# Quantifying Cascading Consequences with SOLID Software Design

CasConUQ development with good SWE practices

Roberto Marrero Ortiz
Mentor - Ph.D. Colin Ponce

July 27, 2023

LLNL-PRES-XXXXXX
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

## Self – Introduction

- Software Engineering Undergrad Student at University of Puerto Rico Mayaguez.
- Summer experience at LLNL:
  - Inclusive and professional work environment.
  - Many learning opportunities through other researchers and lab activities.
  - Great focus on networking and working in a collaborative manner and job enjoyment.

## CasConUQ & Software Development

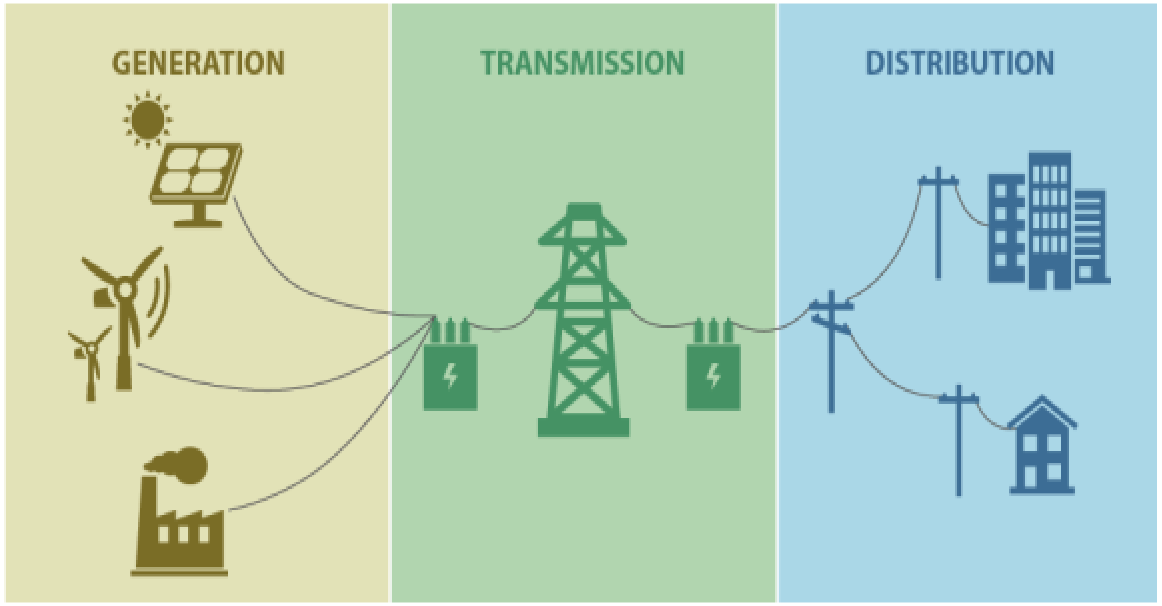
### Presentation Overview

- What is the CasCon Simulator and its connection to CasConUQ?
- What is the purpose of CasConUQ?
- My Software Engineering Approach when Developing the Program.
- Main SOLID principles I follow to help program good and rot free software:
  - Open Closed Principle
  - Interface Segregation Principle
- Future Work & Experience

