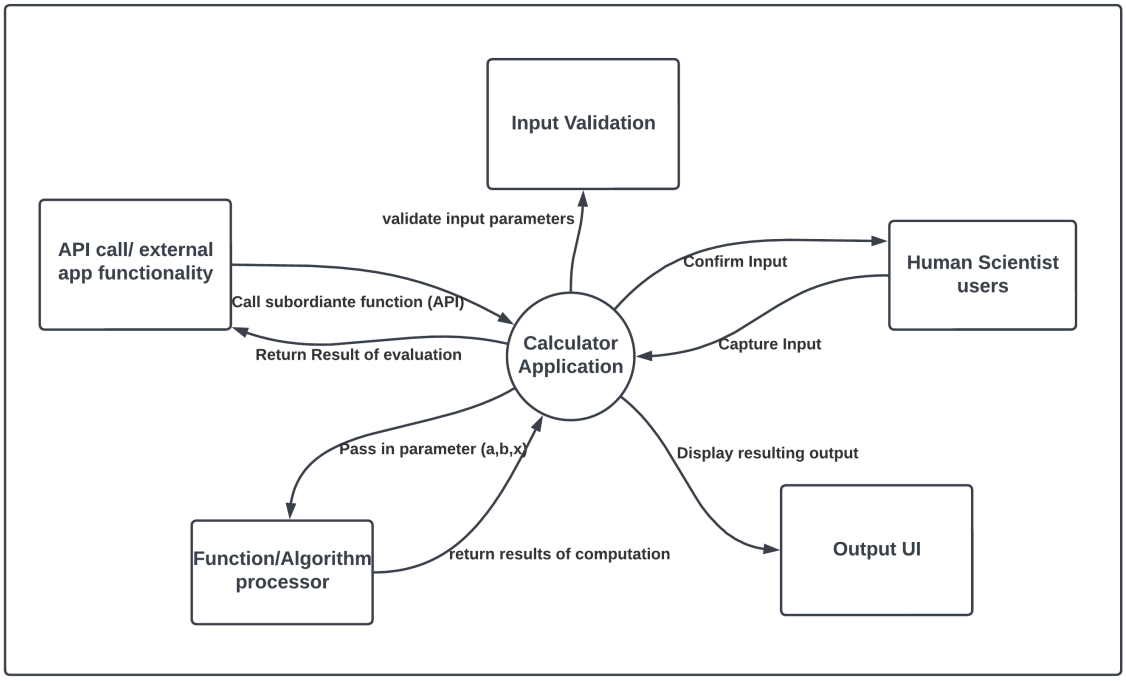

Figure 3: Context of use model for the program $y = ab^x$.

2 Requirement Extraction

In order to obtain a artefact through the software development process, it is necessary to formulate requirement of the system. These artefacts are usually the by-product produced during the development process (DevOps) include relevant designs diagrams and models use as described in the document [6]. We shall consider four main activities in the requirement extraction process as follows.

2.1 Requirements elicitation

At this stage we communicated intensively with stakeholders. In this case this was done by reading project description, clarifying requirements with the lecturer and discussing the assignment with teaching assistant and other fellow student to have more insights on the problem. The various approaches to the implementation of subordinate function were investigated and constraint were defined. The context diagram was developed and the necessary literature was conducted for various implementation. The users include human

users and APIs. Mathematical concepts and implications were considered and there was revision of boundaries and constraints of the function $y = ab^x$. A draft of the UI design was discussed too with stakeholders. At this stage there is a clear understanding of the problem.

2.2 Requirements specification

Here we elaborate requirements formally based on the previous stage of the process. All requirements including functional and non-functional requirements, and constraints are specifies here [5]. As studied in agile methodologies, these activities are carried out incrementally and iteratively as we develop the program by revising requirements. Git and GitHub was used as the version control system throughout the development process.

2.2.1 Functional requirements

- The program must be implemented in Java and the document written in Latex.
- The mathematical expressions in must be typeset in Latex and not as images.
- The artifacts must be placed in a distributed version control system.
- The algorithms must handle every boundary conditions of the domain and codomain of the function.
- The main function ($y = ab^x$) and its subordinate functions (e^x, a^x, $\ln x$, $|x|$) must be implemented from scratch.
- Unit tests (JUnits) must be created to test each corresponding subordinate functions. This is done to test that our solution corresponds to or at least nearly equates to the actual value.
- A variable parameter ϵ may be provided by the user to control the allowable error in the calculations. A default value is used if ϵ is not provided.

