SOEN 6011 Project Documentation

https://github.com/hbcbs110/SOEN6011-project

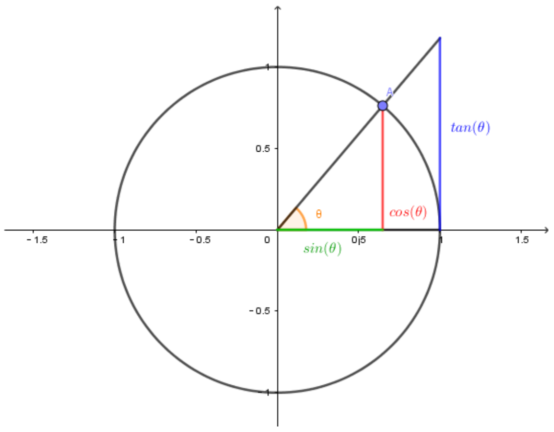
Shuo Gao
40203484
Concordia University

Summer 2022

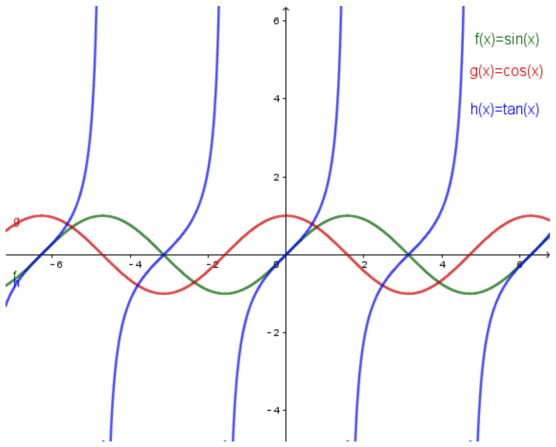
Contents

1 Description

Function used: tan(x) The function tan(x), which is called tangent function, is one of the trigonometric functions (also called circular functions), which are real functions relating an angle of a right-angled triangle to the ratio of two side length. The trigonometric functions are widely used in all sciences and technologies. For an acute angle θ in a right triangle, the tangent function can be defined as: $tan(\theta) = \frac{opposit}{adjacent}$.[1] Here is a graph of tan(x) below in Figure 1:



(a) circular functions



(b) graph of tan(x)

Figure 1: Graph of tangent function tan(x)

1.1 Domain and co-domain

1. **Domain**: $\{x \in \mathbb{R} | x \neq k\pi + \frac{\pi}{2}, k \in \mathbb{Z}\}$
2. **Co-domain**: $(-\infty, +\infty)$ All real number $\mathbb{R}$

1.2 Characteristics

1. $tan(x) = \frac{sin(x)}{cos(x)}$
2. period: π
3. vertical asymptotes: $x = \frac{\pi}{2} + k\pi, k \in \mathbb{Z}$
4. $tan(a \pm b) = \frac{tan(a) \pm tan(b)}{1 \mp tan(a)tan(b)}$
5. $tan(-x) = -tan(x)$

2 Context of use model

The model below is based on the guideline in IEEE Guide for Information Technology System Definition Concept of Operation (ConOps) Document.[2]

1. User: A user who is going to calculate the value of tan(x) with the input x using a calculator.
2. Task: Calculate the output of tan(x) given the input x and who the result on the screen to the user.
3. Environment:
   - Technical environment: Power supply of the calculator. Runtime environment.
   - Non-technical environment: Location of the user while using the calculator.

3 Requirement

3.1 Introduction

Requirements below are written based on the the styles and guidelines given in the ISO/IEC/IEEE 29148 Standard, section 5.2.4 to 5.2.8.[3]