## Project Overview

CasConUQ

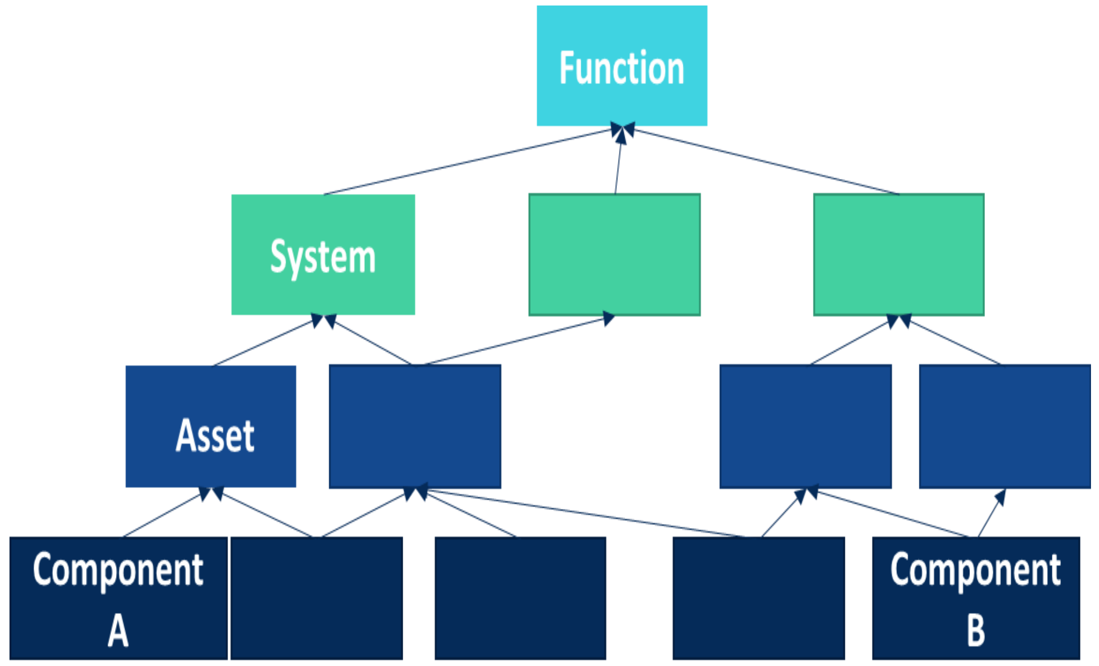
- The project I’m working on focuses on two concepts, cascading failures and uncertainty quantification mainly applied to powers systems.
- A Cascading failure is kind of failure in a system comprising of interconnected parts, in which the failure of a part can trigger the failure of successive parts.
  - Such a failure is especially common in power systems.
- Uncertainty quantification looks to quantifying and characterizing uncertainty in real world systems.
  - Basically, if a normal simulation looks to answer what happens to a system that is subjected to a single set of inputs, UQ tries to ask what happens when the system has many uncertain and variable inputs.



## What is CasCon and How is it connected to my Project?

- CasCon – Modeling framework which allows for the analysis of interconnected components within power and infrastructure systems, and how failures propagate through.
  - If a Components fails how are the rest of the components affected?
  - Many of the aspects are uncertain because they depend on unknown specifics.

- CasConUQ - Modeling programs which allows for the quantification of uncertainties in components of a CasCon Model.
  - Calculating impact of input uncertainties.
  - Gives better understanding of outputs and cascades.
  - Requires more development and algorithms for the analysis of cascading failures in the different infrastructures.