2.2.2 Non-functional requirements

- The program must be user friendly and well documented using software engineering best practices.
- The solution implementation must be scalable for various mathematical calculations and maintainable over time.
- The program must be portable. The algorithms and source code for the program and document (Latex) must run on any integrated development environment for the specified language (Java).
- The algorithms for subordinate functions must be designed to attain high performance in terms of both space and time complexities.

2.3 Requirements verification and validation

Verification is conducted to ensure the software implements the specific function. This is done by testing the implementation against a range of values and comparing them with the actual answers of default calculations.

Validations is carried out to ensure that the software matches various constraints and is traceable to customer requirements. This also help to avoid propagation of errors to successive stages. Unit test were developed for various constraint to satisfy the requirements of the function.

2.4 Requirements management

At this stage, we analyse, document, track, prioritise and agree on requirement while controlling communication with stakeholders. This is done through flash reviews organised by the lecturer to monitor progress thus far and make changes or validate requirements. We also produce relevant documentations and review on the implementation of the software.

3 Source code review and description of test cases

The source code of the program conforms to google coding style and the variable names comply to the camel case standard nomenclature [2]. The code also aligns with the standard programming style for Java. Throughout the program exception handling, error messages and other error handling mechanisms are used to make the program robust, it also contribute to enforcing the constraints of the mathematical expressions. A graphical user interface is used to capture user inputs and display the program output. Google's code review guidelines was use to review the implementation of the function [3].

3.1 Pseudocode and description of Algorithms

In order to build the function we consider various edge and special cases that are important in the evaluation of the function. Part of the function constitute a power function as described in algorithm 1. The algorithm was designed to be simple and to achieve optimal performance. It constitutes a foundation for function.

Algorithm 1 Algorithm for power(x, n)

Require: $n \geq 0$

Ensure: $y = x^n$

```
y ← 1
X ← x
N ← n
while N ≠ 0 do
    if N is even then
        X ← X × X
        N ← N/2                          ▷ setting N to its half
    else if N is odd then
        y ← y × X
        N ← N − 1
    end if
end while
```

The algorithms 2 and 3 are implemented as subordinate functions to the main calcula-

tion. The function $\ln x$ utilises the Taylor series expansion as a scratch solution to compute results as in algorithm 2.

Algorithm 2 Algorithm for $\ln x$

Require: $x > 0$, $nterms$

Ensure: $y = \ln x$

```
result ← 0.0
while x > 2 do
    result ← result + ln 2
    x ← x/2
end while
x ← x − 1
factor ← 1
res ← 0.0
for i ← 1 to nterms do
    res ← res + factor ∗ power(x, i)/i        ▷ External call to Algorithm 1
    factor ← factor − factor
end for
result ← result + res
```

The function e^x uses a default value of $e = 2.7182818284590451$ and the power function to constitute the evaluation as shown in algorithm 3.

Algorithm 3 Algorithm for e^x

Require: $x \in \mathbb{R}$, $nterms$ (Number of terms)

Ensure: $y = e^x$

```
temp ← 1
result ← 1.0
for i ← 1 to nterms do                ▷ Taylor's Series Expansion of ln x
    temp ← temp ∗ i
    result ← results + power(x, i)/temp
end for
```

The absolute value function is implemented as a utility function as part of the testing

process to compare various test results with the allowable error, ϵ.

Algorithm 4 Algorithm for $|x|$

Require: $x \in \mathbb{R}$

Ensure: $y = |x|$

  if x is negative **then**

    $x \leftarrow -x$

  end if

Algorithm 5 Algorithm for ab^x

Require: $x \in \mathbb{R}$, a, x

Ensure: $y = ab^x$

  $result \leftarrow a * power(b, x)$ ▷ External call to Algorithm 1

3.2 Test cases and tool used

The program is implemented using Eclipse IDE for developers (version 2021-09) [1]. The program is illustrated below. The Latex code was written in TeXstudio (version 4.0.0). Java Developer Tools (JDT) such as JUnit (JUnit 5 for Eclipse Oxygen was used) to build test cases. The coverage was computed with JaCoCo library, a plug-in in Eclipse.

Both the Javadoc and test case coverage were exported and stored into the source folder of the program. The graphical user interface was implemented with the *javax.swing* library. The test cases comprise individual tests for each subordinate function using chosen values as shown in figure 4.

3.2.1 Advantages of Eclipse

- Eclipse is a free and open source IDE which supports multiple languages and is a especially suited for Java, it is one of the most popular IDE's for Java.
- It supports a wide range of plug-ins, allowing users to make their own plug-in development environment (PDE).
- Eclipse provides modelling support for projects including Unified Modelling Language (UML).

