**DOCTORAL THESIS**

**STOCKHOLM, SWEDEN 2019**

# SEMANTIC INFORMATION AND PHYSICAL MULTI-DOMAIN MODELING AND SIMULATION FOR POWER SYSTEMS

FRANCISCO JOSÉ GÓMEZ LÓPEZ

*KTH Royal Institute of Technology*
*Department of Electric Power and Energy Systems*
*School of Electrical Engineering and Computer Science*
*SE-10044 Stockholm*
*SWEDEN*

TRITA-EECS-AVL-2019:25
ISBN: 978-91-7873-138-1

*Akademisk avhandling som med tillstånd av Kungliga Tekniska Högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i elkraftteknik, fredagen den 12 april 2019, klockan 15.00, i sal H1, Teknikringen 31, Kungliga Tekniska Högskolan, Stockholm*



*Printed by: Universitetsservice US-AB*

# Abstract

There are different reasons for combining different modeling languages and simulation languages: Exchange of more detailed information about power network components, their parameters and, most importantly, mathematical equations describing their behavior and the exchange of a mathematical description, using equation-based languages (e.g.: Modelica), allows models to be detached from the mathematical solver. This leads to the development of new APIs within software tools, which can handle standardized modeling language used for model implementation. Furthermore, the mathematical description of models and the integration of new simulation standards, such as the FMI, could help avoiding ambiguities on how power system models are implemented, by providing additional means for the exchange of the complete description of models or parts of a model between software tools.

The aim of this thesis is to provide a new approach for the development of power systems modeling and simulation software tools. The thesis is focused on proposing new methods, based on available information and simulation standards for the exchange, modeling, and simulation of power systems dynamic models; and to show a proof of concept of the feasibility of the proposed methods. To this aims, the Common Information Model (CIM) for the modeling and exchange of power system information is studied. Furthermore, the equation-based language Modelica is described and proposed with the aim of complementing the use of the CIM for the modeling and simulation of those dynamics models.

The application of these standards lead to a different view on the modeling and simulation of power dynamic network models. The conventional view is that of black box modeling. The implementation of network model components is strongly connected to the simulation software tool used for steady-state and dynamics calculations. Thus, a modeler or test engineer only has access to the parameters of a model and relies on the software capabilities to calculate the states and the behavior of that model. This thesis proposes a different view per the application of the white box modeling and simulation concept: full detail and transparency on the development of a mathematical description of power system components and discrete events. Moreover, the combination of information standards with equation-based standards to produce network models allows full access and manipulation of the complete model details. Finally, transparency regarding the implementation of software tools can support either information-based, equations-based languages or simulation standards, which are suitable for simulation of dynamic network models.

# Sammanfattning

Det finns olika anledningar att kombinera olika modelleringsspråk och simuleringsspråk: Utbyte av mer detaljerad information om nätverksdelskomponenter, deras parametrar och, viktigast av allt, matematiska ekvationer som beskriver deras beteende och utbyte av en matematisk beskrivning med hjälp av ekvationsbaserade språk (e.g.: Modelica), tillåter modeller att lösas från den matematiska lösaren. Detta leder till utvecklingen av nya API inom mjukvaruverktyg, som kan hantera standardiserat modelleringsspråk som används för modellimplementering. Dessutom kan den matematiska beskrivningen av modeller och integrationen av nya simuleringsstandarder, såsom FMI, bidra till att undvika tvetydigheter om hur kraftsystem modeller implementeras genom att tillhandahålla ytterligare medel för utbyte av den fullständiga beskrivningen av modeller eller delar av en modell mellan mjukvaruverktyg.

Syftet med denna avhandling är att tillhandahålla ett nytt tillvägagångssätt för utvecklingen av systemmodellering och simuleringsprogramvaruverktyg. Avhandlingen är inriktad på att föreslå nya metoder, baserade på tillgänglig information och simuleringsstandarder för utbyte, modellering och simulering av dynamiska modeller för kraftsystem. och att visa ett bevis på konceptet om genomförbarheten av de föreslagna metoderna. För detta syfte studeras den gemensamma informationsmodellen (CIM) för modellering och utbyte av kraftsysteminformation. Dessutom beskrivs och föreslås det ekvationsbaserade språket Modelica med syfte att komplettera användningen av CIM för modellering och simulering av dessa dynamikmodeller.

Tillämpningen av dessa standarder leder till en annan syn på modellering och simulering av kraftdynamiska nätverksmodeller. Den konventionella uppfattningen är den för svart boxmodellering. Implementeringen av nätverksmodellkomponenter är starkt kopplad till simuleringsprogramverktyget som används för beräkning av steady-state och dynamik. Således har en modellerare eller testingenjör endast tillgång till parametrarna i en modell och bygger på mjukvarans förmåga att beräkna tillstånden och beteendet hos den modellen. Denna avhandling föreslår en annan uppfattning per tillämpningen av den vita boxens modellering och simuleringskoncept: fullständig detalj och insyn i utvecklingen av en matematisk beskrivning av kraftsystemets komponenter och diskreta händelser. Dessutom möjliggör kombinationen av informationsstandarder med ekvationsbaserade standarder för att producera nätverksmodeller fullständig åtkomst och manipulation av de fullständiga modelldetaljerna. Slutligen kan insyn i implementeringen av programvaruverktyg stödja. antingen informationsbaserade, ekvationsbaserade språk eller simuleringsstandarder som är lämpliga för simulering av dynamiska nätverksmodeller.

# ACKNOWLEDGEMENTS

Words cannot express those feelings towards all the good people that have met during this journey. For this reason, I prefer not to mention anybody. While I am writing these letters, lots of names are coming and leaving so I prefer to mention no one, because I wouldn't like to forget any of them.

However, I would like to express my deepest and sincere gratitude to my main supervisor. Luigi Vanfretti gave me the opportunity to begin with this enormous adventure outside my home country. I really appreciate his constant encouragement, support and the moments we have been sharing together, traveling, working or just enjoying moments of relax conversation. One of my few objectives in live has been accomplished thanks to this door that he opened for me.

Second, I would like to express my gratitude to my second supervisor, Svein Harald Olsen. Having an advisor from the industry has been a tremendous help to get to know about those decisions taken by power system stakeholders, and how these decisions can affect certain research areas. And particularly useful has been his support to make possible having my research being introduced to the industry.

During this five year of the PhD I had the chance to meet very good persons that have become an important part of this journey, and with who I had the privilege of sharing many good memories and experiences, travelling around the world together or just sharing some BBQ time. I have been very fortunate to meet people from almost all over the world. This is why I would like to thank you to all the office mates who I share daily joy and suffering. Thanks to those who have become closer friends, some of them who started as students and then become department mates and some others that started the PhD journey before me and after me. Thanks to those who have been around for short time, either for an academic stay or as part of our research group during their studies. From everybody I keep a really beautiful and unforgettable memory, and all of you have become very important members of my life.