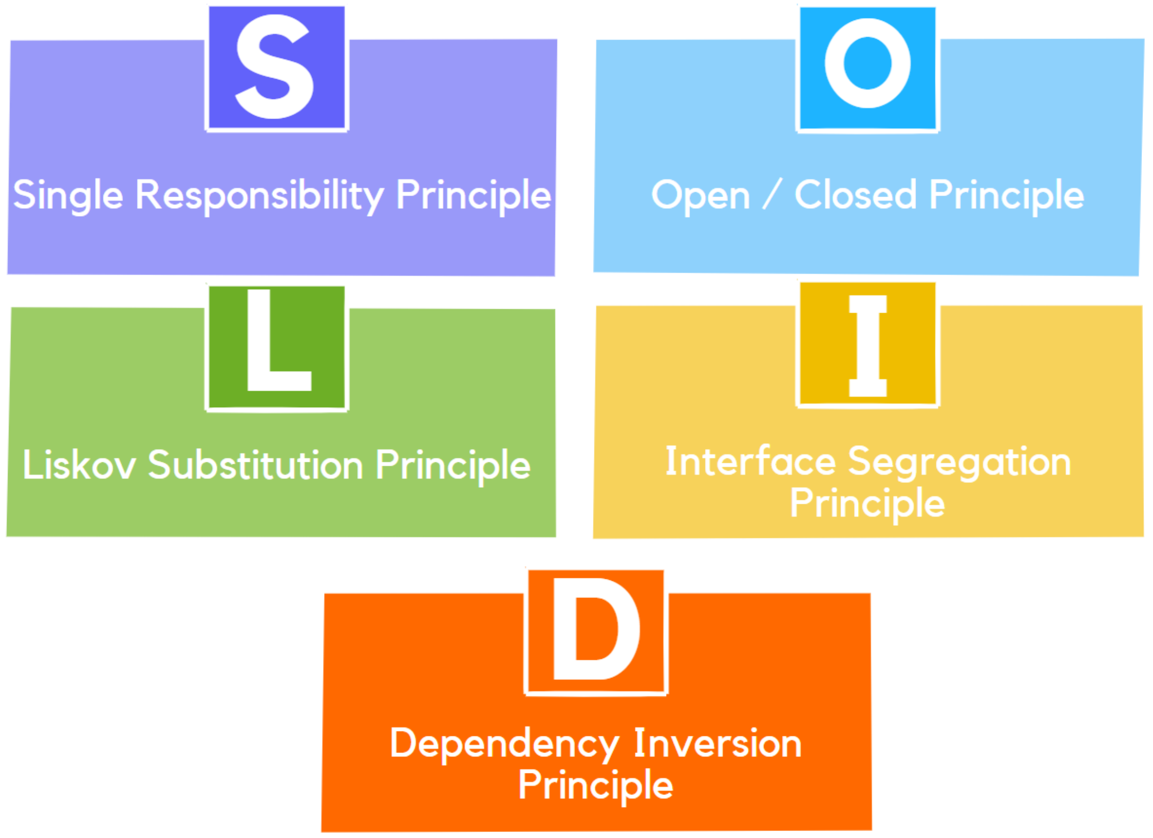


## Approach

PERSONAL APPROACH IN THE DEVELOPMENT

## Software Approach and Followed Principles

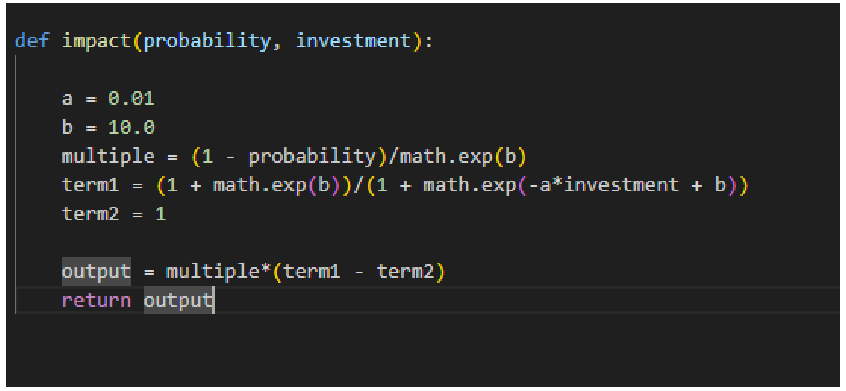
- The main Goal of my tasks is to develop future proof and efficient software classes and methods.
- This allows for methods, variables and other components to be quite modular.
- Meaning, other engineers in the future can very easily extend and add algorithms, equations, functions, etc.….
- To achieve this approach, I use the SOLID SWE principles as my guide.


S
Single Responsibility Principle
O
Open / Closed Principle
L
Liskov Substitution Principle
I
Interface Segregation Principle
D
Dependency Inversion Principle

# CasConUQ Development Using SOLID

- The Works Focuses on the use of:
  - Open/Closed Principle – It states that classes should be open for extension and close for modification.
    - If it works why change it?
    - Add functionality without touching existing code.
  - Interface Segregation Principle – It states that many user specific interfaces are better than a general-purpose one.
    - Allow for flexible code for the future.

- For example, let's say we need to define a function for the CasConUQ to upgrade probability and we know the given algorithm works but maybe In the future another engineers chooses to use a different algorithm.



