Bachelor Thesis

for the Bachelor of Engineering degree program in Mechatronics

Development of a Continuous Integration Framework for OptiSlang Workflows

In co-operation with

Robert Bosch GmbH

Author : Sathvick Bindinganavale Srinath
Matriculation Number : 4020025

Supervisor : Mr. André Haeitmann Dutra
1st Examiner : Prof. Dr.-Ing. Andreas Schiffler
2nd Examiner :

Submission Date : 06/10/2024

Contents

List of Figures

List of Tables

Abbreviations

CAE ..... Computer-Aided Engineering

CD ..... Continuous Deployment

CI ..... Continuous Integration

GUI ..... Graphical User Interface

MOO ..... Multi Objective Optimization

PIDO ..... Process Integration and Design Optimization

SOO ..... Single Objective Optimization

Chapter 1

Introduction

1.1 Overview

This thesis explains about the creation and development of a framework for Optislang workflows.

1.2 Objective

The objective of this thesis are as follows:

- Development of a method to create standalone modules in Optislang, based in MATLAB and Python.(only code based, without using GUI).
- Implementation of modules and workflows without the help of GUI.
- Creation of a framework in Python to create and test modules and workflows in Optislang.
- Establishment of a strategy for automated integration testing in Github for modules based in Python and MATLAB.

1.3 Outline

- In Chapter 2, we are going to discuss about the MOO project, its role in the company and the modules created in Python and Optislang.
- In Chapter 3, we understand the creation of a framework to create standalone modules and workflows in Python.
- In Chapter 4, we will look into the automated integration testing in Github.
- In Chapter 5, we will discuss the results and evaluate the impact of the solution.
- Finally, Chapter 6 provides us the summary of the work, highlighting the achievements and feedback for development in potential areas.

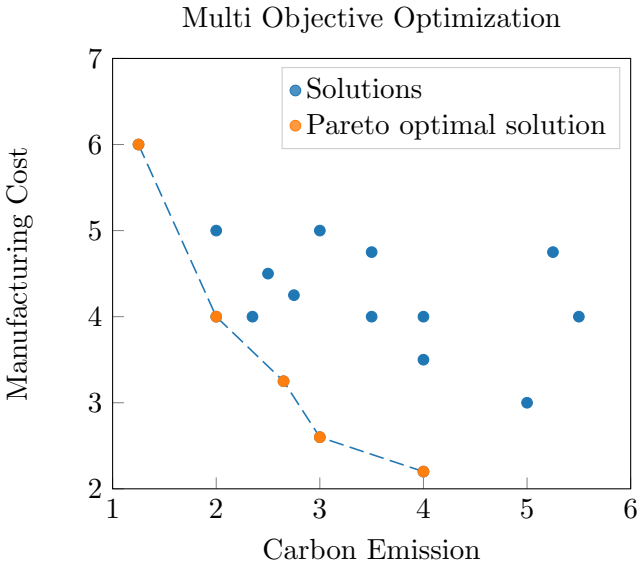
Chapter 2

Multi Objective Optimization (MOO)

2.1 What is MOO?

In today's increasingly complex world, decision-makers often face the challenge of optimizing several conflicting objectives simultaneously. Multi Objective Optimization (MOO) is an optimization that deals with such problems, where multiple objective functions are optimized simultaneously. To understand MOO better, let us consider an example.

Example: Let us consider an example of a car manufacturer. The car consists of many components like engine, body, wheels, etc which can be tweaked. In our case, the manufacturer wants to optimize the car for two objectives: lower manufacturing cost of the car and lower carbon emissions. With considering the input parameters and the objectives, we get many solutions as shown in Figure **??**



In a MOO problem, there typically is no single best solution. Rather, the *goal* is to identify a set of solutions that are optimal in terms of all objectives. In Figure **??**, the best solutions for the given objectives is indicated in red known as pareto optimal solutions. A solution is said to be pareto optimal if no other solution can improve on any of the objectives without worsening at least one of the other objectives.

2.2 Difference between MOO and SOO

Optimization problems, whether single-objective or multi-objective, have the same goal: to find the best solution(s) to a given problem. However, the approach to solving these problems is different.

In Single Objective Optimization (SOO), the goal is to optimize a single objective function, which can either be maximized or minimized. The problem is simpler to define and solve because it involves only one objective. To calculate SOO, we can use methods like gradient descent, linear programming, etc.

