Parameter Optimization Simulation tool for the Impact Framework

Product Vision and Architecture Document

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Authors | Comments |
| 1 | 29/11/2023 | All group members | Initialise and first add contents |
| 2.1 | 01/12/2023 | Telma Gudmundsdottir | Refine background information |
| 2.2 | 03/12/2023 | Dimitrios Bouras | Add possible projects |
| 2.3 | 05/12/2023 | Mohamed Saleh | Add domain scenarios |
|  |  |  |  |

Contents

[1. Purpose and scope of the document 2](#_Toc1380026808)

[2. Product Vision and Requirements 3](#_Toc690267678)

[2.1 Background Information 3](#_Toc1910562295)

[2.2 Business Goals 4](#_Toc847809229)

[2.3 Stakeholders 5](#_Toc1160068963)

[2.4 The Product Context 7](#_Toc865253204)

[Core Features of the System 7](#_Toc153061267)

[2.5 Overview of Requirements 8](#_Toc1099957870)

[2.6 Domain Scenarios (can be converted into stories later (visual representations) 8](#_Toc231567191)

[3. Possible projects 10](#_Toc1051433064)

[3.1 Web UI – Simulation tool 10](#_Toc1205033193)

[3.2 AWS model 10](#_Toc72846091)

[3.3 Carbon Aware Computing Simulation tool 10](#_Toc331795139)

[3.4 Right Sizing Simulation tool 10](#_Toc1710737379)

[3.5 Parameter Optimization Model using Search-based Software engineering 11](#_Toc1261469788)

[4. Product Architecture 11](#_Toc212696532)

[4.1 System components 12](#_Toc2084668980)

[4.2 Functional Views 12](#_Toc1674043463)

[5. Development and Evaluation Plan 13](#_Toc1589732612)

[5.1 Development plan 14](#_Toc1837053606)

[5.2 Evaluation plan 14](#_Toc875125760)

[5.3 Future work (Don’t know if this need) 14](#_Toc1572274181)

# 

# Purpose and scope of the document

*describes the purpose and scope of the document, its intended audience, the approach followed to elaborate this document (e.g. what method was used, who was consulted) and the status of the document.*

This document describes a parameter optimisation model for the Impact Engine Framework (IEF) using search-based software engineering. The model enhances IEF's environmental impact measurement features by providing suggestions on how a given software can be improved. By analysing and fine-tuning software parameters, the model aims to find the best balance between environmental friendliness and operational efficiency to aid users in decision-making.

The team frequently consulted with the client, Green Software Foundation, to ensure that the project vision stayed relevant and valuable to the company for the foreseeable future. The primary contact points at Green Software Foundation were Sophie Trinder, Joseph Cook, Gadhu Sundaram and Asim Hussain. While writing the report, the team held three weekly meetings: one with the clients, one with the supervisor and one where only the team was present. Meetings with the clients were conducted via Google Meet, with additional correspondence done through email. During these meetings, the team could ensure that the project aligned with the client's interest and get further information regarding components like the system and requirements. In the team meetings, the team went over tasks that needed to be done and gave feedback for finished tasks. This time was also used to discuss possible problems or answers that team members might have.

The report is a collaborative effort between all members of the team. Firstly, it explains the background information, business goals, stakeholders, core features of the system, overview of requirements, and domain scenarios. The product evolved significantly during the initial stages, and many original ideas could not be implemented due to IEF being in the alpha state. The ideas would rely on implementations that were likely to change in the near future. An overview of these projects and why they were not chosen is detailed in the report. Finally, the product architecture and a development and evaluation plan are outlined.

This document aims to provide a detailed description, enhance understanding, and serve as a reference for further development of the model.

# Product Vision and Requirements

## Background Information

Software's environmental impact is multifaceted, including greenhouse gas emissions, water usage, and energy consumption. To address these issues, the Green Software Foundation, a non-profit under the Linux Foundation, is dedicated to reducing software's ecological footprint through various projects.

A key project of this foundation is the Impact Engine Framework (IEF), which has evolved throughout the Green Software Foundation's existence. A notable development is the creation of the Software Carbon Intensity (SCI) Specification, which defines a methodology for calculating a software system's total carbon emission. Initially, the challenge was data acquisition, leading to the launch of a project focused on generating necessary data sets. The team later realised that various existing data sources were available. They anticipated that the challenge would shift from sourcing data to selecting the appropriate data set for each use case. This initiative eventually evolved into the SCI Guide, which provides comprehensive documentation on existing data sets and their application.

The project's current phase aims to establish formal standards and tooling, transforming software measurement into a disciplined and widespread practice. The SCI Specification is positioned as the fundamental standard, while the Impact Framework serves as the essential tooling component.

An essential part of the IEF is composability. To make the framework as composable and configurable as possible, a component called a model can be plugged into the framework for additional functionality. There is a standard library of models maintained by the IF core team and another repository of models the community maintains. This report expands on a simulation model that goes beyond calculating software's possible environmental impact. With this model, we aim to help the user even further by using search-based software engineering to analyse and optimise parameters in the form of input data. The output of the model will contain the result and advise the user on the best way to reduce the environmental impact of the software.

Scope of the work

* Development of the optimisation simulation tool
* Integration between each model
* Manifest File
* Analysis of the needs of the users
* Tests
* Documents and tutorials

Scope of the product

* Measuring environmental impact due to **environmentalism consideration**,
  + users might try to find a configuration that achieves the **lowest environmental impact**.
* Evaluate and compare the energy cost for technical configuration due to **financial consideration** (especially for companies),
  + users might want to find a configuration that achieves **best trade-offs between cost and performance**.
* Evaluate and compare the energy consumption for **battery management optimisation** for software products,
  + users might want to find an **energy-efficiency-optimized solution** for the products.