This journey has not been easy at all. I had to leave behind, as many others, that comfort zone which is the life at my home town. I had to leave behind many friends and family that I love, so to discover new cultures and ways of living and working. Finally, for that reason, I would like to say thanks to my parents and siblings, and the old friends from my home university, from basketball and from home town, for their patience, encouragement and constant support… and their visits from time to time.

*“Often is more important*
*to notice that you are wrong,*
*that it is to proof that you are right”*
- Jordan B. Peterson

# Contents

# Chapter 1

# Introduction

## 1.1 Regulations of Power System Modeling, Simulation, and Standards

In 2006, a large system disturbance affected major parts of Europe [1]. The coordination of Transmission System Operators (TSO) in planning and operation procedures such as system security, capacity calculation, and outage planning was not sufficient to avoid that event. Consequently, the European Regulators' Group for Electricity and Gas (ERGEG) defined a set of recommendations stating the need for compliance and consistency and highlighting the importance of the interdependencies and information exchange of the national implementations of grid models, referring to both static and dynamic information upon which these models are implemented.

The adoption of regulation EC 714/2009 for cross-border exchanges in the electricity network to ensure coordination of "data exchange and settlement rules, network security and reliability rules, interoperability rules, and transparency rules" [1], by the The European Network of Transmission System Operators for Electricity (ENTSO-E), and the continuous enhancement of computational technologies such as real-time performance, emphasizes the development of complete power system models. To comply with these rules, stakeholders have developed software tools as a primary means to perform analysis and coordination tasks. The International Electro-Technical Commission (IEC) Technical Committee 57 (TC57) Working Group 13 and the Electrical Power Research Institute (EPRI) have been developing the Common Information Model (CIM) to provide standard modeling semantics for power systems information exchange [3] [4].

Within this context, the ENTSO-E has made large efforts to adopt the CIM to improve the means for information exchange between TSOs. The ENTSO-E has adopted different IEC CIM profiles to comply with the regulation and mandates [5]; thus, the Common Grid Model Exchange Standard (CGMES) [6] was approved. Conformity of modeling, simulation, and analysis tools to CGMES is carried out through Interoperability (IOP) tests [7]. The CGMES reflects current TSO requirements for accurate modeling of the ENTSO-E area for power flow, short circuit computations, and dynamic simulations.

Simulation models representing the dynamic behavior of a power system are generating interest at the European level. The ENTSO-E has released a series of requirements for the implementation of network codes which require dynamic models for simulations in trade-off analysis for the connection of different energy sources to the grid [8]. Therefore, standardization processes and collaborative tools to enable stakeholder interactions are required to improve the two above (models and regulations, i.e. network codes) as well as to adapt to new requirements while ensuring interoperability, security, and privacy [1][5].

## 1.2 General View of Software Requirements for Modeling and Simulation Tools

Standardization processes and collaborative tools are required to improve modeling and simulation of electrical engineering systems. The implementation of new network codes at the European level have led to the development of a Common Grid Modeling Exchange Standard (CGMES). Moreover, it requires the development of dynamic models

which expose their physical behavior in simulations used in trade-off analyses. These analyses are of particular importance due to the increasing number of renewable variable energy sources connected to the grid. The development of such models leads to the development of new requirements for power system studies, while ensuring the interoperability, security, and privacy of the models. The use of open standards for modeling and simulation could allow the export of all necessary model information and physical behavior to guarantee consistency between software vendors and stakeholders.

In a specific example, a technical regulation for grid connections stipulates that "… the transmission system operator must maintain and expand the simulation models continuously…." This technical regulation also states that "The simulation model must be supplied in the form of a block diagram which mainly by means of logical and mathematical functions – primarily transfer functions in the Laplace / z_domain – describes the properties of the PV power plant" [9]. These requirements indirectly state that software vendors must embrace the utilization of new computer programming techniques that offer better reusability of code and allow information exchange of steady-state and dynamic behaviors.

However, traditional power system simulation tools only partly comply with such requirements because they do not allow the export of all the necessary model information to guarantee consistency (i.e., an explicit realization using a strict mathematical representation of the model). Furthermore, power system models based on transfer functions, which are usually a rough approximation of reality – specially lacking proper non-linearity representation – could be replaced by or combined with other domains' mathematical representations. Thus, power system multi-domain models could provide better representation of real system behaviors, which in-turn gives better information that can help in the decision-making process where simulation is needed.

Most of the requirements in relation to power systems applications and analysis methodologies are developed with natural language. However, the ambiguity of natural language makes it difficult to extract specific information requirements, and it is up to the engineer to interpret those requirements and their relationships, leaving an open door to ambiguity [10]. Industry practices are more concerned with the direct implementation of technical requirements, mainly because issues concerning business objectives and productivity incentives [11]. The development of legacy power systems simulation tools is mostly guided by direct prerequisites from the tool's users, e.g., very few specialists at TSOs and by proprietary-tool vendors [7] using a traditional cathedral-like method [12].

During the course of the ITEA 3 OpenCPS EU Project [13], Requirements Engineering (RE) for power systems models has been exploited within the design and implementation of physical systems [13], with the use of the Systems Modeling Language (SysML). Current methods from RE allow the definition of Modeling & Simulation (M&S) functionalities in isolation from the development of the model. In the power systems domain, such functionalities have been mostly studied as part of the same model or application developed, i.e., the development of different network representations as attached to the development of M&S tools.

## 1.3 Other Engineering Domains in Power Systems Modeling, Multi-domain Modeling and Simulation

The efforts to achieve a more sustainable energy supply imply a transition to an overarching grid architecture that allows acceptable levels of reliability and security and affordable prices [15]. Due to the growing adoption of Intermittent Energy Resources (IER) for power generation, the challenges that their intermittent nature imposes on power grid

operation requires special attention. Reliable operation of power systems with high penetration of IERs depends, among other aspects, on more trustable forecasts and accurate models that can be used by various power system analysis tools, particularly those that involve power systems simulations. On the one hand, the variability of wind and solar power can be modelled as slow and fast fluctuations. On the other hand, the power increase/reduction required to deal with these power fluctuations can be achieved by means of ramping sources like gas natural turbines and their fast frequency regulation capabilities; hence, the operational flexibility of gas power plants makes them a good backup for IERs, as seen in the United States over the last decade.