3.2 Requirements Set

1. Requirement 1
   - Identification: FR1
   - Version Number: 1.0
   - Owner: Shuo Gao
   - Stakeholder Priority: High
   - Risk: Low
   - Description: The user shall make sure that he/she inputs a real number, or a valid double number. The real number may contains numbers from 0 to 9, period ".", letter e, and plus or minus sign "+" and "-".
   - Rationale: The function tan(x) only supports real numbers as input. To avoid error of calculating
   - Difficulty: Easy
   - Type: Functional Requirement
2. Requirement 2
   - Identification: FR2
   - Version Number: 1.0
   - Owner: Shuo Gao
   - Stakeholder Priority: High
   - Risk: Low
   - Description: The input of the tan(x) fanction should be radians, not degrees.
   - Rationale: For the convenience of calculating and implementation of the algorithm, the argument of a trigonometric function is defined as the radians of angle. There for the function tan(x) takes radians as input.
   - Difficulty: Easy
   - Type: Functional Requirement
3. Requirement 3
   - Identification: FR3
   - Version Number: 1.0
   - Owner: Shuo Gao
   - Stakeholder Priority: High
   - Risk: Low
   - Description: Each time the tan(x) function handles only one input and generates one output.
   - Rationale: For the convenience of calculating, the program processes one input at a time.
   - Difficulty: Easy
   - Type: Functional Requirement
4. Requirement 4
   - Identification: FR4
   - Version Number: 1.0
   - Owner: Shuo Gao
   - Stakeholder Priority: High

- Risk: Low
- Description: The function tan(x) should output a correct result, at least 8 to 10 digits are precise.
- Rationale: As a calculator the output should be correct and the precision should be in a practical range.
- Difficulty: Easy
- Type: Functional Requirement

5. Requirement 5
   - Identification: NFR1
   - Version Number: 1.0
   - Owner: Shuo Gao
   - Stakeholder Priority: High
   - Risk: Low
   - Description: The function should return the result shortly.
   - Rationale: Calculator needs good performance.
   - Difficulty: Easy
   - Type: Non-Functional Requirement
6. Requirement 6
   - Identification: NFR2
   - Version Number: 1.0
   - Owner: Shuo Gao
   - Stakeholder Priority: High
   - Risk: Low
   - Description: The program must run in the environment where JRE(Java Runtime Environment) is installed.
   - Rationale: The program is based on Java programming language and it requires JRE to run. Thus JRE have to be installed in the environment.
   - Difficulty: Easy
   - Type: Non-Functional Requirement
7. Requirement 7
   - Identification: NFR3
   - Version Number: 1.0
   - Owner: Shuo Gao
   - Stakeholder Priority: High
   - Risk: Low
   - Description: The calculator should not crash when there is an exception during input or calculating.
   - Rationale: Reliability.
   - Difficulty: Easy
   - Type: Non-Functional Requirement

3.3 Assumptions

1. The input number x should be a real number in radian.
2. The input should not be a string or an expression.
3. If the input is not a valid number, an exception will be thrown and ask the user to input again.
4. The output of the function will be a real number in $(-\infty, +\infty)$.

4 Algorithm

4.1 Mind Map

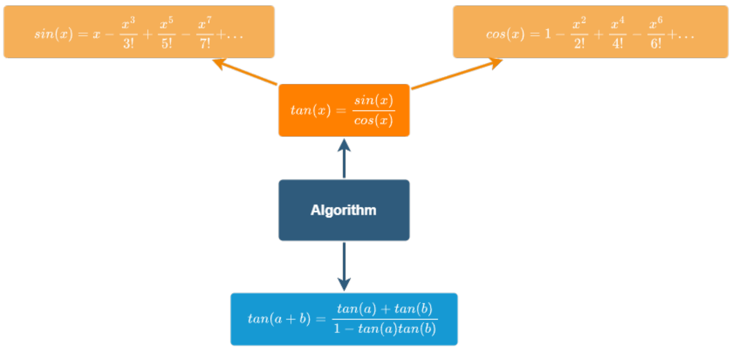

$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$
$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$
$tan(x) = \frac{sin(x)}{cos(x)}$
Algorithm
$tan(a+b) = \frac{tan(a)+tan(b)}{1-tan(a)tan(b)}$