## Business Goals

|  |  |  |
| --- | --- | --- |
| Goal | Details | Explanation/Justification |
| Optimisation | * Provide users optimisation suggestions / solutions for their input configuration. * Demonstrate trade-offs between (financial) cost, performance and environmental impact to users. | * Giving users with an idea of improvement should be one main purpose of this framework. * Users should not only know about the environmental impact for their software, but also have an idea of how to make it better. |
| Comparison | * Enable users to compare their input configurations in parallel, explicitly highlight those significant differences. * Compared data should not only come from the input parameters, but also include the simulation results’ output, and output that might produce from middle-process. | * Comparison between different options should be a frequently demanded functionality. Especially for software developers / companies. * In reality, environmental issues are not the only thing considered by stakeholders, but also cost and performance. Giving users a comprehensive understanding of the difference and trade-offs between their technical options means something more to them. |
| Automation | * Automate the simulation procedure as more as possible, minimize unnecessary manual operations. * Only crucial operations should require decisions from users. * Automation of deployment should also be taken into account. | * The entire IEF and optimisation process should ideally require as little manual operations as possible * So we can promote IEF to a wider range of target audience. |
| Versatility/Flexibility | * There should be inputs for multiple parameters such as location, time, etc * Allow more flexibility when it comes to generating statistics and ideas for reducing emission | * It does not exist an universal solution for all softwares * So users can choose what parameters they wish to optimise * Give more flexibility and can also help the simulation to generate better suggestions |

## Stakeholders

*Describes the main stakeholders and their interests.*

The IF web tool has the following stakeholders.

| **Stakeholder** | **Concerns, wishes and expectations related to the project** |
| --- | --- |
| Anyone involved in software development | * They will use the framework to assess and optimize the environmental impacts of their applications. * IF will impact the configurations of their app, like their design and code. It will drive them to use more efficient algorithms. * They hope that the framework is user-friendly and intuitive [1]. |
| Green Software Foundation | * GSF may concerns the development and operation costs, reputation and strategic objectives. * GFS expects that software measurement will become a mainstream activity, i.e. thousands of professionals will be working to decarbonise software and enable businesses to thrive in the software measurement ecosystem [2]. |
| Open-source community | * They care about whether all contributions from the project comply with open-source licenses to foster transparency and community involvement [1]. * They hope that all developers in the open-source community can be able to contribute, benefit from and adopt it [1]. |
| Large corporations | * They might use the environmental impact of their software as a selling point. * IF can help them find ways to optimise the energy efficiency of their software, potentially saving money on energy consumption. * They care the availability of IF since they expect being able to access the framework on all environments and platforms with ease [1] [3]. |
| Cloud Service Providers | * They will provide essential information such as the energy consumption, server utilization of various computing environments for the framework. * IF may affect the infrastructure or resource provisioning of cloud service providers. |
| **Secondary Stakeholder** | **Concerns, wishes and expectations related to the project** |
| NGO | * They hope that the framework would have a meaningful impact on the environment and could raise public awareness of the environment. |
| Environmental Engineers / Researchers | * Their works or studies will be affected by the data based on the result generated by the framework. |
| National and local government politicians | * The data and insights gained from the framework will affect their policy and regulatory standards related to environmental sustainability. |
| UCL Team | * Develop, implement, test, maintain and evolve the user-friendly solutions to help users simulate, measure, and monitor the impact of the software on the environment. |
| Energy supplier | * They may expect IF to provide insights into the future demand for energy based on the the result of the framework. This can assist them in planning and optimizing energy production and distribution. |
| Competing framework or tools | * The IF framework may affect their users’ stickiness. They don't want IF to be the first choice for users who need environmental impact assessment framework. |
| General Public | * IF will increase their demand for green technology solutions, which can drive innovation and development of other environmental technologies. |

## The Product Context

## Core Features of the System

**1. Measurement Model**

* Extend the IEF by developing a flexible measurement model for software environmental impact
* Extend the capabilities of the existing Software Carbon Intensity (SCI) specification to offer a flexible measurement model for software environmental impacts
* Easily calculate software impact metrics, such as carbon, water, or energy without the need for writing any code

**2. Simulation Tool**

* Create a simulation tool to project and analyse the environmental impact during the design phase, enabling users to make informed choices
* This feature gives users the ability to optimize their software design decisions by evaluating different scenarios, based on their budget, context, and other business constraints
* Allow users to easily change the model to represent “what if” scenarios, reducing the cost by validating different software architectures without making any changes

**3. Modular Design**

* Modular design will allows for the integration of custom models or selection from a range of pre-existing models, knows as plugins
* Follow the IF design philosophy that all models should be plugins, so that the configuration is as modular as possible
* Provide clear separation between the engine and models

**4. Integration Capability**

* Seamlessly integrate with various environments, such as cloud, bare-metal, and virtualized systems, to assess software impacts comprehensively