Ensuring the correctness of the more complex physical models of gas turbines relies on the availability of data from the manufacturers; thus, it is reasonable to expect that they can provide such modeling data [16]. Furthermore, the interoperability or exchange of data between the Smart Grid Architecture Model (SGAM) layers, domains, and zones can be used in explaining the complex interaction of different engineering domains. However, the information used in the upper layers of the architecture does not reflect the physical systems of the Component Layer in detail. The information extracted from the analysis of the data can be modified by assumptions regarding how the behavior of the physical components and their models are simplified.

The models for those cyber-physical systems can be developed via the combination of different semantic modeling languages with the aim of providing a better understanding and interoperability of the technical layers within the SGAM. Therefore, an additional standard is considered for the new methodology for modeling and simulation. The information and specifications provided by the ISO 15926 standard for modeling power plant operation [17] are studied to support the CIM in modeling the dynamic behavior of the power system components. Basic main features have been gathered from this standard so they can be utilized for enhancing the different context layers wherein the CIM is classified. In Chapter 4, this new standard is described, and some semantics and syntax details are proposed in conjunction with the CIM.

## 1.4 Adoption of Modelica for Power System Dynamics Modeling

Efficient information exchange between TSOs, Distribution Systems Operators (DSO), and generation companies is required for network planning and power systems operations: e.g., steady-state analysis, dispatch, etc. These functions demand a high degree of coordination and consistency in data exchanges, which can be significantly streamlined through a common data exchange standard. European regulations on information exchange have created new requirements for analysis tools, the main one being the adoption of the IEC Common Information Model (CIM) that may help interoperability across applications [1].

While the IEC CIM provides a solid basis for information exchange, it does not provide any guarantee a model's simulated response. The IEC CIM for Dynamic defines a relevant standard model with parameters for exchange. This implies that the physical dynamic behavior is only represented in a block diagram rather than in a strict mathematical form (equations). In other words, the lack of a strict mathematical description defining each of the power system models is not exchanged (only parameters and pictorial diagrams). Thus, interoperability can only be unassailable though a comparison of several implementation results.

Open standard modeling languages for simulation, such as Modelica [18], can provide the mathematical description of physical systems, which is attractive to exchange user-defined models. The Modelica language can preserve the consistency of the behavior

exposed by power system dynamic models when simulated in different tools [19]. The authors' previous work in [20] proposes a proof of concept regarding a Model Execution Engine leveraging Modelica tools, showing that commercial Modelica compilers, such as Dymola, and open-source Modelica compilers, such as OpenModelica, can be used for power systems model simulation and analysis. Other tools such as the ModelicaML [21] help to link SysML-based models to derive Modelica code, allowing the derivation of equation-based models of entire systems from the mathematical description of each component and their interconnections (system structure).

From the above, it becomes evident that the means to harmonize and interface the different information modeling and physical modeling computer languages can be very attractive in supporting power system model exchange and dynamic applications. The new methodology presented in this thesis is the application of Model-to-Model (M2M) transformations from CIM to Modelica, which will be described in more detail in Chapter 5. The M2M method presented herein allows one to work directly with the information and mathematical descriptions and computer implementation of dynamic models, independent from specific analysis tools. The M2M method proposed requires mapping between CIM/UML and the Modelica language, shaping Modelica models of physical power systems for dynamic simulations.

## 1.5 Engine-Based Software Architecture for Different Power Systems Analysis Workflows

According to [22], modern simulation software should be conceived of as working with discrete-event and continuous systems. Such software will consider constructs with which to build models, a simulation engine to calculate a model's dynamic trajectories and means for control and observation of the simulation as it progresses. Moreover, a simulator of continuous systems approximates the solution to a set of differential equations, the choice of integration method, and the requirements for accuracy and precision. A discrete-event simulation executes events scheduled by its components in the order of their event times. The simulation engine produces dynamic behavior from an assemblage of components. For continuous-time simulation, software development is easier than for discrete-event simulation. Continuous-time simulation deals exclusively with relationships between real variables, allowing for simple programming when the equations are in proper form and suitable numerical algorithms are available.

However, the software should be built keeping a clear separation between the model and the solution algorithm. When talking about Wide-Area Power System (WAPS) simulation, it is important to clearly define which data and how such data should be exchanged between components to produce the general dynamic behavior. In many proprietary power systems, simulation software is a common practice to encourage the integration of a solution algorithm strongly connected to their own models. This approach has several drawbacks, such as limitations and extensibility and portability of the model for execution in other simulation engines. In contrast, the equation-based object-oriented language, such as Modelica, offers a strict separation between the model and the solver, as well as model portability thanks to its standardized language definition [18]. This allows to develop power systems simulation tools with the ability to add new algorithms to existing model structures without the need of modifying the models.

These software characteristics were studied the development of the FP7 iTESLA Project [23]. One of the outcomes of this project was the provision of a tool for coordinating power system operational security between different TSOs and for providing recommendations on the use of renewable generation in defense plans. The coordination of different TSOs requires providing support for a wide range of relevant formats. Data models used in

simulation may be different, may use different data formats, and may be used for on-line and for off-line simulations. However, the data may be the same. Even though there are commercial, well-stablished software packages, such as PSS/E, Power Factory, Neplan or Eurostag within the power system community, efforts made in the iTESLA Project show that there is a lot of value in the application of open standards for modeling and simulation. The deeper exploitation of this standard lead to a new modeling methodology and development of new software architectures for analysis and simulation.

The design of the software architecture proposed in this thesis explores the different parts of the conventional methodology for systems modeling and simulation, but with the application of the already-mentioned standards to amplify the conventional means of analyzing power systems. The new methodology described within this thesis considers the combination of CIM information models with Modelica physical models to create a complete network model. Furthermore, the methodology also considers the choice between different Modelica compilers, as well as the choice of different mathematical solvers to run the dynamic simulations. Chapter 6 will present the design for the implementation of the software architecture based on engines, supporting the methodological requirements designed in Chapter 3.

## 1.6 Contributions & Outline

The standards for information modeling, equation-based modeling, and simulation, mentioned previously, will be described in more detail in the following chapters. These standards have the common characteristic of allowing the application of a white box modeling and simulation perspective. The thesis will not use conventional mathematical foundations for explaining model and software development. Rather, the research is focused on the use of Model Driven Software Engineer (MDSE) principles to describe a new proof of concept for power system studies. This monograph aims to describe different stages for this proof of concept: from the definition of requirements for new software tools, to a prototype implementation of those requirements. This proof of concept justifies the choice of the standards and explains how they can be combined to develop different workflows.

In this section, the main contributions and outline of the thesis are summarized including the publications that each chapter is based upon.