Figure 2: Mind map

4.2 Algorithm 1

By definition, tan(x) can be calculated with formula $tan(x) = \frac{sin(x)}{cos(x)}$. So we can calculate sin(x) and cos(x) first then tan(x). They can be calculated using Taylor series shown below[1]:

$$sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

$$cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Algorithm 1

```
Require: $x \in \mathbb{R}$
Ensure: $y = tan(x)$
  $x \leftarrow input$
  $y \leftarrow sin(x)/cos(x)$

  function SIN($x$)
      $sign \leftarrow f \leftarrow 1$                ▷ Sign and factorial factor of each term
      $a \leftarrow x$                                ▷ Term of Taylor series without factor
      $s \leftarrow 0$                                ▷ Result of sin(x)
      for $i \leftarrow 1$ to 100 do                  ▷ Process Taylor series
          $s \leftarrow s + sign \times a/f$
          $a \leftarrow a \times x \times x$
          $sign \leftarrow -sign$
          $f \leftarrow f \times (2 \times i + 2) \times (2 \times i + 3)$
      end for
      return $s$
  end function

  function COS($x$)
      $sign \leftarrow f \leftarrow 1$                ▷ Sign and factorial factor of each term
      $a \leftarrow 1$                                ▷ Term of Taylor series without factor
      $c \leftarrow 0$                                ▷ Result of cos(x)
      for $i \leftarrow 1$ to 100 do                  ▷ Process Taylor series
          $c \leftarrow c + sign \times a/f$
          $a \leftarrow a \times x \times x$
          $sign \leftarrow -sign$
          $f \leftarrow f \times (2 \times i + 1) \times (2 \times i + 2)$
      end for
      return $c$
  end function
```

4.3 Algorithm 2

The function tan(x) has a characteristic below:

$$tan(a+b) = \frac{tan(a) + tan(b)}{1 - tan(a)tan(b)}$$

Based on this characteristic, we can iteratively add small numbers into the angle until it reaches x. Assuming x is greater than zero, if not let $x = -x$ using the characteristic that $tan(-x) = -tan(x)$. Then the tan(x) can be calculated as below:

$$x_0 = 0, x_{i+1} = x_i + \Delta_i, x_n = x$$

$$tan(x_{i+1}) = \frac{tan(x_i) + tan(\Delta_i)}{1 - tan(x_i)tan(\Delta_i)}$$

The Δ_i and $tan(\Delta_i)$ are pre-defined constant values and $\Delta_{i+1} \leq \Delta_i < x$ for all i. All we need to do is iteratively calculate the value until $x_i = x_n = x$.

Algorithm 2

```
Require: $x \in \mathbb{R}$
Ensure: $y = tan(x)$
  $x \leftarrow input$
  $x_0 = 0$
  $y \leftarrow 0$
  $i \leftarrow 0$
  while $x_0 < x$ and $i < length(\Delta)$ do                    ▷ Iterate
      $y \leftarrow (y + tan(\Delta_i))/(1 - y \times tan(\Delta_i))$
      $x_0 \leftarrow x_0 + \Delta_i$                              ▷ Accumulate $x_i$
      $i \leftarrow i + 1$
  end while
```

4.4 Technical Reasons of Selecting Each Algorithm

4.4.1 Algorithm 1

- **Advantages**: Result is more precise.
- **Disadvantages**: Longer than Algorithm 2. Factorial may overflow if too many terms in Taylor series are used.

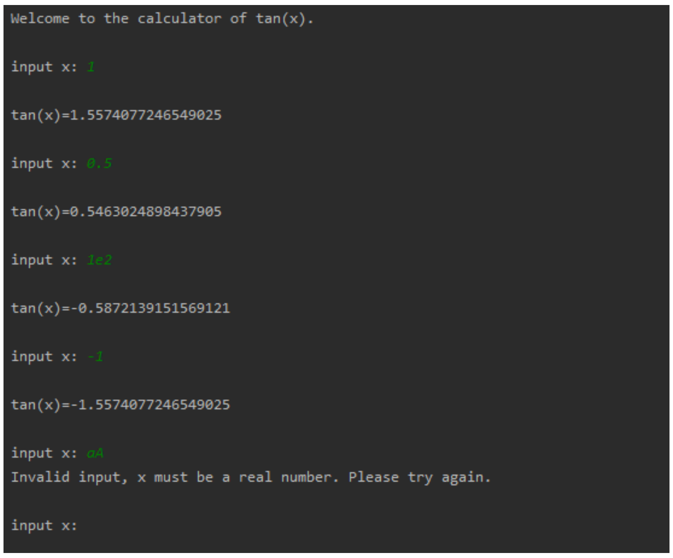
4.4.2 Algorithm 2

- **Advantages**: As an iterative algorithm it is easier to implement. Will not overflow during calculating.
- **Disadvantages**: Not as precise as Algorithm 1. The Δ_i and $tan(\Delta_i)$ need to be defined in advance.