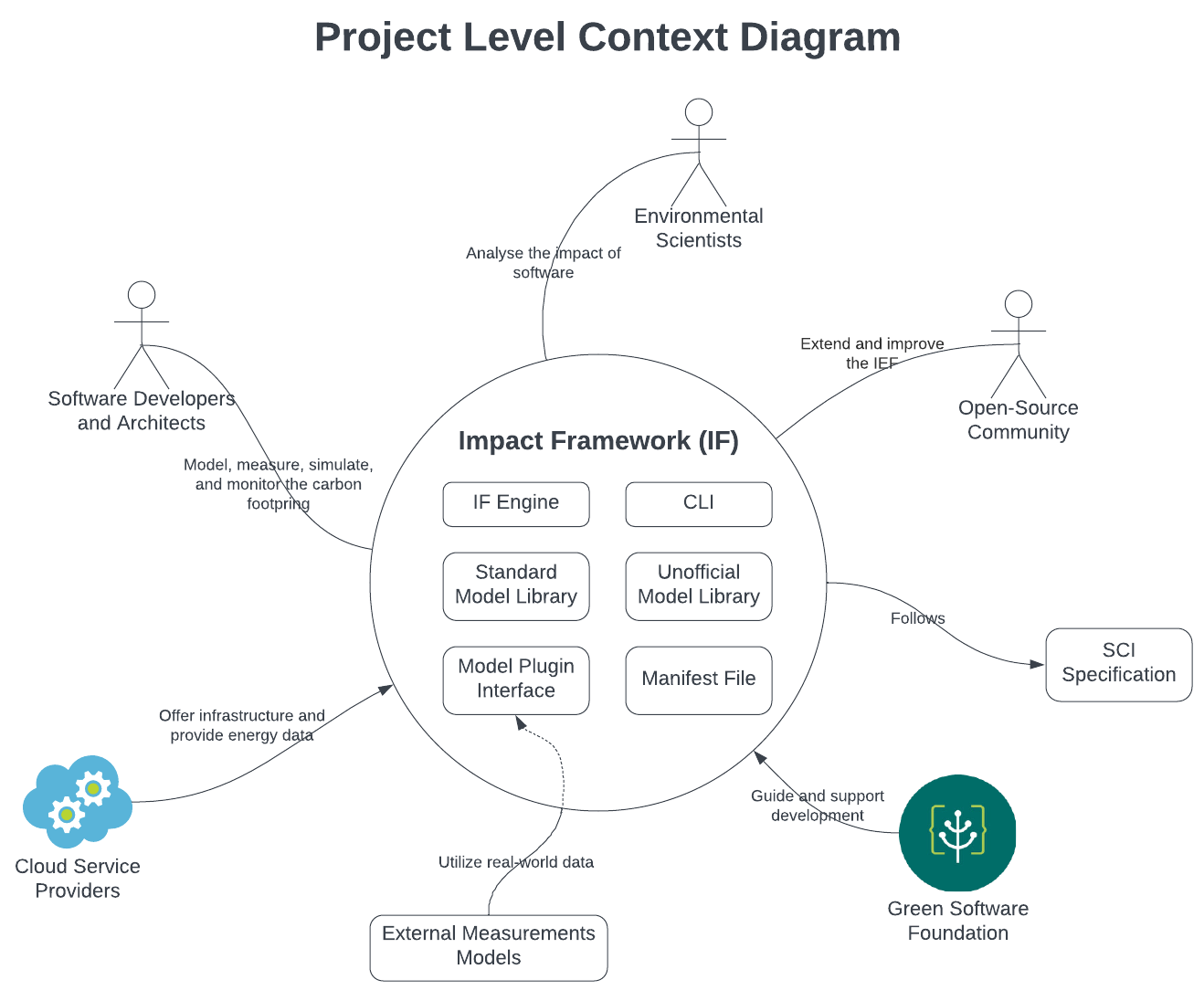
**5. User-friendly Interface**

* The design is easy-to-use, intuitive and accessible for software engineers and environmental scientists, encouraging widespread adoption
* Simple installation via *yarn* or *npm* package managers
* All that is needed, is to define the scope and parameters in a simple YAML manifest file, known as impl, the IF handles the rest

**6. Compliance**

* Adhere to the guidelines and objectives of the Green Software Foundation, promoting sustainability in software

Context diagram of the system as-is:



*Context diagram describing the world in which the system operates and the interactions of the system with users and other systems.*

## Overview of Requirements

*A summary of the main architecturally significant functional and quality requirements for the system.*

*Do not include detailed requirements - if you have detailed requirements such as gherkin scenarios you can present them in an appendix, or better in an online repository with a link.*

## Domain Scenarios (can be converted into stories later (visual representations)

**Scenario for existing System:** A company using Azure instances tasks a Developer with assessing the environmental impact of its software

1. The Developer installs Node.js and initialises a project directory for the Impact Framework.
2. The Developer installs the Impact Framework and selects the appropriate model plugins for the project, this includes models that are designed for Azure instances.
3. The Developer creates a manifest file which includes the configurations and input data that is required to measure the application’s impact on the environment.
4. The Developer defines the components of the application and the relationships between them in the manifest file
5. The Developer runs the Impact Framework CLI tool (impact-engine) with the path to the manifest file to start the impact calculation
6. The Developer reviews the output data displayed in the console, which shows the result of the calculation of the environmental impact of their software.
7. The Impact Engine saves the output data to another YAML file so it can be used for reporting or further analysis

**Parameter-Optimization Scenario:** A company running servers on the cloud wants suggestion on how to reduce the environmental impact of its software

1. The Developer installs Node.js and initialises a project directory for the Impact Framework.
2. The Developer installs the Impact Framework and selects the appropriate model plugins for the project, including the Parameter-Optimization model.
3. The Developer creates a manifest file which includes the configurations and input data that is required to measure the application’s impact on the environment.
4. The Developer defines the components of the application and the relationships between them in the manifest file.
5. The Developer runs the Impact Framework CLI tool (impact-engine) with the path to the manifest file to start the impact calculation and to receive suggestions for improvement.
6. The Parameter Optimization model uses search-based techniques to find optimal combinations of server locations and operation schedules that could reduce the environmental impact.
7. The Parameter Optimization model outputs a set of optimized parameters, e.g. changed server locations, operation times, and predictions of their environmental/operational impacts.
8. The environmental impact calculations for each component are repeated, using the suggested set of optimized parameters.
9. The Impact Engine saves the output data to another YAML file so it can be used for reporting or further analysis.