### 1.6.1 Chapter 2: State of the art

This chapter introduces the background and the scope of this thesis. It offers a description of the technologies related with the regulations mentioned in section 1.2, the information standards and Modeling and Simulation (M&S) semantics for power system dynamics and other computational-based technologies and methodologies applied in this thesis. This chapter collects different parts from the different publications supporting this thesis. And it presents the problem of model initialization with Modelica models, which have been submitted into:

- M. SABATE, G. LEON, M. HALAT, J.B. HEYBERGER, **F.J. GOMEZ** AND L. VANFRETTI, "ASPECTS OF POWER SYSTEM MODELING, INITIALIZATION AND SIMULATION USING THE MODELICA LANGUAGE," IEEE POWERTECH 2015.

### 1.6.2 Chapter 3: Requirements for Modeling and Simulation

Previous considerations have led us to define a formalization of functional and non-functional requirements for a new modeling methodology and development of standard-compliant software tools, which are gathered and formally described in this chapter. The implementation of these requirements takes into consideration the use of available information modeling and equation-based modeling standards, as well as software

technology for model-exchange and co-simulation. The work on this chapter is to be submitted as a journal paper in:

- **F. J. GÓMEZ,** M. AGUILERA CHAVES, L. VANFRETTI "SOFTWARE ENGINEERING REQUIREMENTS FOR CYBER-PHYSICAL POWER SYSTEM MODELING TOOLS", TO BE SUBMITTED IN IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, 2019

### 1.6.3 Chapter 4: Multi-Domain Information and Physical Modeling & Simulation

The adoption of other information standard form the thermo-mechanical domain could leverage the dynamics information from the CIM Dynamics Profile. This chapters presents a proof-of-concept of the implementation of new semantic information within the CIM, to create more detailed dynamics information models. And how Modelica could enhance those models with dynamic behaviour modelling. Results of this chapter have been published in:

- **F. J. GÓMEZ**, M. AGUILERA CHAVES, L. VANFRETTI AND S. H. OLSEN, "MULTI-DOMAIN SEMANTIC INFORMATION AND PHYSICAL BEHAVIOR MODELING OF POWER SYSTEMS AND GAS TURBINES EXPANDING THE COMMON INFORMATION MODEL", IN IEEE ACCESS, VOL. 6, PP. 72663-72674, 2018.
- M. AGUILERA, L. VANFRETTI AND **F.J. GÓMEZ**, "EXPERIENCES IN POWER SYSTEM MULTI-DOMAIN MODELING AND SIMULATION WITH MODELICA & FMI: THE CASE OF GAS POWER TURBINES AND POWER SYSTEMS," *2018 WORKSHOP ON MODELING AND SIMULATION OF CYBER-PHYSICAL ENERGY SYSTEMS (MSCPES)*, PORTO, 2018

### 1.6.4 Chapter 5: Model-to-Model Transformation for Power Systems Dynamics Simulations

This chapter focuses on the semantic characteristics of the CIM canonical model and the Modelica languages. A prototype for automatic model generation show how Modelica models can be generated combining the information from CIM and the model behavior from the OpenIPSL Modelica library. Results from this chapter have been published in:

- **F. J. GÓMEZ**, L. VANFRETTI AND S. H. OLSEN, "CIM-COMPLIANT POWER SYSTEM DYNAMIC MODEL-TO-MODEL TRANSFORMATION AND MODELICA SIMULATION," IN *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS*, VOL. 14, NO. 9, PP. 3989-3996, SEPT. 2018.

The detail software implementation of this prototype is submitted and is under revision in:

- **F. J. GÓMEZ**, L. VANFRETTI, M. AGUILERA CHAVES AND S. H. OLSEN, "CIM-2-MOD: A CIM TO MODELICA MAPPING AND MODEL-2-MODEL TRANSFORMATION ENGINE", SOFTWAREX, 2018

The mapping between the CIM and Modelica was discussed during the previous ENTSO-E Interoperability Test organized in Brussels on 2016. Its aim was to propose the Modelica language to complement the exchange of dynamic information from the CGMES information models. This proposal and it appears as part of the CGMES 2.5 in Annex F. The document is available in this web site https://www.entsoe.eu/digital/common-information-model/ or directly from this link: https://docstore.entsoe.eu/Documents/CIM_documents/IOP/CGMES_2_5_TechnicalSpecification_61970-600_Part%201_Ed2.pdf

### 1.6.5 Chapter 6: Software Architecture Design

The proof-of-concept of a software architecture based on different computational engines with the use of those semantic standards is presented to support a methodology for modeling and simulating dynamic models with Modelica compilers. The chapter focuses on modular abstract level design of different engines for different tasks related to modeling, simulation and analysis. This chapter is based and tries to provide deep analysis on the results from submitted to:

- **F.J. GÓMEZ**, L. VANFRETTI, SVEIN H. OLSEN, "A MODELICA-BASED EXECUTION AND SIMULATION ENGINE FOR AUTOMATED POWER SYSTEMS MODEL VALIDATION", INNOVATIVE SMART GRID TECHNOLOGIES (ISGT) EUROPE, ISTANBUL, OCT. 12-15, 2014

### 1.6.6 Chapter 7: Conclusions & Future Work

The thesis is concluded with a summary.

**Other Publications**

- M.A. ADIB MURAD, **F.J. GOMEZ**, AND L. VANFRETTI, "EQUATION-BASED MODELING AND SIMULATION OF THREE-WINDING, AND REGULATING TRANSFORMERS USING MODELICA," IEEE POWERTECH 2015.

- M.A. ADIB MURAD, **F.J. GOMEZ**, AND L. VANFRETTI, "EQUATION-BASED MODELING OF FACTS USING MODELICA," IEEE POWERTECH 2015.

- **F. J. GÓMEZ**, L. VANFRETTI AND S. H. OLSEN, "BINDING CIM AND MODELICA FOR CONSISTENT POWER SYSTEM DYNAMIC MODEL EXCHANGE AND SIMULATION," 2015 IEEE POWER & ENERGY SOCIETY GENERAL MEETING, DENVER, CO, 2015

- L. VANFRETTI, M. A. A. MURAD AND **F.J. GÓMEZ**, "CALIBRATING A VSC-HVDC MODEL FOR DYNAMIC SIMULATIONS USING RAPID AND EMTP SIMULATION DATA," 2017 IEEE POWER & ENERGY SOCIETY GENERAL MEETING, CHICAGO, IL, 2017

- L. VANFRETTI, M. BAUDETTE, A. AMAZOUZ, T. BOGODOROVA, T. RABUZIN, J. LAVENIUS, **F. GÓMEZ**, "RAPID: A MODULAR AND EXTENSIBLE TOOLBOX FOR PARAMETER ESTIMATION OF MODELICA AND FMI COMPLIANT MODELS", SOFTWAREX, VOLUME 5, 2016