5 Implementation and Debugging

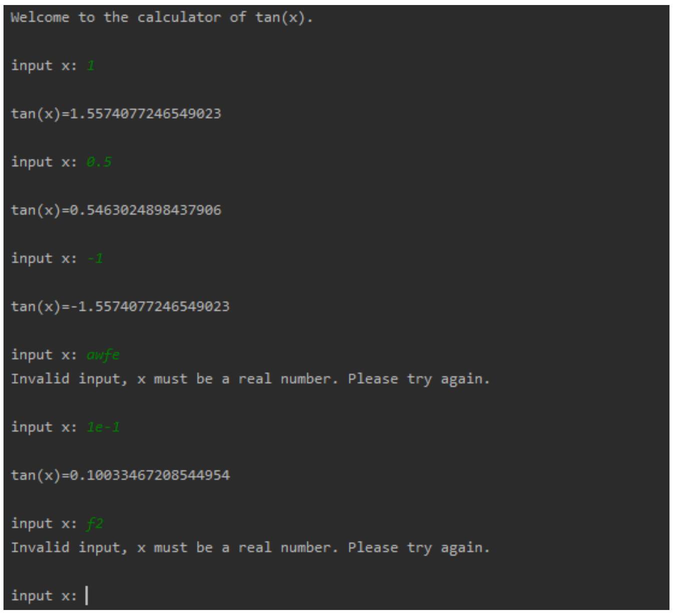
5.1 Implementation

The function is implemented from scrach in Java. No built-in library like Math provided by Java is used. The source code implementing Algorithm 1 and Algorithm 2 are Function.java and Function2.java. They are compiled into Function.jar and Function2.jar. The user interface is textual and error handling is supported. Below are some screenshots of the interface and error messages.



```
Welcome to the calculator of tan(x).

input x: 1

tan(x)=1.5574077246549025

input x: 0.5

tan(x)=0.5463024898437905

input x: 1e2

tan(x)=-0.5872139151569121

input x: -1

tan(x)=-1.5574077246549025

input x: aA
Invalid input, x must be a real number. Please try again.

input x:
```

(a) Output of Function.java



```
Welcome to the calculator of tan(x).

input x: 1

tan(x)=1.5574077246549023

input x: 0.5

tan(x)=0.5463024898437906

input x: -1

tan(x)=-1.5574077246549023

input x: awfe
Invalid input, x must be a real number. Please try again.

input x: 1e-1

tan(x)=0.10033467208544954

input x: f2
Invalid input, x must be a real number. Please try again.

input x:
```

