Provisional Title:
The Big Little Book of CERC Terminology

Current Maintainer: Raman

November 2022

Preamble

This is to serve as a reference book when people use terminology. Changes in terminology from the ones listed in the book must be explicitly highlighted when presenting within the group for the group's sanity and clarity.

This is a living document that has to be maintained by the group. All changes will be performed by the Maintainer assigned to the book. The book is passed from maintainer to maintainer so that there is only one maintainer at any point of time.

Responsibilities of the Document Maintainer

- Ensure timely update of the definitions within the book. (Current update rate: Once a week)
- Add definitions as and when new concepts are added or update definitions when existing concepts are amended.
- Approach user if clarifications are required and try to come up with a general meaning based on discussion with the software and scientific team.

Rules for suggesting edit and updates

1. Follow style of example listed in Chapter 8.
2. For edits in already existing definitions, reference the term and Chapter number in Chapter 8 when suggesting the edit.
3. Give reason for edit in definition and why it makes sense.
4. If adding new term, please give definition and example suggestions, cite if any.
5. For suggesting the edit, either add your name to the suggestion OR add a new command and textcolor in the "main.tex" file so that we can reach you if further clarifications are required. (refer to part titled "Functions

to assign different users with different colors" in "main.tex" file)

Chapter 1

CERC Platform and General Terminology

CERC Platform: The platform is a collection of code and software that when combined together provides concrete orientation on how to manage and collate data from different sources in a coherent manner to form input files for other software (in-house, third-party, proprietary or otherwise). The platform also serves as the interface between the other software used to perform different computations (based on the use-case in hand). The results from those computations are then used to represent/visualise the various flows of material and energy that underpins all human and societal interactions. The platform represents systems and processes to adequate levels of abstraction, depending on the use-case. The platform allows for simulation, optimization, Hardware-in-Loop (HiL, check entry in Chapter 7) tests and forecasting for the various sectors and helps identify retrofit and strategic improvements for different objectives such as GHG emissions reduction, cost reduction etc.

Example: *The CERC platform can be used to collect data about the cement manufacturing plants on the Montréal island, calculate their energy requirements and emissions outputs and identify how to optimally produce the desired amount of cement while also providing other services for Montréal city. The platform can then be used to visualise the results using a graphical interface.*

Central Data Model: A Central Data Model (CDM) is a semi-flexible data structure intended to exchange information between data sources and modeling tools. It must, therefore, homogenize the information fluxes to become independent of the structures and needs of ones and others. Being semi-flexible means that the structure can be extended to include new domains or increase the applicability of the old ones.

Example: *The CERC CDM is designed to create digital twins of cities. Initially, it only modeled buildings but it is being extended step-by-step to include other domains such as utility networks, traffic, waste management, etc.*

HUB: HUB is the name given to the backend consisting of multiple workflows enabling data collection, reconfiguration and analysis for the use-cases of the CERC Platform

Example: *For more details regarding the working of the HUB, refer to the Documentation in Gitlab and the Urban Simulation Platform documentation created by Hadise Rasoulian*

Catalog/Catalogue: A catalog/catalogue is a rigid or semi-rigid data structure, in its simplest form, a file or group of files, that provide information (technical/commercial) regarding components that form a system within any domain. Catalogs are intended to facilitate coherent sharing of data sources between users, while also offering the opportunity to extend the content within the catalogs. The components are listed with relevant details and associated data is tabulated. Each catalog has unique identifying parameters/characteristics that allow them to be unambiguously used within the CERC Platform. These unique identifiers are very case-specific and are sometimes provided by Standard organisations and manufacturers. Also listed could be the dominant/standard component connection configurations and how they may be used to satisfy use-cases/output requirements.