- L. VANFRETTI, M.A. ADIB MURAD, **F.J. GÓMEZ**, G. LEÓN, S. MACHADO, J. B. HEYBERGER, S. PETRITENEUD, "TOWARDS AUTOMATED POWER SYSTEM MODEL TRANSFORMATION FOR MULTI-TSO PHASOR TIME DOMAIN SIMULATIONS USING MODELICA" IEEE PES INNOVATIVE SMART GRID TECHNOLOGIES EUROPE, LJUBLJANA, 2016

- LUIGI VANFRETTI, SVEIN H. OLSEN, V.S. NARASIMHAM ARAVA, GIUSEPPE LAERA, ALI BIDADFAR, TIN RABUZIN, SIGURD H. JAKOBSEN, JAN LAVENIUS, MAXIME BAUDETTE, **FRANCISCO J. GÓMEZ-LÓPEZ**, "AN OPEN DATA REPOSITORY AND A DATA PROCESSING SOFTWARE TOOLSET OF AN EQUIVALENT NORDIC GRID MODEL MATCHED TO HISTORICAL ELECTRICITY MARKET DATA", DATA IN BRIEF, VOLUME 11, 2017, PAGES 349-357

# Chapter 2

# State of the Art & Background

## 2.1. SGAM & Cyber-Physical Systems

The Smart Grid Architecture Model (SGAM), developed by the Grid Coordination Group/Reference Architecture Working Group (SG-CG/RA) [5], in the context of the European Commission's Standardization Mandate M/490 [24], is the result of the migration process from the power grid to the Smart Grid, presenting different perspectives and methodologies regarding the development and conceptualization of power grid functionalities on an interoperable manner. The interoperability aspects of the Smart Grid are modelled by the Grid Wise Architectural Council (GWAC), which built a model that structures the power grid regarding enhancements on organizational and technical interoperability aspects, composed by eight interoperability layers and divided in three categories (see Figure 1):

- ***Organizational*** establishes business functions and identifies which parts of these functions lie outside the organization's boundaries, for recording political and economic objectives in policies and regulations, and for a better understanding of the scope and task of specified functions.
- ***Informational*** seeks to transfer the organizational aspects into an appropriate information model, for a better understanding of the use of the information for a specific function, and the concepts contained in the message exchanged between systems.
- ***Technical*** covers the basic connectivity needed to establish networked communication between distributed assets and applications within its lowest layers, for a better understanding of message data structures with a standardized mechanism to exchange using physical and logical connections between systems.

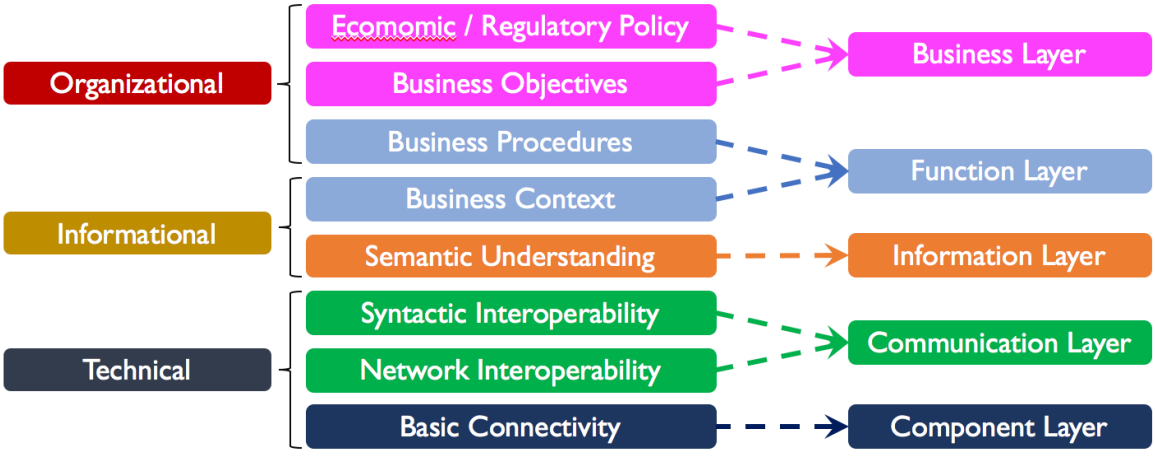


*Figure 1. GWAC model*

The SGAM proposes different layers that cover the organizational, informational, and technical aspects of the use and development of enhanced power systems within the Smart

Grid. The eight layers of the GWAC relate to the five layers of the SGAM, wherein their definition is modified to propose the concept of cyber-physical systems (see Figure 2):

- The ***Business Layer*** presents the business view of information analysis and information exchange as related to the cyber-physical systems.
- The ***Function Layer*** describes system use cases, functions and services including their relationships from an architectural viewpoint.
- The ***Information Layer*** presents the information that is being used and exchanged between functions, services, and components. It contains information objects and the underlying canonical data models.
- The ***Communication Layer*** describes protocols and mechanisms for the interoperable exchange of information between components.
- The ***Component Layer*** describes the physical distribution of all participating components that compose the smart grid as a cyber-physical system.

The adoption of the SGAM by TSOs, DSOs and other players within the electrical grid has elicited new use cases and requirements to comply with interoperability requirements at the level of the Information Layer. Moreover, it leads to continuous development and maintenance of new semantic information, such as the Common Information Model, software tools, and hardware components. Current available technologies for modeling and simulation lead to the development of horizontal integration platforms, design composers, component exporters, library of components, and integration mechanism to compose new physical models.

The Smart Grid can be better conceived as a Cyber-Physical System, defined as "co-engineered interacting networks of physical and computational components that provide the foundation of critical infrastructure, form the basis of emerging and future smart services, and improve the quality of life in many areas." Therefore, the interaction of such new software (computational components) and hardware (physical components) technologies produces a huge amount of data, through which its processing and analysis creates the foundation of understanding a power grid as a Smart Grid.

## 2.2. Information Modelling Semantics and Technologies

In the real world, there is a lot of ambiguity in modelling the same kind of information among different applications and processes. Open standards such as the Common Information Model (CIM) [4] and the Common Grid Model Exchange Standard (CGMES) [6] are conceived of as closing the gap between applications for information exchange. To exchange information between two software applications, the traditional way is to map the respective internal applications' models together.

The context is preserved by manually examining the terms in each internal structure to determine which data are equivalent. Directly mapping two applications together is often the easiest short-term solution. However, those internal models – or databases – may store more information than needed or more detailed representations of power networks, which might affect the long-term mapping solution. Thus, there is a need to expand those mappings into a common language for application interoperability.