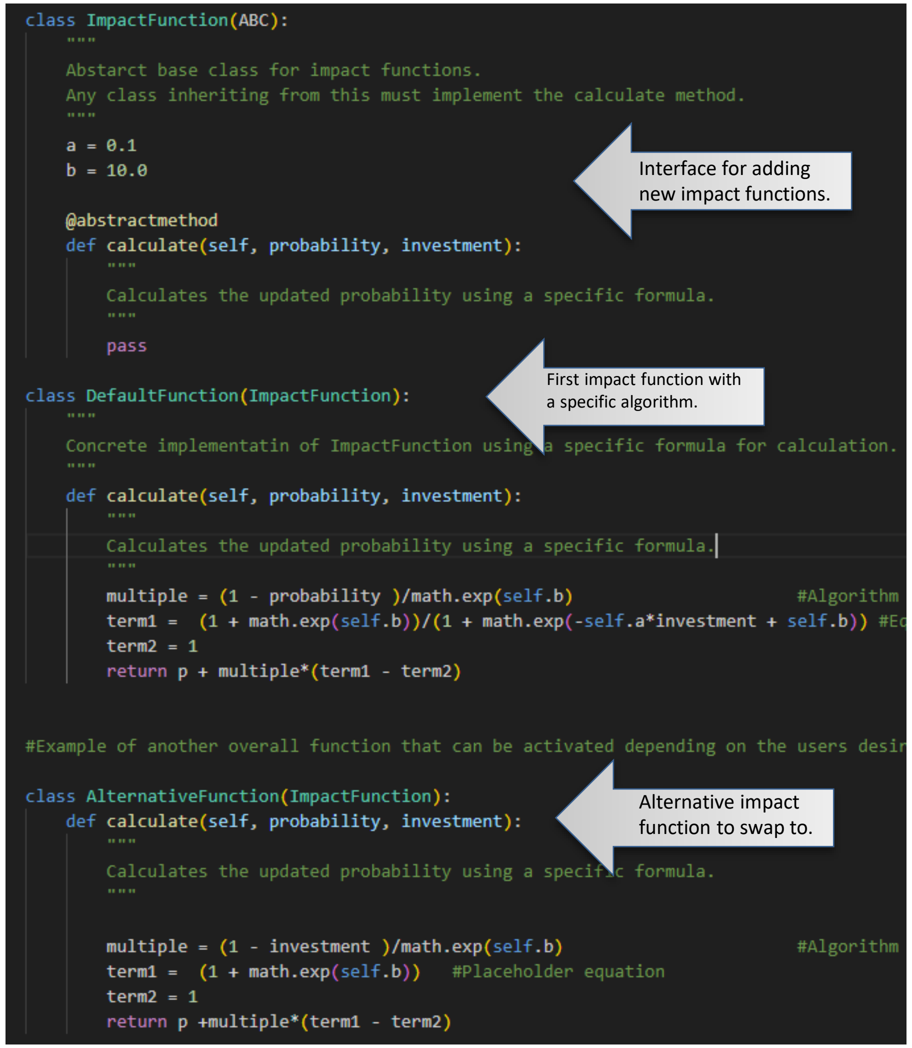
```
def impact(probability, investment):

    a = 0.01
    b = 10.0
    multiple = (1 - probability)/math.exp(b)
    term1 = (1 + math.exp(b))/(1 + math.exp(-a*investment + b))
    term2 = 1

    output = multiple*(term1 - term2)
    return output
```

- We can implement a simple interface to follow and allow other algorithms and equations to be added in the future in case it is found necessary. This would follow the Open/Closed and Interface Segregation Principles as well as the rest of SOLID.

## SOLID Development Example


Interface for adding new impact functions.
First impact function with a specific algorithm.
Alternative impact function to swap to.

```python
class ImpactFunction(ABC):
    """
    Abstarct base class for impact functions.
    Any class inheriting from this must implement the calculate method.
    """
    a = 0.1
    b = 10.0

    @abstractmethod
    def calculate(self, probability, investment):
        """
        Calculates the updated probability using a specific formula.
        """
        pass

class DefaultFunction(ImpactFunction):
    """
    Concrete implementatin of ImpactFunction using a specific formula for calculation.
    """
    def calculate(self, probability, investment):
        """
        Calculates the updated probability using a specific formula.
        """
        multiple = (1 - probability )/math.exp(self.b)                          #Algorithm
        term1 =  (1 + math.exp(self.b))/(1 + math.exp(-self.a*investment + self.b)) #Eq
        term2 = 1
        return p + multiple*(term1 - term2)


#Example of another overall function that can be activated depending on the users desir

class AlternativeFunction(ImpactFunction):
    def calculate(self, probability, investment):
        """
        Calculates the updated probability using a specific formula.
        """

        multiple = (1 - investment )/math.exp(self.b)                           #Algorithm
        term1 =  (1 + math.exp(self.b))   #Placeholder equation
        term2 = 1
        return p +multiple*(term1 - term2)
```

- Here the algorithm and function was first implemented trough an interface and abstract classes which allow for easy expansion of the classes and allows for new functions to be added by other developers in the future.
- In this case we can easily have additional impact functions added in the case a developers decides to have it.
- We can then implement them into the run files for the CasConUQ program depending on the necessity.

## Future Work & Conclusion

- CasConUQ seeks to quantify uncertainties using the CasCon cascading consequences model.
- My work has been to continue its development using the SOLID principle to further develop my skills as a software engineer.
- LLNL has given me an amazing opportunity and learning environment I will truly never forget.

- Continue Development of functions and interfaces in the CasConUQ.
  - Debugging Current program.
  - Developing an interface for Encoding Formats and tests.
  - Continue Learning about power system failures and forms of quantification.
  - Enjoy, Learn and grow here in LLNL, thanks to other engineers, lectures and interns.

## Thank You

CHRES CONSORTIUM OF HYBRID RESILIENT ENERGY SYSTEMS DE-NA0003982

Special thanks to CHRES, LLNL and Dr. Colin Ponce.

Roberto Marrero Ortiz

Lawrence Livermore National Laboratory