Example: *Heat Pump catalogue should consist of the heat pump models produced, heat pump type, manufacturer name, output temperatures, nominal capacities, typical configurations for the heat pumps (e.g., configurations when used for space heating only, Domestic Hot Water/DHW purposes only, both space heating and DHW, combinations with solar thermal/PV), storage tank data, circulation pump data, compressor type and associated technical data, valve types etc. Building construction archetypes will be organized in a catalog with unique identifying parameters such as Year of Construction, Location and Building Function.*

Factory: A factory is any pipeline (1) created within the platform to either collect data together and form input data or be used for dispensing the outputs and/or transforming and giving files of different formats. Factories broadly fall within three categories: import, export and ‘catalog expose’ factories. The pipelines used to collect data together and form the necessary inputs are the import factories, the pipelines used for providing the results and creating custom formats for various workflows form the export factories while the ”catalog expose” factories provide a homogeneous interface (with or across the different?) the data sources. At present, the following factories have been developed/are under development: Geometry, Usage, Sensors, Weather, Energy Systems, Construction, Life Cycle Assessment (LCA) and Customised imports.

Example: *The building geometry files are called and transformed into the appropriate structure for simulation by the geometry imports factory. The collected geometry can then be transformed into another file format using the geometry export factory. Similarly, input files for schedules/usage falls under the usage factory.*

Library: A library is a collection of related modules (defined as per the rules of the language, the library is coded in). It contains code that can be used repeatedly in different programs and for different use-cases. Libraries may include configuration data, documentation, help data, message templates or examples. Library code is organized in such a way that it can be used by multiple programs that have no connection to each other, while code that is part of a program is organized to be used only within that one program.

Example: *Scipy[1] and Pandas[2] are examples of Python Libraries.*

Frontend: The layer of abstraction that connects the user to the backend of the hardware/platform/environment.

Example: *The graphical interface of a smartphone/tablet is the frontend of the phone/tablet. A web browser acts as a frontend to the data servers holding the information on the different services that you require from the different websites.*

Backend: The part of the platform/hardware/software that houses the computational logic, data requisition and storage and the different abstractions of the systems and/or pilot plants/emulation of parts of the system.

Example: *The hardware of a smartphone/tablet is one of the many backends for the smartphone/tablet. The data servers hosting the information and addresses/URIs of the websites form the backend for the world-wide web.*

Use-case: A use-case refers to a problem/scenario or set of scenarios that provide information on the level of detail required from the abstractions created, the time horizons and time steps required and the boundaries of the subject under study. The use-case also decide the tools, softwares and methodology required to tackle the problem.

Example: *Energy Management of a building/systems and its systems is one of the many use-cases that can be created from a group of buildings and knowledge of the systems within those buildings. System sizing and retrofit is another use-case.*

Framework: The Cambridge dictionary defines a framework as both a supporting structure around which something can be built and a system of rules, ideas, or beliefs that is used to plan or decide something [3]. With reference to the Platform, both apply, it is essentially a standard set of rules that guide the development of applications, functions and tools which can be used for different use-cases. This allows the platform to be extensible and adaptable.

Example: *Django and Flask are two examples of Python web frameworks.*

Workflow: A Workflow is a sequence of operations/tasks that ensures completion of any process from start to finish. This sequence is usually well-organised and repeatable, but, will yield different results and will require different interactions for different processes. Workflows are usually complex, have non-computational elements, can loop and can be non-linear.

Example: *Modern-day assembly lines are well-defined workflows that transform raw materials to finished products within the industrial setting. A factory, both in an industrial setting and with reference to the platform consists of many workflows.*

Pipeline: A pipeline is a chain of processing functions, classes and objects that is usually linear and does not loop. It allows for easy tracking and completion of tasks within parts of a process that is envisioned within a workflow. Rather simplistically put, a pipeline is a subset of a workflow or a codified representation of parts of a workflow, that runs without any intervention and within a fraction of a time of an entire workflow to provide one output.

Example:

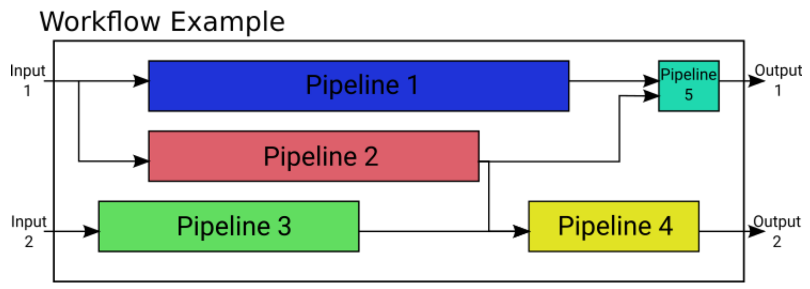

Figure 1.1: Workflow and pipeline example

Simulation: Simulation, according to [4], is “the process of designing a model (using mathematical and logical assumptions) of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system”. These system representations/models are predominantly used to calculate system behaviour over a period of time and can be either static or dynamic. For further details and more detailed explanation of terminology, refer to Section 1.1 or relevant sections in subsequent chapters (Cross-connections to be added).