*Section 2 can include additional subsections relevant to the business analysis of the system under study, for example a domain conceptual model, description of workflows, domain scenarios, a risk analysis, reference to standards, analysis of competitors, technology opportunities, etc.*

# Possible projects

## Web UI – Simulation tool

The original idea was to create a web UI for the IEF that would also serve as a simulation tool. By providing an accessible and user-friendly interface for the Impact Framework, which currently operates through a command line, the team aimed to help users model, measure, simulate, and monitor the environmental effects of their software without the need for high level programming knowledge. The IF tries to tackle a very difficult and complex issue: the environmental impacts of modern software, which often runs on diverse environments and involves numerous components .In turn the web UI would be able to simplify the use of the IF even more by adding another tool, an abstraction level above it so that users don’t have to interact with the IF at all. By simplifying the input process, and enhancing the output visualization, it would be possible for users to try different inputs and scenarios to find the one with the smallest environmental impact.

Unfortunately, this option is not viable due to the maturity level of the project. The IEF has just come into alpha and everything a UI will rely on (types and number of inputs) will change in the next 6-12 months.

## AWS model

There is currently an unofficial model for the IEF called Azure-importer that allows you to provide some basic details about an Azure virtual machine and automatically populate your IMPL input file with usage metrics that can then be passed along a model pipeline to calculate energy and carbon impacts.

A proposed project idea was to create a similar model for the AWS cloud platform to measure the environmental impact of VMs or other cloud services provided by them. Such a model would help expand the IEF userbase with AWS users which would be extremely significant since AWS is the current market leader for cloud infrastructure with a 33% market share. After discussion with the IEF development team though it was decided that this was not a viable project since AWS does not expose similar APIs to Azure to get the required metrics. Thus, it would not be possible to provide a working model within the available timeframe.

## Carbon Aware Computing Simulation tool

A model which if added to a pipeline would shift a workload to the least carbon moment

<More information is awaited from the Green Software Foundation Developers>

## Right Sizing Simulation tool

Most users pick servers that are far too powerful for their needs, so they run at low utilizations.

Right sizing is the task of picking a more appropriate server.

A model which helps figure out the right sized server to use given the workload utilization.

<More information is awaited from the Green Software Foundation Developers>

## Parameter Optimization Model using Search-based Software engineering

The team is tasked with developing a new model that seamlessly integrates into the current model pipelines. This model will be incorporated following the data import models, like azure-importer, but preceding the models responsible for converting energy measurements into various impact metrics, such as carbon emissions.

* This option requires the implementation through a model and not through an external tool that invokes the impact engine for two reasons:  
  The creation of models for such functionality aligns with the IEF team’s way of working and best practices and it was strongly advised by them to pursue the model approach
* The data that will need to be manipulated rest in the middle of the calculation pipeline so a tool before or after the impact engine would not have access to that.

Utilizing established search-based techniques and similar heuristics, the model aims to conduct efficient explorations and identify parameter combinations within acceptable limits. The ultimate goal is to minimize the impact metrics as desired.

The first and most obvious parameters to be optimized would be location and time but further investigation might reveal more parameters that could be optimized using this method.

# Product Architecture

## System components

System as-is components:

* **IEF CLI:**

The only interface currently available for interacting with the IEF to utilize its functionalities.

* **IEF Impact Engine:**

Performs impact computations based on the input IMPL file and generates a Manifest file containing the computed impact.

* **IMPL Manifest File:**

A formal report in YAML format that details all assumptions, inputs, and models used for calculating the impact. It describes the system to be measured and serves as an input.

* **Model:**

A model converts an input into a specific output impact metric.

* **Model Pipeline:**

A chain of models. Calculating a component's impacts often requires using multiple models in sequence. Each model takes as input the outputs of the previous model in the chain, all working together to calculate impacts from inputs.

* **Standard Library of Models:**

A standard library of models built and maintained by the IF core team.

* **Unofficial Library of Models:**

A separate repository for models that community members maintain.

* **Model Plugin Interface:**

A common class interface that every Model Plugin needs to extend and implement.

* **Output Manifest File:**

A formal report in YAML format detailing not just the end impact but also all the assumptions, inputs, and models used in calculating the impact. (Can be in a CSV format as well.)

## Input and Output Manifest files

An Impact Manifest is a file format based on [yaml](https://circleci.com/blog/what-is-yaml-a-beginner-s-guide/) to represent a [Graph](https://github.com/Green-Software-Foundation/if-docs/blob/master/docs/06-specification/graph.md), it's also sometimes called Impact YAML or IMPL when we are talking about the input manifest files.

Just like a Graph, an IMPL is a calculation manifest containing everything you want to measure and how you want to measure it.

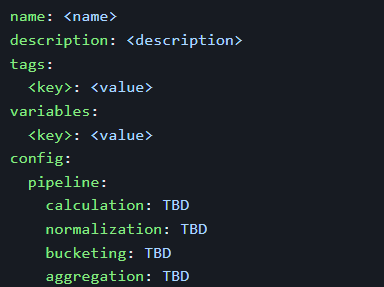
Manifest files being YAML means it's more human-readable and can be used as a formal method of writing use cases, such as SCI use cases.