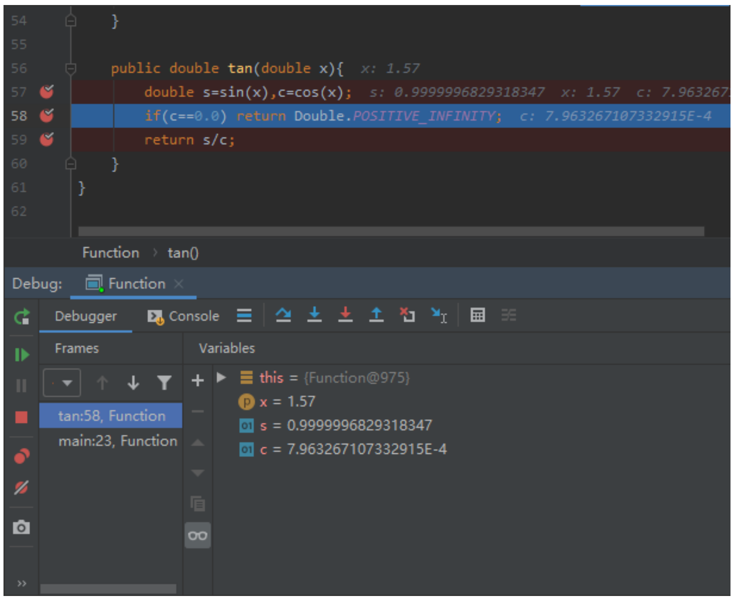
(b) Output of Function2.java

Figure 3: Results of the program

5.2 Debugging

In order to debug efficiently, the built-in debugger of the IDE IntelliJ IDEA is used. The debugger is a powerful tool. It allows users to monitor and interfere with the code and quickly catch the bugs in their code. As a built-in tool, it works well with IDE itself. Users can easily access it and debug with it during coding. The IDE provided multiple configurations of it and each process such as resume, step over and step into is easy to access. To monitor a variable, users can easily type name of the variable or just use the editor of IDE to select it. Breakpoint can be added by clicking the left side of the editor and can also be modified during debugging. The figure 4 shows the screen shot of the debugger.

- **Advantages**: As a built-in debugger it is integrated well with the IDE. It has GUI and easy to process. It provides real time feed back.
- **Disadvantages**: Sometimes it is slow when monitoring too many variables or variables which take large space. It is not so powerful when monitoring multi-thread or multi-process program.



```
    }

    public double tan(double x){  x: 1.57
        double s=sin(x),c=cos(x);  s: 0.9999996829318347  x: 1.57  c: 7.963267
        if(c==0.0) return Double.POSITIVE_INFINITY;  c: 7.963267107332915E-4
        return s/c;
    }
}
```

```
tan:58, Function
main:23, Function

this = {Function@975}
x = 1.57
s = 0.9999996829318347
c = 7.963267107332915E-4
```

Figure 4: Debugger in Intellij IDEA

5.3 Quality

5.3.1 Source code

Each algorithm of the code only contains one source code file and it does not require outer data to run. It only requires JRE (Java Runtime Environment) to run. During runtime, the function iteratively calculates each term of a series and does not store the previous data. Therefore, the code is space-efficient and portable. The source code is short and well organized with functions. Each functions take charge of one calculating task and can be easily debugged. Thus the code is maintainable.

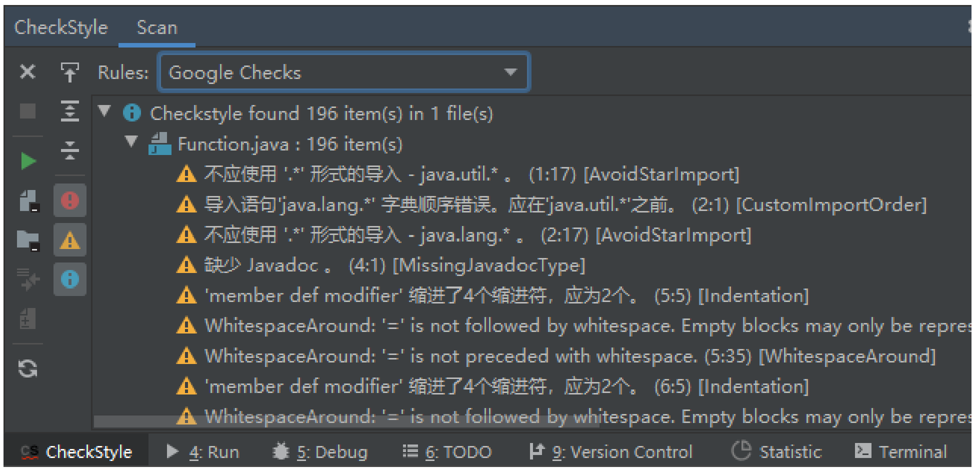
5.3.2 Program

After debugging and testing the source code, the program is confirmed correct under a certain error limit. The program might crush when user inputs a string which is not a valid real number or a double number, or is a real number that causes divided by zero error, and this situation can be avoided by error handling. When there is an invalid input, an error message is thrown telling the user to try again. This makes the program robust. To get the correct answer efficiently, the algorithms are well designed. In both algorithm 1 and algorithm 2, the loops will only run hundreds of times in total. The result can be calculated almost instantly with well precision. Users can quickly get the correct answer which shows the program time-efficient and usable.

5.3.3 Pragmatic Quality

In order to check and maintain the pragmatic quality of the Java source code, the tool **Checkstyle** is used. It is a tool that helps programmers write Java code that adheres to a coding standard. It can check many aspects of the source code, find class design problems, method design problems, and also be able to check layout and formatting issues. In IDE IntelliJ IDEA Checkstyle is installed as a plugin and used within the IDE shown in figure 5. It checks the source code following the specific rule standard.