Example: *Input Simulation Example*

Optimization: In a vernacular sense, optimization is just a method of finding the best possible solution in any situation where there are multiple options available. In mathematics, it is the process by which we can find the best solution for problems where the calculation of the best solution is made difficult due to the large number of constraints, changes in the conditions with time and the large number of solutions which one can’t possibly calculate by hand in a small amount of time. There are different kinds of optimization, each with their own strengths and weaknesses. Optimization is usually chosen by the desired use-case. For further details and more detailed explanation of certain kinds of optimization and associated terminology, refer to Section 1.2 or relevant sections in subsequent chapters (Cross-connections to be added).

Example: *Input Optimization Example*

Model: A model is just a simplified representation or abstraction of how any object (called system) functions as devised a designer (you or me) of the system. This can be represented either by creating a scaled-down version and/or logical and/or numerical statements. A model can also be a representation that defines the rules and relationships between the logical and/or numerical statements and the prerequisite information that is needed to mimic the behaviour of the system to the level of accuracy desired by the designer of the model. Within the platform, the term model is used predominantly in two forms: data models (ontological models) and computational models.

Example: *Any toy train network is a model of the real train network. The model accuracy of the toy train network will vary depending on the designer's desire for detail and accuracy to the real-life system, associated rules and calculations.*

Data model: A data model is a representation of either a whole system or parts of it to communicate connections between data points and associated structures (related to ontological and semantic modelling). The goal is to illustrate the types of data used and stored within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes. Data models can be of the following types: Logical (big picture view of what it represents, contains and some relationships), Conceptual(Detailed view of relationships between parts and associated datatypes) or Physical (how and where is the data physically stored).

Example: *That*

Computational model: Computational models are models created to simulate, optimize and study complex systems and phenomena using a mix of mathematics, physical sciences and computer science. Computational models are usually solved using computers owing to model complexity. Computational models attempt to represent the system/behaviour of interest as a set of mathematical equations (partial differential or otherwise). This representation might require high levels of detail depending on the degree of accuracy desired or might utilise approximation methods and correlations to sacrifice accuracy for speed of computation.

Example: *Simulink, INSEL, EnergyPlus, Labview, FLUENT and other simulation softwares utilise computational models to represent fluid flows, energy flows, energy systems and buildings among other systems and phenomena.*

Machine Learning:

1.1 Simulation

Simulation Software:

Dynamic Simulation: Mathematical models of such systems would be naturally viewed as dynamic in the sense that they evolve over time and therefore incorporate time. A dynamic model includes time in the model. The word dynamic is derived from the Greek word *dynamis*, meaning force and power, with dynamics being the (time-dependent) interplay between forces. Time can be included explicitly as a variable in a mathematical formula or be present indirectly, for example through the time derivative of a variable or as events occurring at certain points in time

Example: *Dynamic simulations can utilise discrete or continuous-time steps*

Static Simulation: A static model can be defined without involving time, where the word static is derived from the Greek word *statikos*, meaning something that creates equilibrium. Static models are often used to describe systems in steady-state or equilibrium situations, where the output does not change if the input is the same. However, static models can display a rather dynamic behavior when fed with dynamic input signals

Example:

Design Period:

Models:

1.2 Optimization

Solvers:

Models:

1.3 Geographical Assets

NRCAN Vector and Raster assets: `https://ftp.geogratis.gc.ca/pub/nrcan_rncan/`

Example:

1.4 Visualisation

1.5 Programming

IDE: Acronym for Integrated Development Environment (IDE). IDEs are special tools/software that allow programmers to develop new software. Different IDEs will have different features integrated into them and cater to different coding philosophies. As a result, there might be different flavours. Some of the usual features of IDEs are:

- Programming language syntax check.
- Programming language syntax highlight.
- Code hints.
- Code auto-completion.
- Easy compilation/execution.
- Version control integration

Example: *PyCharm is an example of an IDE developed by JetBrains for Python*

Coding:

Classes:

Example:

Objects:

Example:

Inheritance:

Example:

Unified Modeling Language:

Example:

Visual Programming/Coding: It's referred to a coding style where the program isn't wrote by using written expressions with a specific syntax, but combining visual elements connected in a specific way

Example:

1.6 Gamification

Chapter 2

Buildings

Urban Building Energy Modelling: UBEM or Urban Energy Building Modelling is used

2.1 Weather

EPW(EnergyPlus Weather File): EnergyPlus uses weather files of a certain format.

Example: *EPW file for Montreal is used as an input for the EnergyPlus Simulations*

World Meteorological Organization (WMO): Since weather systems and climatic conditions extend beyond international boundaries, it is necessary to exchange weather information freely throughout the world. This requires coordination and standardization of practices and procedures for efficient exchange of weather transmissions. To promote these services and to further the application of meteorology to aviation, shipping, agriculture and other human activities, the World Meteorological Organization was established by the United Nations in 1951. Its weather reporting codes are called International Codes.

Example: *WMO ID of Montreal International Airport is 716270, corresponding Meteorological Service of Canada Climate ID for the Airport is 7025251*

CWEC (Canadian Weather Year for Energy Calculation):

TDY (Typical Downsized Year):

TMY (Typical Meteorological Year):

TRY (Typical Reference Year):

HVAC: Heating, Ventilation and Air Conditioning Systems provide the heating and cooling requirements for each building. This may consist of a mix of different systems varying depending on location and standardised practices. For further details on the individual systems, refer the appropriate terms in Chapter 3

Example:

Urban Heat Islands (UHI):

Example:

:

Example:

BCVTB (Building Control Virtual Test Bed): The Building Control Virtual Test Bed is a project run and maintained by the Lawrence Berkeley National Laboratory for the purpose of testing control and energy management purposes in Hardware-in-Loop (HiL) settings. BCVTB uses Ptolemy, Energy-Plus, OpenModelica and other related softwares. The BCVTB environment can be found here: `https://simulationresearch.lbl.gov/bcvtb/FrontPage`

Example:

Chapter 3

Energy Systems

Carbon Sequestration Systems:

Example:

Heat Pumps:

Example:

Cogeneration/Polygeneration:

Example:

Chapter 4

Electric Grid and Energy Management

4.1 Energy Management

Flexibility:

Example:

Chapter 5

Transport

OD matrix:

Example:

Chapter 6

Circular Economy

6.1 Green Roofing

6.2 Waste Management

Black Water:

Example:

Chapter 7

Experimental Setups and Internet-of-Things

ADC Pin:

Hardware-in-Loop:

Testbed:

Example:

Pilot Plant:

Example:

FPGA: A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing

Example:

7.1 Internet-of-Things (IoT)

Internet-of-Things (IoT: A buzzword/keyword, much talked about, concerning networks of sensors, devices and objects with access to the wider web. An IoT network could be developed for a building, a neighbourhood, a city or across many cities. Currently, there is little consensus on what it actually implies. So, each time one uses IoT, the author/contributor has to define it for their use-case.

Example:

Standards:

Example:

Protocols:

Chapter 8

Edit, Update Requests and New terms

Example usage

SUGGESTED TERM/TERM DEFINITION CHANGE: INSERT NEW DEFINITION HERE

Example: *EXAMPLE OF TERM. You can also add images if you feel like to clarify your definition here*

New terms for definition and definition correction

Bibliography

[1] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," Nature Methods, vol. 17, pp. 261–272, 2020.

[2] W. McKinney, "Data Structures for Statistical Computing in Python," in Proceedings of the 9th Python in Science Conference, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[3] C. Dictionary, "Framework," in Cambridge dictionary. Cambridge University Press, 2022. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/framework

[4] R. E. Shannon, Systems simulation; the art and science. Prentice-Hall, 1975.

Index