Manifest files can be computed on the command line using the [Impact-Engine](https://github.com/Green-Software-Foundation/if-docs/blob/master/docs/06-specification/impact-framework.md) tool, printing out the results to file or STDOUT. This output file is called the Ompl file.

**Use Cases:**  
  
There are several use cases for a Manifest file.  
  
1)A formal report  
  
An Manifest file is a computable calculation manifest. A formal report detailing not just the end impact, but all the assumptions, inputs, and models used in calculating the impact.  
  
Being a very formal structure, it can be parsed by software, compared to other reports, adjusted, run, and verified.  
  
Currently, in the GSF several case studies have been written to calculate an SCI score for an application, these can all be re-written in Manifest file format.  
  
2)An executable impact calculation manifest  
  
The command line tool [Impact](Impact.md) can compute an Manifest file and generate impact metrics.   
  
3)To bootstrap code  
  
Manifest files will be able to represent simple calculation manifest but to handle larger, more complex systems, we will have to write Graphs as code using our SDK.  
To help bootstrap the process, humans can write the high-level structure using Manifest file and run through a tool to generate starter Graph code in any language our SDK supports.

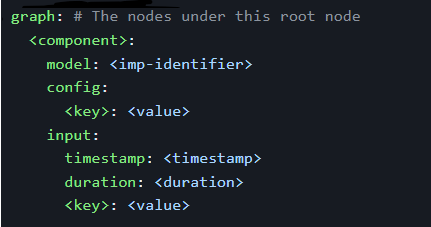
**Structure of the Impl file:**

Genral Info:



Then the graph is described which could have the following different fields:

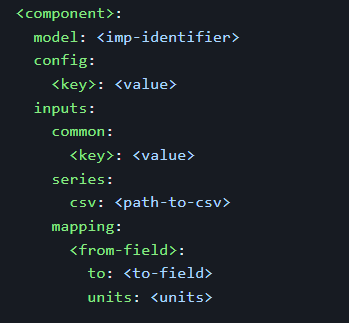
Single Input:



Multiple Inputs:

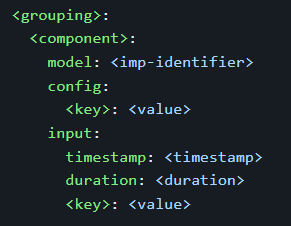


Multiple Inputs from csv:

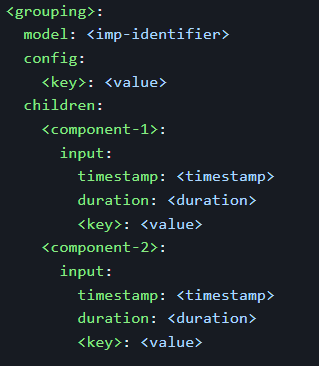


And the above components can be grouped in 2 different ways:

Single Grouping:



Advanced grouping:



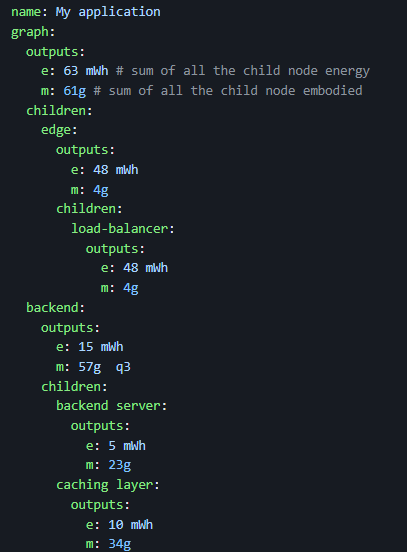
**Example:**

A simple 3 component web server application running on GCP, Azure, and AWS and using multiple models and specifically calculating an SCI score:





Once the above Impl is computed through the Impact Engine it will return an Ompl file like this:



## Models

There are several models used by the Impact Engine. There are the official models which are part of the standard library of models and the unofficial models which are part of the unofficial library of models. We will provide a summary of all the existing models at this point in time and explain a few of them in more details while also providing example impl and ompl files.

### Official Models

* [Cloud instance metadata](https://github.com/Green-Software-Foundation/if-models/blob/main/src/lib/cloud-instance-metadata/README.md): Looks up detailed metadata about a given cloud instance type, including the physical processor being used.
* [E-MEM](https://github.com/Green-Software-Foundation/if-models/blob/main/src/lib/e-mem/README.md): Calculate the energy expended due to memory usage, by multiplying the energy used in GB by a coefficient.
* [SCI-E](https://github.com/Green-Software-Foundation/if-models/blob/main/src/lib/sci-e/README.md): Calculates the sum of all energy components.
* [SCI-M](https://github.com/Green-Software-Foundation/if-models/blob/main/src/lib/sci-m/README.md) - Calculates the embodied carbon for a component.
* [SCI-O](https://github.com/Green-Software-Foundation/if-models/blob/main/src/lib/sci-o/index.ts) - Calculates the operational carbon from the total energy and grid carbon intensity.
* [SCI](https://github.com/Green-Software-Foundation/if-models/blob/main/src/lib/sci/README.md): Calculates the software carbon intensity.
* [SHELL](https://github.com/Green-Software-Foundation/if-models/blob/main/src/lib/shell/README.md) - A model that enables external models in any language to be run in a child process
* [TDP-FINDER](https://github.com/Green-Software-Foundation/if-models/tree/main/src/lib/tdp-finder): Looks up the thermal design power for a given processor in a local database.

