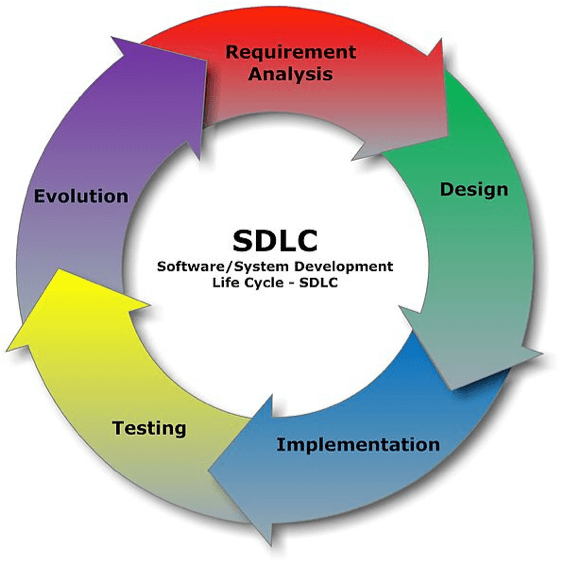
**Sample ChatGPT Thread: Software Engineering**

In this exercise we will try to use ChatGPT to help us perform some software engineering tasks. The example software we aim to build using ChatGPT is a calculator app in Python. Before involving ChatGPT in the process of developing the calculator app, let’s look at the standard software development life cycle:



In this particular exercise, we will let ChatGPT do 4 out of the 5 stages of SDLC: Requirement Analysis, Design, Implementation, and Testing. Let’s get started.

**Key:**

* **Query to ChatGPT**
* **ChatGPT’s Response**
* **Analysis of Response**

**Query to ChatGPT:**

I want to develop a simple calculator in Python. Can you generate a formal requirements elicitation for the calculator software?

**Response:**

Certainly! Here's a formal requirements elicitation for the calculator software:

**Title: Calculator Software**

**1. Functional Requirements:**

1.1 The calculator software shall provide support for basic arithmetic operations, including addition, subtraction, multiplication, and division.

1.2 The software shall allow the user to input two numbers for performing calculations.

1.3 The software shall provide an option for the user to select the desired arithmetic operation.

1.4 The software shall perform the selected operation on the input numbers and display the result to the user.

1.5 In case of division, the software shall handle the scenario where the divisor is zero and display an appropriate error message.

**2. User Interface Requirements:**

2.1 The software shall present a user-friendly interface to facilitate input and display of information.

2.2 The user interface shall provide clear instructions and options for the user to select the desired operation and enter input numbers.

2.3 The result of the calculation shall be displayed prominently and clearly to the user.

**3. Input Requirements:**

3.1 The software shall accept numerical input for the two numbers.

3.2 The software shall validate user input to ensure it is numeric and handle input errors gracefully.

3.3 The software shall accept the user's choice of operation as an input and validate it against the available options.

**4. Output Requirements:**

4.1 The software shall display the result of the calculation to the user with appropriate formatting.

4.2 In case of division by zero, the software shall display an error message indicating the impossibility of the operation.

**5. Error Handling Requirements:**

5.1 The software shall handle input errors, such as non-numeric input or invalid operation choices, and provide informative error messages to the user.

5.2 The software shall handle division by zero errors and display an error message indicating the impossibility of the operation.

**6. Performance Requirements:**

6.1 The software shall execute arithmetic calculations efficiently to provide a responsive user experience.

6.2 The software shall handle calculations involving large numbers or decimal points accurately and efficiently.

6.3 The software shall respond to user inputs in a timely manner, minimizing any noticeable delay.

**7. Usability Requirements:**

7.1 The software shall have a simple and intuitive user interface to enable easy usage by users with varying levels of technical expertise.

7.2 The user interface shall provide clear instructions and feedback to guide the user throughout the interaction.

**8. Compatibility Requirements:**

8.1 The software shall be compatible with commonly used operating systems, such as Windows, macOS, and Linux.

8.2 The software shall function correctly on different hardware configurations and screen resolutions.

**9. Security Requirements:**

9.1 The software shall not store or transmit any user data or personally identifiable information.

9.2 The software shall be free from vulnerabilities and designed to resist common security threats, such as code injection or unauthorized access.

**10. Maintenance and Support Requirements:**

10.1 The software shall be maintainable and modular, facilitating future updates or enhancements.

10.2 The software shall be accompanied by appropriate documentation, including installation instructions, usage guidelines, and troubleshooting information.

10.3 The software development team shall provide technical support and address user queries or issues promptly.

Please note that these requirements are a starting point and can be refined or expanded based on specific project needs and constraints.

***Analysis of the response:***

*ChatGPT has done a great job in coming up with the formal requirements for the calculator app. The requirements look clear and concise.*

**Next Query to ChatGPT:**

Can you show me an object oriented design for the calculator program above?