- **Advantages**: Can find violations of code conventions. The rules can be customized.
- **Disadvantages**: It only check coding conventions but can't find bugs. Some of the information may not be necessary.



```
Rules: Google Checks
Checkstyle found 196 item(s) in 1 file(s)
  Function.java : 196 item(s)
    不应使用 '.*' 形式的导入 - java.util.* 。 (1:17) [AvoidStarImport]
    导入语句'java.lang.*' 字典顺序错误。应在'java.util.*'之前。 (2:1) [CustomImportOrder]
    不应使用 '.*' 形式的导入 - java.lang.* 。 (2:17) [AvoidStarImport]
    缺少 Javadoc 。 (4:1) [MissingJavadocType]
    'member def modifier' 缩进了4个缩进符，应为2个。 (5:5) [Indentation]
    WhitespaceAround: '=' is not followed by whitespace. Empty blocks may only be repres
    WhitespaceAround: '=' is not preceded with whitespace. (5:35) [WhitespaceAround]
    'member def modifier' 缩进了4个缩进符，应为2个。 (6:5) [Indentation]
    WhitespaceAround: '=' is not followed by whitespace. Empty blocks may only be repres
```

Figure 5: Checkstyle

6 Unit Test

The unit test is performed with JUnit. Here are the test cases below.

1. Test Case 1
   - input: 0.0
   - output of Algorithm 1: 0.0
   - output of Algorithm 2: 0.0
   - expected output: 0.0
   - result: Pass
2. Test Case 2
   - input: 1.0
   - output of Algorithm 1: 1.5574077246549025
   - output of Algorithm 2: 1.5574077246549023
   - expected output: 1.55740772465490223
   - result: Pass
3. Test Case 3
   - input: -1.0
   - output of Algorithm 1: -1.5574077246549025
   - output of Algorithm 2: -1.5574077246549023
   - expected output: -1.55740772465490223
   - result: Pass
4. Test Case 4
   - input: 0.5
   - output of Algorithm 1: 0.5463024898437905
   - output of Algorithm 2: 0.5463024898437906
   - expected output: 0.5463024898437905
   - result: Pass
5. Test Case 5
   - input: 1.5
   - output of Algorithm 1: 14.101419947171722
   - output of Algorithm 2: 14.10141994717149
   - expected output: 14.101419947171719
   - result: Pass
6. Test Case 6
   - input: 1.57
   - output of Algorithm 1: 1255.76559150
   - output of Algorithm 2: 1255.76559150
   - expected output: 1255.76559150
   - result: Pass
7. Test Case 7
   - input: -1.5
   - output of Algorithm 1: -14.101419947171722
   - output of Algorithm 2: -14.10141994717149
   - expected output: -14.101419947171719

- result: Pass

8. Test Case 8
   - input: 1e2
   - output of Algorithm 1: -0.5872139151569121
   - output of Algorithm 2: -0.5872139151569177
   - expected output: -0.5872139151569291
   - result: Pass
9. Test Case 9
   - input: 3.14159265358979323846264
   - output of Algorithm 1: 0.0
   - output of Algorithm 2: 0.0
   - expected output: 0.0
   - result: Pass
10. Test Case 10
    - input: 2.5e4
    - output of Algorithm 1: -1.0180399498523853
    - output of Algorithm 2: -1.0180399498567576
    - expected output: -1.0180399498575736
    - result: Pass
11. Test Case 11
    - input: 1e6
    - output of Algorithm 1: -0.37362445390517723
    - output of Algorithm 2: -0.3736244538993233
    - expected output: -0.3736244539875990
    - result: Pass

References

[1] Wikipedia contributors. *Trigonometric functions — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Trigonometric_functions&oldid=1101002187`. [Online; accessed 4-August-2022]. 2022.

[2] *IEEE Std 1362-1998: IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document*. IEEE., 1998. URL: `https://books.google.ca/books?id=Y3RCzAEACAAJ`.

[3] "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering". In: *ISO/IEC/IEEE 29148:2018(E)* (2018), pp. 1–104. DOI: `10.1109/IEEESTD.2018.8559686`.