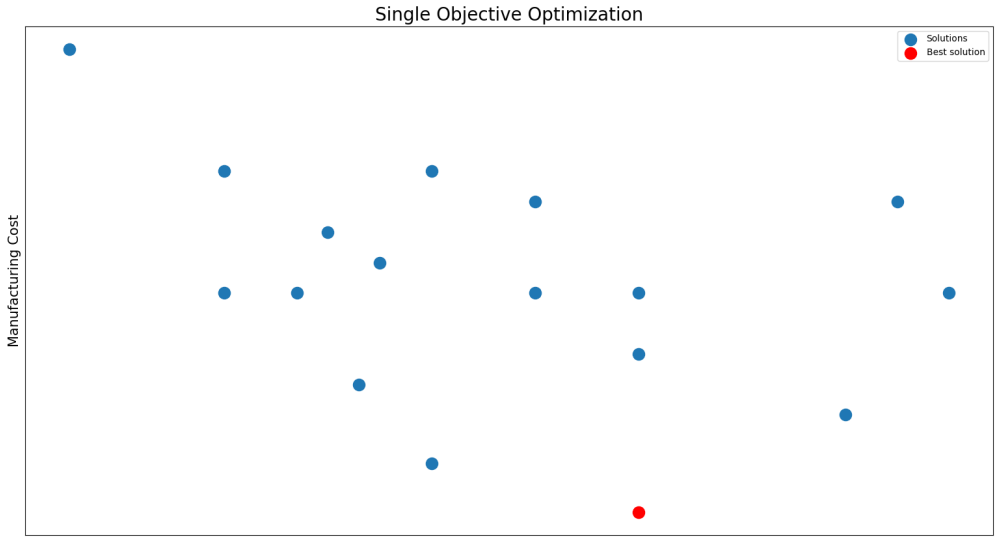


Figure 2.1: Example of SOO

In Figure 2.1, we have considered the same example given in section 2.1. But, here, we are considering only one objective, which is to minimize the manufacturing cost. The best solution is indicated in red.

In Multi Objective Optimization (MOO), the optimization involves two or more objective functions simultaneously. The problem is more complex because the objectives are often conflicting. Unlike SOO, where we have a single best solution, in MOO, we have pareto optimal solutions. To calculate MOO, we can use methods like pareto optimization, scalarization method, weighted sum method, ϵ-constraint method, etc.

While SOO focuses on finding the best solution according to a single criterion, MOO addresses the more complex task of balancing multiple, often conflicting objectives. The choice between SOO and MOO depends on the nature of the problem at hand and the goals of the decision-maker. Understanding the differences between these approaches is crucial for selecting the appropriate optimization technique and achieving the desired outcomes.

2.3 Optislang

To calculate MOO, we need a software platform that can handle the complexity of the problem. Ansys Optislang [1] is such a software platform, which is used for design exploration, CAE based sensitivity analysis and optimization in conjunction with any product development tool. It is a Process Integration and Design Optimization tool or in short, a PIDO tool. Process Integration refers to automate and orchestrate manual simulation processes and to realize complex workflows. Design Optimization aims for better understanding of your design, optimizing the product, identify an improved design which has the desired qualities and resulting in a best design by reliability analysis and statistical analysis.

Optislang uses several solvers to look into aspects like mechanical, technical, mathematical and any other problems. This is easier in Optislang as it provides integration to create toolchains of many external programs like ANSYS, MATLAB, Excel, Python, CATIA and many more.

Our department utilizes Optislang for solving MOO problems, as it includes algorithms specifically designed for MOO.

2.4 Modules and Workflows

2.4.1 Modules

Modules are created by the system developers. Modules include a simulation model as a calculation with defined interfaces for coupling with other modules. These modules are either defined in MATLAB or Python. Each module is designed to tackle/improve a specific issue. To document and collaborate with other system developers, each module is versioned and stored in a specific manner in a repository in GitHub Enterprise.

Figure 2.2 shows us an example of how each module is maintained in our GitHub.

- `01_Specification` has all the requirements for the module to run.
- `02_Model` contains all parts to run the model. This can also be used as a playground for the development of a model.
- `03_Test` withholds all relevant documents regarding unit tests or integration test can be found here. This is one of the important part of the module.
- `04_Input` carries all the initial parameters or functions to be defined at the start of a module.
- Documentation explaining functioning and usage of the module can be found under `05_Documentation`.

2.4.2 Workflows

Our department develops automatized workflows for power electronics products considering functional loads to reliability indications. Workflows are a sequence of modules designed for a fast calculation performance characteristic like temperature or reliability indication. To develop a workflow, Optislang is used.

Every architectural workflow has a GitHub repository that is maintained in a similar way to how modules are being maintained.