### Unofficial Models

* [Azure importer](https://github.com/Green-Software-Foundation/if-unofficial-models/blob/main/src/lib/azure-importer/README.md): Grabs usage metrics from an Azure virtual machine, given user credentials and virtual machine details.
* [Cloud Carbon Footprint](https://github.com/Green-Software-Foundation/if-unofficial-models/blob/main/src/lib/ccf/README.md): Calculates usage metrics using the Cloud Carbon Footprint APIs.
* [WattTime](https://github.com/Green-Software-Foundation/if-unofficial-models/blob/main/src/lib/watt-time/README.md): WattTime is an external service for looking up grid emissions based on location.
* [TEADS-CPU](https://github.com/Green-Software-Foundation/if-unofficial-models/blob/main/src/lib/teads-curve/README.md): Calculates the energy in kWh used by the CPU
* [TEADS-AWS](https://github.com/Green-Software-Foundation/if-unofficial-models/blob/main/src/lib/teads-aws/README.md): Calculates the energy in kWh used by the CPU using a model specific to AWS instances.
* [Boavizta](https://github.com/Green-Software-Foundation/if-unofficial-models/blob/main/src/lib/boavizta/README.md): Calculates energy and embodied carbon using the Boavizta APIs.

### Detailed Examples:

**Sci-e:** it is a model that simply sums up the contributions to a component's energy use. The model returns energy which is used as the input to the sci-o model that calculates operational emissions for the component.

### Inputs:

### At least one of the following are required:

* energy-cpu: energy used by the CPU, in kWh
* energy-memory: energy used by memory, in kWh
* energy-gpu: energy used by GPU, in kWh
* energy-network: energy used to handle network traffic, in kWh

Plus the following which are required:

* timestamp: a timestamp for the input
* duration: the amount of time, in seconds, that the input covers.

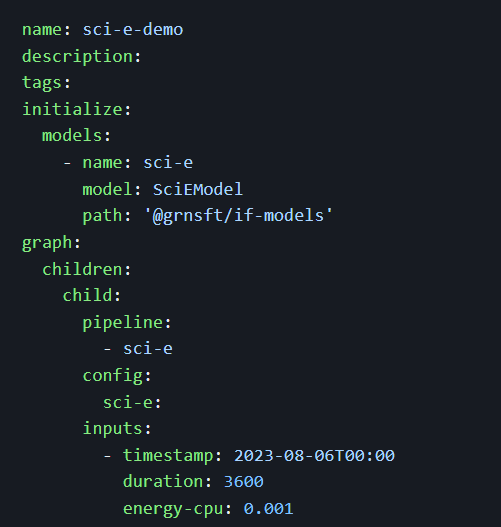
Outputs:

* energy: the sum of all energy components, in kWh

Energy is calculated as the sum of the energy due to CPU usage, energy due to network traffic, energy due to memory and energy due to GPU usage.

So using the following fomula:  
energy = energy-cpu + energy-network + energy-memory + e-gpu

In any model pipeline that includes sci-o, sci-o must be preceded by sci-e. This is because sci-o does not recognize the individual contributions, energy-cpu, energy-network, etc, but expects to find energy. Only sci-e takes individual contributions and returns energy.

Example impl:  


**Cloud Instance Metadata**: it is a model that allows you to determine an instance's physical processor and thermal design power based on its instance name.

**Inputs:**

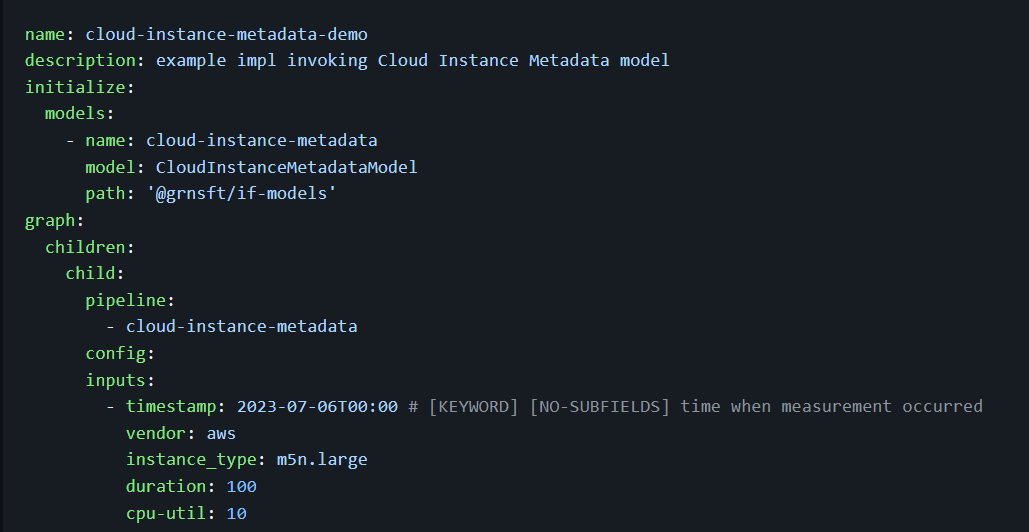
* cloud-vendor: the cloud platform provider, e.g. aws
* cloud-instance-type: the name of the specific instance being used, e.g. m5n.large

