



1. Motivation and Outline

Original text by

Prof. Antonello Monti (+ AE)

Traditionally, modeling and simulation has been conducted within **a single physical domain type**. Electrical engineers used to develop electrical systems primarily focusing on **grid modeling** and mostly neglecting other interacting domains. One of the significant exception to this rule has been a simplified representation of the rotating mass for stability analysis (Kundur, Balu, and Lauby 1994; Ulbig, Borsche, and Andersson 2014).

In the last decade though, we have experienced a growing interest towards **multi-physics design** in many different areas of applications. Two concrete examples of this type are the avionics industry and ship industry in relation to programs such as More Electric Aircraft (Hafez and Forsyth 2009; Rahrovi and Ehsani 2019) and All Electric Ship (Sulligoi, Vicenzutti, and Menis 2016; Thongam et al. 2013). The development of such projects has shown the limitation of the single-domain approach and has paved the way to comprehensive approaches for multi-physical modeling and simulation.

One response to the problem has been the development of **co-simulation** interfaces and standards (Blochwitz and al. 2011; Gomes et al. 2018; Müller et al. 2016). Co-simulation offers the user the possibility to operate in a domain-specific environment and to relegate the integration of other simulation platforms to be mostly a software challenge.

While this approach has some clear advantages from user perspective, first of all the possibility to continue operating with the same tools also in the new operating conditions, co-simulation tends to have a significant computation and implementation overhead given by the co-presence of more than one simulation platform. Nevertheless, many tools exist simplifying the co-simulation implementation overhead, e.g. (Ochel and al. 2019; Widl et al. 2013).

Furthermore, it is quite easy to face versioning issue created by the fact that the project as a whole is stored in multiple files. There are so many technical difficulties. Detailed illustration goes beyond this text.

Proposition 1.1 In summary, one day co-simulation will be most effective if it converges to the state that a modeler is neither aware nor technically caring that his descriptive models are executed via co-simulation.

Consequently there is a growing interest towards **multi-physical modeling languages**. Multi-domain languages provide an interesting solution, majorly because they are languages and not simulation platforms.

This means that the modeling effort and the solution effort are clearly separated. This separation brings an incredible benefit from the user point of view that is able to migrate from one simulation tool to another without repeating the modeling effort. This is because the same model specification is used among the different tools.

Proposition 1.2 Converting from a tool user to a model designer via a well-designed modeling language extends the horizon of realizable applications to an extent that is beyond imagination.

There are many simulation languages of this type available s.a. VHDL-AMS(Ashenden, Peterson, and Teegarden 2003), gPROMS (*Process Systems Enterprise*) and **Modelica** (Elmqvist 2014). The last language has been developed as an initiation for a standard modeling language used by a large modeling community in many modeling domains. This has resulted in an open-source non-property specification language continuously maintained, supported and progressed both from academia and industry.

Outline of this book

The outline of the book is structured as follows.

In Part I we first review **the major challenges** in power system modeling and simulation. These challenges impose **new requirements** for modeling and simulation tools aiming at **modern power system modeling applications**. In order to appreciate the evolution process of such modern modeling languages, we reveal how the accumulation of progressive


milestones along many decades led to the concepts on which some current state of the art of modeling languages are based on.

In Part II, we dive into the Modelica language, giving a non-familiar reader though rather a compact overview of the language but hopefully generous enough to recognize its potentials in the scope of power system modeling applications. This demonstration is based on one of the existing and elegant open-source libraries. This particular library provides the opportunity to demonstrate many syntactically significant features of the Modelica language.

In Part III we overview current state of the Modelica language. Particularly, we summarize

- **the language specification**, the underlying process of its development through the Modelica association
- **the Modelica Standard Library** and some of its major components significant for power system modeling,
- **the functional mockup interface** including related projects
- as well as other significant related topics that may matter the reader

Furthermore, reflecting our humble struggle to eliminate the thirst of a knowledge seeker, we provide a comprehensive list of **open-source Modelica libraries** in the field of (and in many areas related to) power systems. The list includes libraries in the field of energy in buildings and cities as well as other fundamental methodological libraries, e.g. for generating statistical distributions or for imitating a hybrid paradigm.

 Additionally, another list of **commercial Modelica technologies** is aimed. However, tool vendors, library developers and interested parties need to report us their relevant technologies and provide us adequate text, references and corresponding websites.

Finally, since power systems models are naturally large-scale, in Part IV we demonstrate current **scalability challenges** facing Modelica-based tools. These are addressed together with recently and actually conducted research efforts for overcoming these scalability challenges. We summarize with **a list of benefits the Modelica language** can offer in the context of modern power system modeling applications. We overview **a list of potential extensions** to the Modelica language in order to enable further capabilities, e.g. partial differential equations or prototyping of simulation-optimization problems.