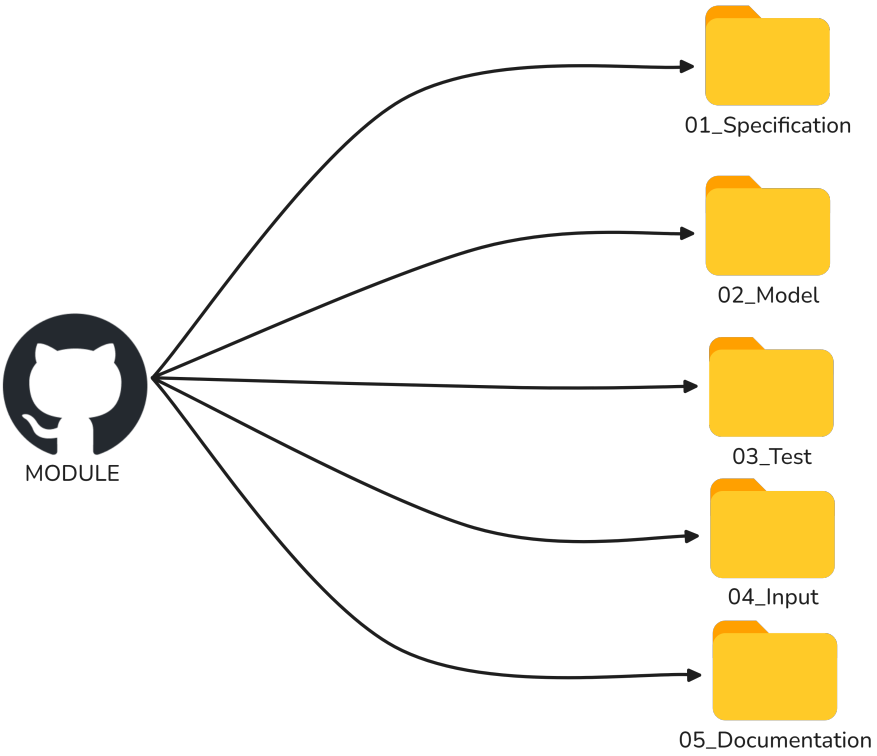


Figure 2.2: GitHub repository of Arrhenius module

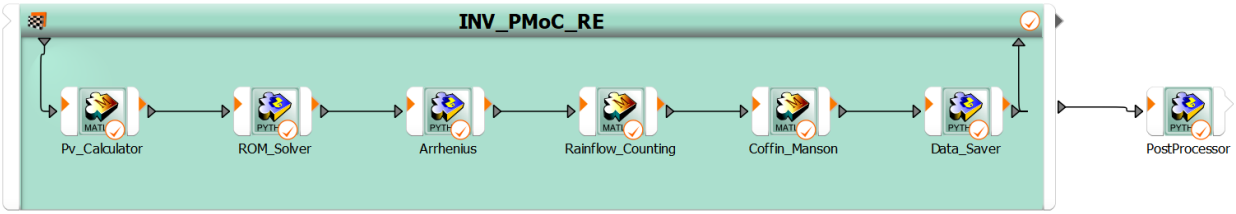


Figure 2.3: Example of a workflow in Optislang

2.5 Current Problem

Chapter 3

Application of my Thesis

To overcome the problems discussed in section 2.5, this thesis proposes a solution to automate the process of testing standalone modules in Optislang. Since, the process is automated, the testing of modules needs to be done without the help of the GUI. To achieve this, a Python [2] framework is created to test the modules in an according manner. To use the framework in an automated manner, a CI pipeline is created using GitHub Actions. The pipeline is triggered whenever a new commit is pushed to the repository. The pipeline runs the tests on the modules in a virtual machine and checks if the results are as expected. If the tests fail, the pipeline notifies the developer about the failure. The developer can then look into the issue and resolve it.

3.1 DevOps

Devops is the combination of a set of practices, tools which helps to automate and integrate the processes between software and organizations.

DevOps practices play a crucial role in the development of the automation process described in this thesis. By integrating Continuous Integration (CI) and Continuous Deployment (CD) pipelines, we ensure that the testing of modules is efficient and reliable. This helps us to improve productivity and reduce human error. The DevOps approach allows for seamless collaboration between development and operations teams, ensuring that the testing framework and the modules it tests are consistently maintained and updated. This integration of DevOps practices not only enhances the quality of te software but also accelerates the development lifecycle, enabling faster delivery of new updates and features.

In summary, the application of DevOps in this thesis demonstrates how modern software engineering practices can be applied to automate and streamline the testing process, leading to more robust, efficient and reliable software solutions.

3.2 Continuous Integration (CI)

3.3 Code quality

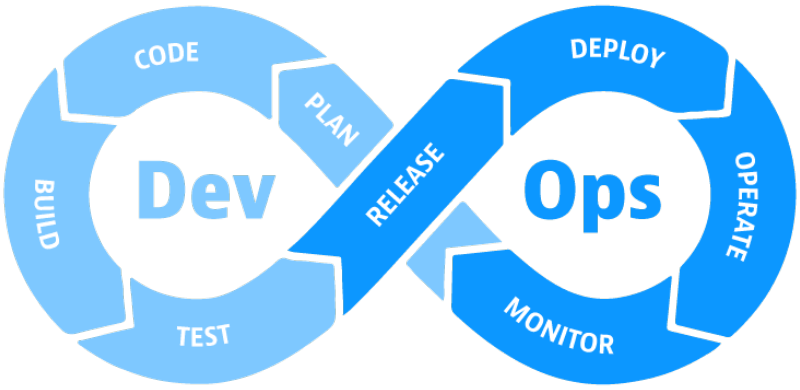


Figure 3.1: DevOps lifecycle

Chapter 4

Creation of a CI Framework

Bibliography

[1] Ansys optislang. https://www.ansys.com/products/connect/ansys-optislang. Online; Accessed 10/05/2024.

[2] Python. https://www.python.org/. Online; Accessed 07/08/2024.