**Outputs:**

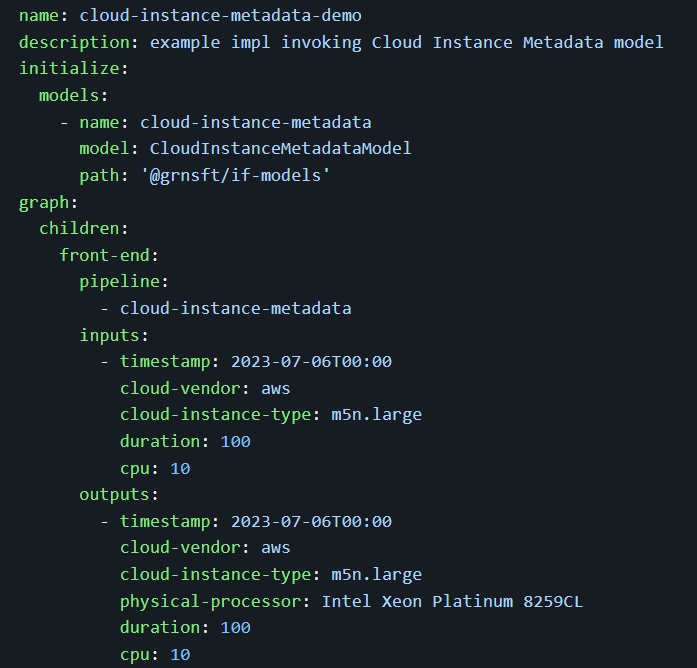
* cloud-instance-type: echo input instance-type
* cloud-vendor: echo input vendor
* physical-processor: physical processor used in the given instance
* vcpus-allocated: number of vCPUs allocated to this instance
* vcpus-total: total number of vCPUs available to this instance

IEF implements this plugin using data from Cloud Carbon Footprint. This allows determination of cpu for type of instance in a cloud and can be invoked as part of a model pipeline defined in an `impl`.  
  
Cloud Instance Metadata currently implements only for 'AWS'.

**Example Impl:**



**Example Ompl:**



**Azure Importer:**

The Azure importer model allows you to provide some basic details about an Azure virtual machine and automatically populate your impl with usage metrics that can then be passed along a model pipeline to calculate energy and carbon impacts.

**Prerequisites:**

Create an Azure VM instance

Provide an identity to access VM metadata and metrics

Create An App registration/service principal for the Azure Importer

Provide IAM access

Add credentials to `.env`

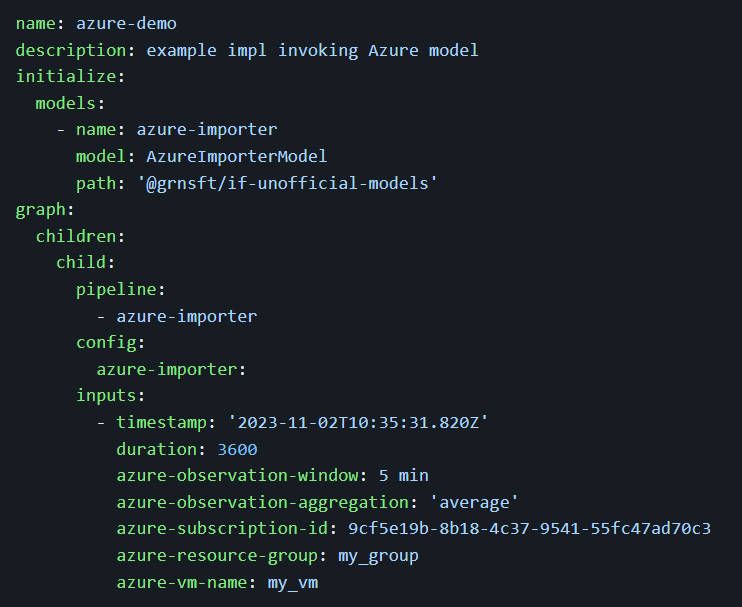
**Inputs:**

* timestamp: An ISO8601 timestamp indicating the start time for your observation period. We work out your timespan by adding duration to this initial start time.
* duration: Number of seconds your observation period should last. We add this number of seconds to timestamp to work out when your observation period should stop.
* azure-observation-window: The time interval between measurements (temporal resolution) as a string with a value and a unit, e.g. 5 mins. The value and unit must be space-separated.
* azure-observation-aggregation: This indicates how you want metrics to be aggregated between each interval. The recommended default is average.
* azure-subscription-id: Your Azure subscription ID, e.g. 9cf5e19b-8b18-4c37-9541-55fc47ad70c3
* azure-resource-group: Your Azure resource group name
* azure-vm-name: Your virtual machine name

**Outputs:**

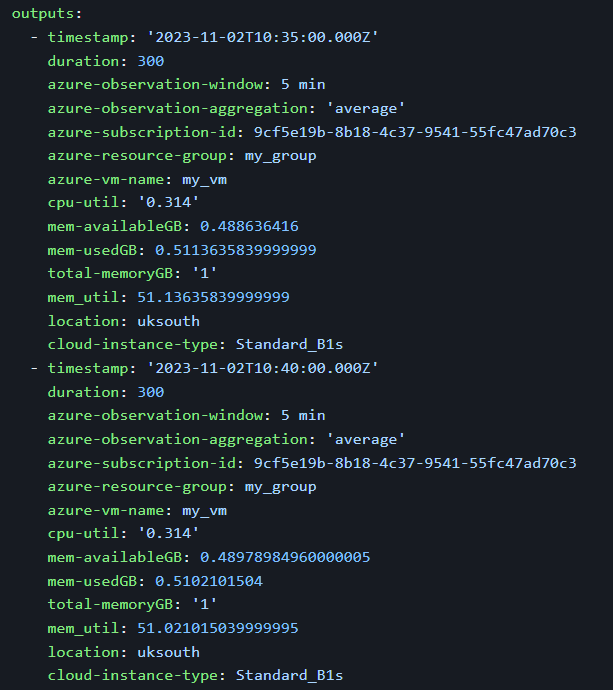
* duration: the per-input duration in seconds, calculated from azure-observation-window
* cpu-util: percentage CPU utilization
* cloud-instance-type: VM instance name
* location: VM region
* mem-availableGB: Amount of memory *not* in use by your application, in GB.
* mem-usedGB: Amount of memory being used by your application, in GB. Calculated as the difference between total-memoryGB and memory-availableGB.
* total-memoryGB: The total memory allocated to your virtual machine, in GB.
* mem-util: memory utilized, expressed as a percentage (memory-usedGB/total-memoryGB \* 100)