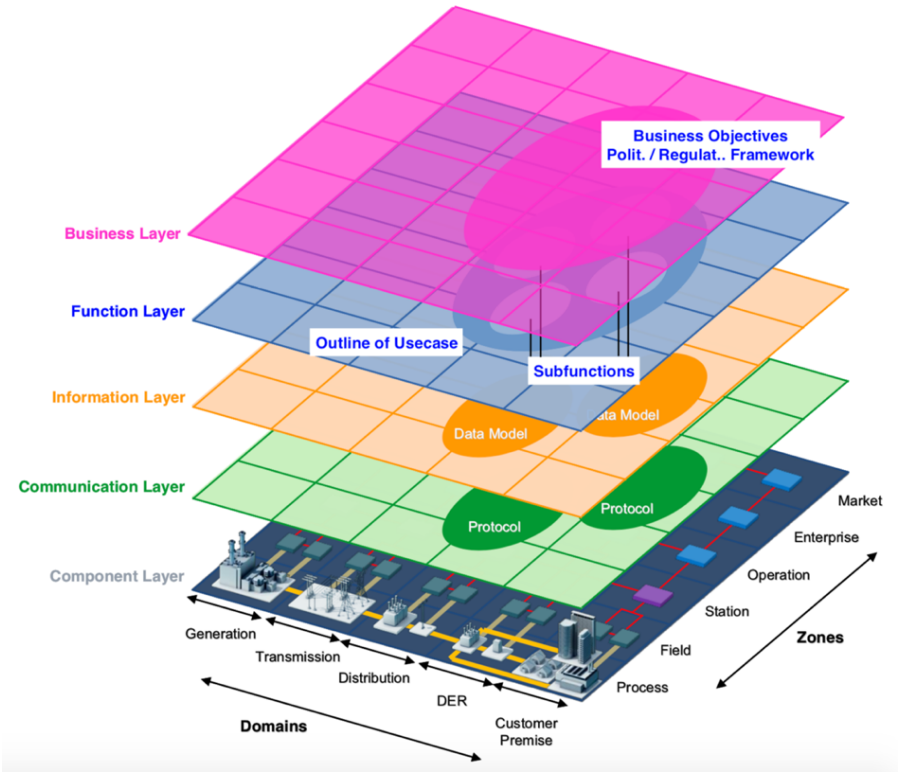


*Figure 2 Smart Grid Architecture Model*

## 2.3. The Common Information Model and the Common Grid Exchange Model Standard

The development of an information and communication technologies infrastructure is needed for the control of power transmission and distribution, planning, and operation procedures within the power grid. The use of standard modelling languages is indispensable for reaching a proper interoperability level within the context of coordination of TSOs. The CIM covers the needs for power network analysis studies and provides information models and semantics for consistent information exchange among different software tools, which could implement different sets of mathematical solvers and provide different implementations of the same power resource model (see Figure 3). With its steady-state behavior and a description on its dynamics behavior, the CIM semantic representation defines all the basic components and topology of the power network.

The CIM is represented by the IEC 61970, the IEC 61968, and the IEC 62325, each of which covers different use cases [3].

- IEC 61970 provides an Energy Management System Application Program Interface that covers an extensive data model for power system modelling and energy utility data exchange;
- IEC 61968 is the Application Integration at Electric Utilities - System Interfaces for Distribution management, covering a data model for secondary grid operations like billing, network extension, assets, metering, and application messaging; and
- IEC 62325 includes a set of data models for energy market communications between market participants and market operators.

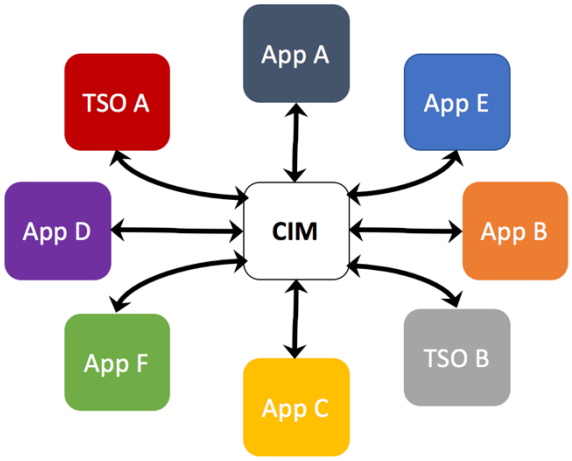


*Figure 3. CIM standard as a central format for interoperability*

The CIM model UML's structural semantic defines sets of classes, attributes, and different association relations among classes, thus representing power system physical objects and their properties. The network's information is classified into different packages that constitute the whole CIM canonical model. The CIM is divided in different component interface specifications that define different information exchange models for different application categories: network information or assets. The focus of this thesis is on different model specifications within the IEC 61970 standard [4].

- IEC 61970-301 CIM Base defines a static information UML package containing UML semantics for the physical characteristics of the power network and electrical and non-electrical characteristics of equipment static models.
- IEC 61970-302 CIM for Dynamics Specification [25] defines a dynamic information UML package containing the UML semantics for the dynamic characteristics of equipment models. The CIM for Dynamics profile includes the parameter specification of these regulating equipment models. While this is very useful for humans to exchange knowledge, a computer-readable description would require a unique mathematical equation specification to guarantee consistency when performing dynamic simulations of such models [4].

Depending on the use of these models, the CIM is divided in different subsets of data, namely ***CIM Profiles***. They limit the scope of the CIM by including only essential classes and restricting the associations' cardinality between classes, which are needed for specific application and operations (see Figure 4):

- The Equipment profile is specified by IEC 61970-452, defining classes and attributes with basic information of equipment models covering the model definitions within the IEC 61970-301.
- The Topology profile is specified by the IEC 61970-456 standard, describing how a model is connected.
- The State Variable profile, also specified within the IEC 61970-456, is the result of a topology processor and power flow calculations.
- The Dynamics profile, specified by the IEC 61970-457 and based on the IEC 61970-302, defines the classes and attributes for the behavior of the equipment.