**Response:**

Certainly! Here's an object-oriented design for the calculator program:

class Calculator:

def \_\_init\_\_(self):

self.num1 = None

self.num2 = None

self.operation = None

def set\_numbers(self, num1, num2):

self.num1 = num1

self.num2 = num2

def set\_operation(self, operation):

self.operation = operation

def calculate(self):

if self.operation == '1':

return self.num1 + self.num2

elif self.operation == '2':

return self.num1 - self.num2

elif self.operation == '3':

return self.num1 \* self.num2

elif self.operation == '4':

if self.num2 != 0:

return self.num1 / self.num2

else:

return None

class CalculatorApp:

def \_\_init\_\_(self):

self.calculator = Calculator()

def display\_menu(self):

print("Select operation:")

print("1. Add")

print("2. Subtract")

print("3. Multiply")

print("4. Divide")

def get\_user\_choice(self):

choice = input("Enter choice (1, 2, 3, or 4): ")

return choice

def get\_user\_numbers(self):

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

return num1, num2

def perform\_operation(self):

self.calculator.set\_numbers(num1, num2)

self.calculator.set\_operation(choice)

return self.calculator.calculate()

def display\_result(self, result):

if result is not None:

print("Result:", result)

else:

print("Error: Cannot divide by zero")

def run(self):

self.display\_menu()

choice = self.get\_user\_choice()

num1, num2 = self.get\_user\_numbers()

result = self.perform\_operation()

self.display\_result(result)

# Create an instance of the CalculatorApp class and run the calculator

calculator\_app = CalculatorApp()

calculator\_app.run()

In this design, we have two classes: Calculator and CalculatorApp.

The Calculator class encapsulates the state and behavior related to the calculator's operations. It has methods to set the input numbers and operation, perform the calculation, and return the result. The Calculator class handles the core arithmetic operations.

The CalculatorApp class represents the user interface and interaction with the calculator. It has methods to display the menu, get the user's choice and input numbers, perform the operation using the Calculator class, and display the result to the user.

This design separates the responsibilities between the Calculator class and the CalculatorApp class, promoting modularity and separation of concerns. The Calculator class focuses on calculations, while the CalculatorApp class handles the user interface and flow of the calculator application.

Please note that this is a simplified object-oriented design, and in a real-world scenario, additional considerations such as error handling, input validation, and user feedback can be incorporated based on the project's requirements and complexity.

***Analysis of the response:***

*ChatGPT has come up with an object oriented design for the app. It has also implemented the solution in Python. However, paying close attention to the solution immediately shows us that there is a gap between the formal requirements defined by ChatGPT and its implementation. For example, ChatGPT never implemented one of the very basic requirements of input error handling (requirement 5.1), although it did implement the error handling of zero-division (requirement 5.2).*

**Last Query to ChatGPT:**

Can you generate test cases for the app?

**Response:**

Certainly! Here are some test cases you can use to verify the functionality of the calculator app:

Test Case 1: Addition

* Choice: 1
* First number: 5
* Second number: 3
* Expected output: Result: 8

Test Case 2: Subtraction

* Choice: 2
* First number: 10
* Second number: 7
* Expected output: Result: 3

Test Case 3: Multiplication

* Choice: 3
* First number: 4.5
* Second number: 2
* Expected output: Result: 9.0

Test Case 4: Division (non-zero divisor)

* Choice: 4
* First number: 15
* Second number: 3
* Expected output: Result: 5.0

Test Case 5: Division by zero

* Choice: 4
* First number: 8
* Second number: 0
* Expected output: Error: Cannot divide by zero

Test Case 6: Invalid choice

* Choice: 5
* Expected output: Invalid choice

Feel free to use these test cases or add your own to validate the functionality of the calculator app.

***Analysis of the response:***

*ChatGPT has come up with multiple test cases based on the implementation. Here also, we do not see any test cases to check for the validity of the input data. It is most likely happening due to the fact that in the implementation there is no handling of input data error although input data error handling was part of the requirements. This shows that the inconsistency between the requirements and the implementation actually propagates through the entire process from one stage to the next.*

***Final remark: Why are we seeing what we are seeing?***

ChatGPT is a machine learning model that was trained on a huge volume of data. What it is generating is not necessarily facts or correct. Its solutions are very likely to be incomplete. The reason why I chose a simple calculator app for this exercise is that there are hundreds of examples online of implementations of a simple calculator in Python, which ChatGPT has likely learned from. However, hardly any of these examples are coming from real life calculator software. Real-life software development practices such as proper Error Handling Requirements, Security Requirements, Compatibility Requirements etc. are hardly implemented in academic settings. Therefore, ChatGPT fails to implement those requirements, although these were part of the requirements elicitation which ChatGPT itself came up with in the first place.