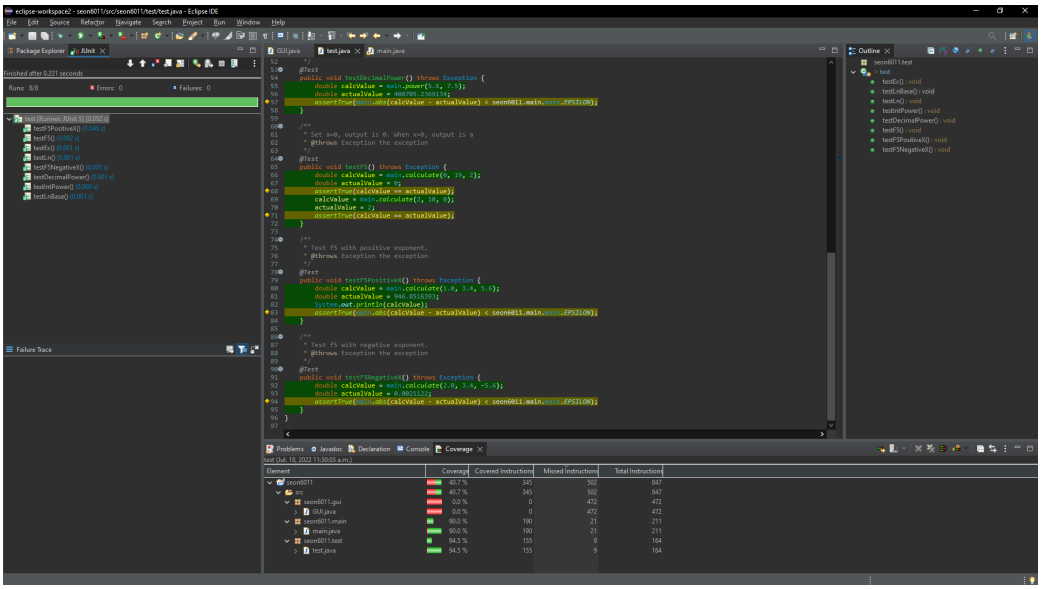

Figure 4: Test cases showed in the debugger.

3.2.2 Disadvantages of Eclipse

- Eclipse tends to be significantly slow to run compared with other IDEs such as NetBeans due to insufficient memory allocation.

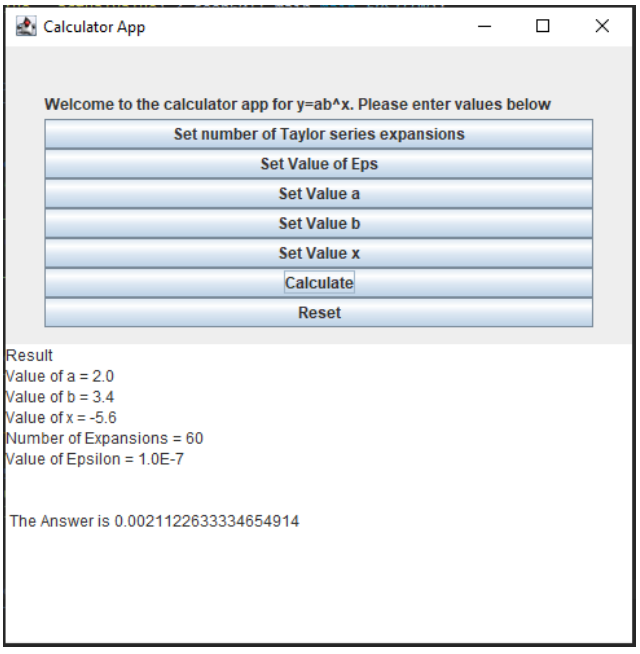

Figure 5: Graphical user interface for Calculator App $y = ab^x$.

4 Conclusion

In conclusion, we described, implemented and documented the scientific calculation of the function $y = ab^x$ using software engineering best practices. The program was built conceptually and tested in Java programming language, we designed a also designed a context of use that fits with the implementation.

References

[1] eclipse. Eclipse documentation. https://help.eclipse.org/latest/index.jsp, 2022. Accessed: July 18, 2022.

[2] Google. Google Software engineering best practices. https://google.github.io/eng-practices/, 2022. Accessed: July 18, 2022.

[3] Google. How to do a code review, Google. https://google.github.io/eng-practices/review/reviewer/, 2022. Accessed: July 18, 2022.

[4] P. Johnston. Software System Context Model, 2022. Accessed: July 18, 2022.

[5] Y. Li, E. Guzman, K. Tsiamoura, F. Schneider, and B. Bruegge. Automated requirements extraction for scientific software. *Procedia Computer Science*, 51:582–591, 2015.

[6] A. Ravid and D. M. Berry. A method for extracting and stating software requirements that a user interface prototype contains. *Requirements Engineering*, 5(4):225–241, 2000.