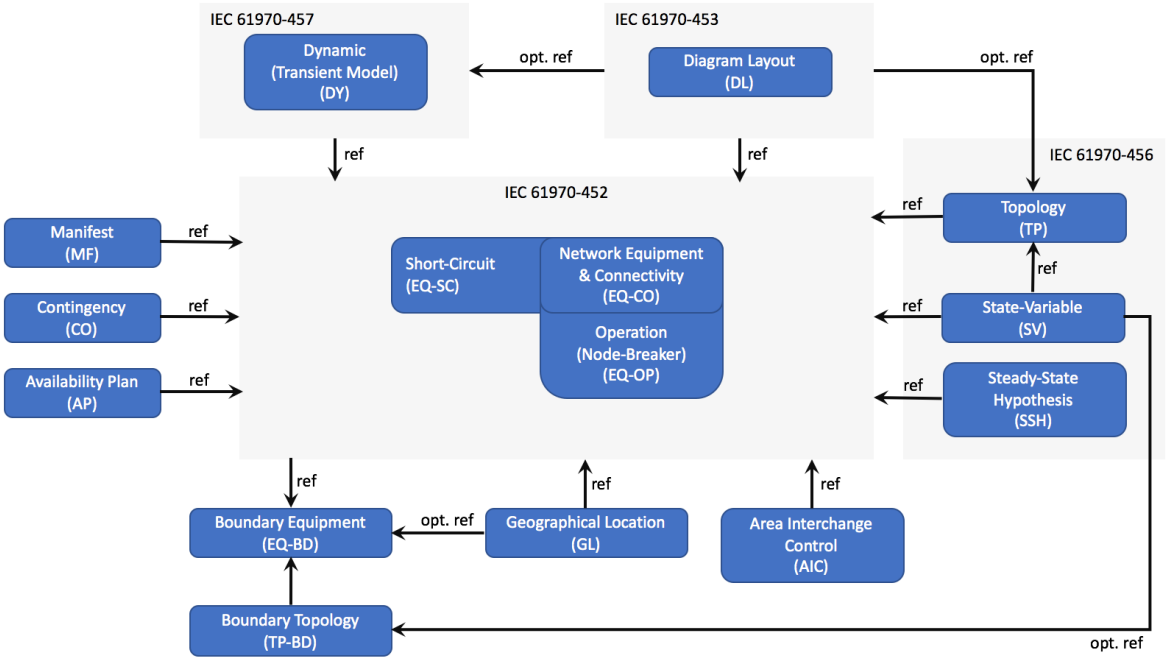


*Figure 4. Classification of CIM Profiles*

The implementation of these profiles is performed by using Semantic Web Language Resource Description Framework (RDF) defined by World Wide Web Consortium (W3C) [26]. The serialization syntax is RDF/XML, which are extensions of the eXtensible Markup Language (XML). The IEC 61970-552 CIM XML Model Exchange Format define language style and implementation rules of the data syntax of CIM.

The CIM/CGMES (IEC 62970-600-1 and -2) standards provide information models of a plant's components from the power system perspective, which might result in a limited definition of physical components' dynamics. The adoption of available standards for projects related to power plants, such as the ISO 15925, can benefit the exchange and reuse of complex plant models information, mitigating the high costs of reformatting information to move it from one proprietary system to another.

### 2.3.1. ISO 15926 for Power Plant Modeling

Both the CIM and CGMES offer a canonical model, from an electrical or power system perspective, of the information used in any kind of network analysis, such as load forecasting or optimal power flow. However, the CIM/CGMES do not cover internal information processes that may occur inside power plants, instead modeling those processes as data objects with a set of corresponding attributes; i.e., the CIM/CGMES do not model the task of designing, specifying, and purchasing a process instrument for the substitution of plant equipment or they do not model the effect of opening a valve as a gain within a turbine governor model. The former aspect may not be strictly necessary to model because it affects the construction of a plant, but the later aspect can have an impact on the dynamic behavior of a turbine model that might affect the energy generation or synchronization of a synchronous machine which produces electricity.

In the same way as the CIM, the ISO 15926 acts like a babel fish, standardizing the description for computer representation of technical information about process plants. ISO 15926 is an international standard that provides an ontology that is aimed at representing process plant lifecycle data, including facilities of oil and gas production [27]. The ISO 15926-2 [29] specifies a stable generic data model with around 200 concepts that enables the exchange and integration of process plant data. Another relevant part of the standard,

namely ISO 15926-4, defines a large set of expandable common reference data elements known as the Reference Data Library (RDL) [30]. Moreover, the ISO 15926 provides reference data models and templates of industrial automation systems for the integration of life-cycle data for process plants including oil and gas production facilities, implemented in OWL [31], resulting in a set of rules for information exchange. This standard covers:

- Specification of requirements to produce, process, and transport process materials.
- Specification and selection of information about market available materials and equipment.
- Installation, maintenance, replacement, and commissioning of plant equipment.
- Production and process operations, including process conditions and consumption, yields and quality of process material, such as:
    - Oil and gas product transport systems,
    - Safety and control systems,
    - Electricity generation and supply systems, and
    - Steam generation and supply systems.

The ISO 15926 standard offers a user-oriented language with basic terminology for representing integrated plant operations, represented with OWL modeling semantics**.** However, it requires being mapped to a GPML [31], such as the UML, to facilitate the creation models of this domain and ensures interoperability with the CIM domain. Thus, the modeling capabilities of the ISO 15926 are studied to gather the relevant features that can be integrated within the CIM.

This ISO standard considers a Thing the most general entity type, which is specialized into two subtypes (see Figure 5). The entity type *Possible Individual* (PI) is required for the representation of industrial plant data, because it is a thing that exists in space and time, such as physical objects. A *PhysicalObject* defines a PI that is a distribution of matter, energy or both, and its subclass *FunctionalPhysicalObject* determines the physical objects by their function.

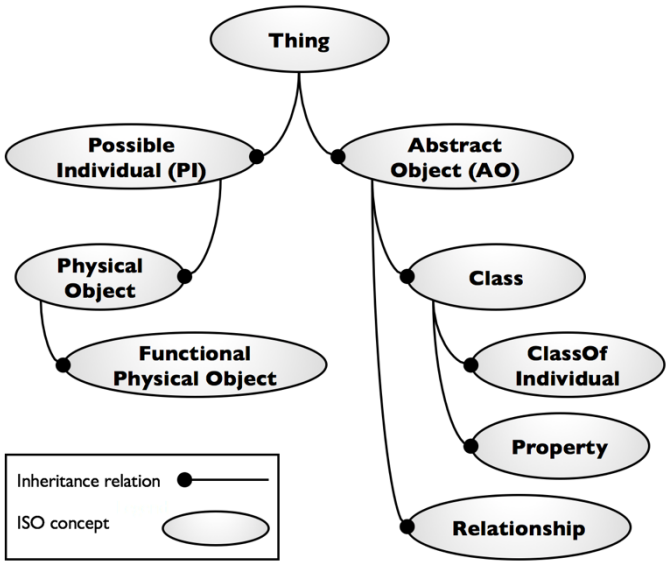


*Figure 5. Subset of entity types from the ISO 15926 generic data model*

The entity type *AbstractObject* (AO) is defined as a thing that does not exist in space-time. Therefore, AO models classifications and relationships between PIs. At this, ISO 15926-2 implements a comprehensive classification schema. For this paper, we focus on subtypes Class and *ClassOfIndividual*, which is used for the classification of PI instances, the subtype Property that can be quantified to a number on a scale, and Relationships for the proposed structural modeling.