**Example Impl:**



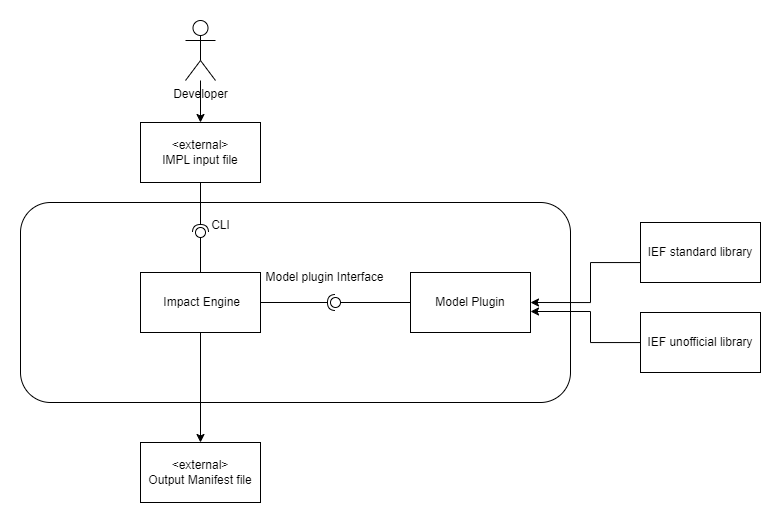
**Example Ompl:**

The Impl will be like the Impl with the following section added at the end



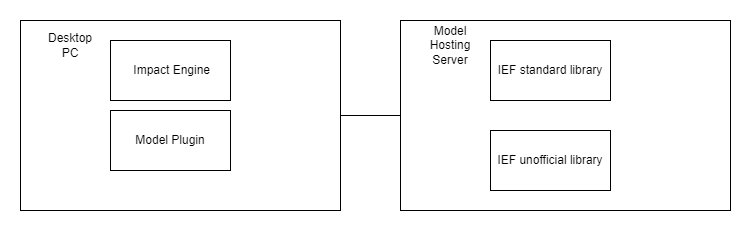
## Functional Views

Functional view of the system as-is:



## Deployment Views

The deployment viewpoint of the system as-is:



## System Qualities

**Security**

Security goal model

|  |  |  |
| --- | --- | --- |
| Assets | Sensitivity | Security Assets and Goals |
| Software configuration input | The configuration, design, and technology of the software involve business secrets. | Confidentially, Integrity, Availability |
| Output Manifest File |  | Confidentially, Integrity, Availability |
|  |  |  |

Security policy

|  |  |  |
| --- | --- | --- |
| Assets | Development Team (Tool users) | GFS |
| Software configuration input | Full access with audit |  |
| Output Manifest File | Full access with audit |  |

**Performance**

|  |  |
| --- | --- |
| Concerns | Description |
| Throughput |  |
| Response time |  |
| Predictability |  |
| Scalability |  |
| Hardware Resource Requirements |  |

*The structure of this section may vary from group to group.*

*We expect all reports to include:*

* *a functional view of the system with clear description of functional elements, their responsibilities and interfaces;*
* *a deployment view if the system is composed of multiple nodes*
* *a discussion of how the architecture supports all important system qualities outlined in Section 2.4.*
* *a discussion of trade-offs and key architectural decisions*

*The section can include other architectural views that are relevant to describe the system and rationale for important architectural decisions.*

*You can select sections from Nick Rozanski and Eoin Woods' architecture description template, available on Moodle, that are appropriate for the analysis of your systems. You can also take sections from the structure of design documents at Google, also available from the course Moodle page.*

*See the marking sheet for information to include in this section.*

# Development and Evaluation Plan

*This section will define your development and evaluation plans for the term 2 project. How will you split the term 2 project into small increments? How will you test and evaluate the success of your project?*

*What you intended to develop in term 2 may be a subset of the requirements and architecture components defined in the previous section. This subset must be a self-contained product. Make sure your description of your development plan is consistent with and make clear references to requirements and architecture elements in the previous sections.*

## Development plan

## Evaluation plan

We plan to evaluate the optimization of our tools on different platforms, such as private cloud, public cloud, bare-metal, virtualized, containerized, mobile, laptops, desktops, embedded, and IoT. By observing the program running on the IF framework and observing the changes in indicators before and after using our optimization tools, we can determine the effectiveness of our optimization.

Firstly, we will identify the indicators for judgement, such as the estimate of energy consumed, TBD.

* Evaluation Criteria
  + Reduction ratio of software carbon emissions before and after optimization
    - Comparing the carbon emissions of the software system before implementing optimization with those after the optimization
  + Usability
    - Evaluating the user-friendliness of the Parameter Optimization model.
    - We can evaluate it by questionnaire.
  + Flexibility
    - How well the model can adapt to different software design scenarios and user requirements.

## Future work (Don’t know if this need)

**Reference**

[1] 2023 COMP0101 Group Project briefs

[2] 2023. Background | Impact Framework. https://if.greensoftware.foundation/background Accessed: December 6, 2023.

[3] 2023. Background | Impact Framework. https://if.greensoftware.foundation/overview Accessed: December 6, 2023.