## 2.4. Equation-Based Modeling Languages and Technologies

A power system consists of a set of devices that are typically described by continuous dynamics and discrete events. In equation-based modeling, the implementation of complete model or its individual components are represented by a set of *Differential-Algebraic Equations* (DAEs) and its corresponding variables. However, in most cases, their implementation is ambiguous and often tightly coupled to both the computer programming language and the numerical methods used to obtain the steady-state or dynamic response of the system. Thus, it becomes difficult to reuse and maintain power network models, because they are confined to a specific computer program [33].

The use of equation-based modeling languages for modeling and simulation allows the decomposition of complete models into sub-models that can be shared and reused (e.g., the equation-based model of an excitation system that can be used in different power plant configurations [34]). Based on abstract concepts of mathematics and computer science technologies, equation-based modeling supports the development of complex models and sub-models from different engineering domains, which can interact with each other and create multi-domain models [35]. Thus, it is possible to utilized acausal modeling to implement connections between components without specifying any data-flow direction. This allows one to model components of physical objects using equations from different domains that can be connected [36].

Modelica is an object-oriented equation-based modeling language, which allows the representation of cyber-physical systems using a strict mathematical representation for computer simulations of dynamic systems. This is possible thanks to the standardized Modelica language definition, which is based on mathematical equations and permits multidomain modeling capability. Modelica provides the notion of classes and objects which contains elements that can be variable declaration and equation sections. These **class** elements define the data and behavior of the objects or models and are classified in certain stereotypes:

- **Model** is the almost the same as a **class**, though the *model* keyword can better be used to represent high-level models, such as the whole representation of a network, and the *class* keyword works better in representing the models at the physical component level, such as the model of energy consumer component, e.g.:

```
1.  model NetworkGrid
2.     Generator Gen1;
3.     Bus Bus1;
4.     Load Load1;
5.  equation
6.     connect(Gen1.p, Bus1.p);
7.     connect(Bus1.p, Load1.p);
8.  end NetworkGrid;
```

- The **Connector** class is used to represent the structure of ports or interface points of a component, mainly without equations, e.g.:

```
connector Pin
   Voltage V;
   flow Current I;
end Pin;
```

- The **Record** class specifies data without behavior, e.g.:

```
record PowerFlow
   Real voltage;
   Real angle;
   Real activePower;
   Real reactivePower;
end PowerFlow
```

- **Block** is a class with fixed causality, so, for each connector variable of the class, it specifies whether it has input or output causality, e.g.:

```
block ActivePower
   input Real voltage;
   input Real current;
   output Real P;
equation
   P= voltage*current;
end ActivePower
```

- **Package** is a class which contains references to other classes, so they can be classified according to the problem they try to solve.

Modelica is suitable for unambiguous modeling and simulation. A model in Modelica is totally decoupled from the mathematical solver that is used to provide a numerical solution of the model equations [37]. This characteristic guarantees an unambiguous way of modeling and simulation. This model is compiled by a Modelica compiler into machine code. This code can be executed together with a numerical solver of choice, capable of solving sets of DAEs, discrete equations, or a combination. Thus, a Modelica model is not coupled to any specific solver – these are available depend on each tool's implementation – and it can be easily exchanged between software environments or specific applications capable of compiling Modelica code. This is a certain advantage in terms of model development, because the modeler can focus on the actual model creation, without worrying about the underlying simulation engine.

The Open Source Modelica Consortium (OSMC) provides open-source libraries for different fields, e.g., thermal models and mechanical models, offering a choice for power system model exchange of information, even those that may be the same in a huge variety of proprietary tools. Simulation tools and API based on Modelica, such as OpenModelica [38] or JModelica [40], facilitate the development of well-known industry models. The use of Modelica also permits developers and engineers to modify or add new components and equations and to share these modifications, within the same model.

### 2.4.1. Modelica for Power Systems: The OpenIPSL Case

From the work developed within the FP7 iTESLA Project, two important facts are identified in terms of information exchange coordination. First, TSOs should be able to easily import the data to be assessed. Context suggests that the use of the CIM format should play a major role in the definition of static data. Second, the use and acceptance of user-defined models in an open format thus allowing the modelling of specific and new

devices are also expected. The use of Modelica language for modelling these user-defined models provides a choice in making this exchange of model information open for TSOs. A prototype of software enables users to transform power system networks from the CIM format to Modelica [39], but it locks models within an internal data structure and forces the model to have its power flow solution solved again.

Power system models used in time-domain simulations can be categorized into Electro-Magnetic Transient, Phasor Time Domain, or Quasi Steady State [40]. The modeling of such models may also vary depending on the kind of studies to be performed or the simulation's solver approach [41]. And limitations on power system tool may also have influence in the modeling approach. The kind of studies may vary depending on their application; for instance, transformers models may be used for lightning overvoltage studies or to represent specific physical phenomena, and transformer models might be represented by models based on leakage inductance or transmission line modelling.

An example of equation-based modeling for those applications is presented in [42], where two different Modelica transformer models have been developed and simulated under different test cases. The choice of solver may also have an influence in the modeling, making it difficult to evaluate the quality of the model among different tools that are used for the same kind of studies. An example of differences in the performance of different solvers is also studied in [44]. The work shows the use of Modelica for modeling FACTS devices as well as the differences in solving the dynamic behavior of those components under dynamic events.

To close the gap between modeling and simulation limitations and exchange of model information, Modelica models for power systems components have been implemented, following the mathematical representation of these models and used to develop different network models [45]. The use of Modeling ensures unambiguous modeling, providing consistent exchange of the information to represent dynamic behavior and to perform consistent simulations in different tools. Those previous works were developed to show how Modelica models are decoupled from any mathematical solver. They accept modifications in their equations and their parameters, providing unambiguous simulation results among different simulations tools [42].

The Open Instance Power Systems Library (OpenIPSL) [34], a Modelica library, is the result of such works. The OpenIPSL contains different categories of power system components' models, which are used phasor time-domain dynamic simulations. The OpenIPSL provides as many typical "test networks" as possible, developed in the direction that is more suitable for researchers in a transparent, open-source software approach, offering maximum compatibility with power system dynamic simulation within Modelica-based workflows such as the OpenModelica compiler [38].

### 2.4.2. Initialization of Modelica Models

For a well-defined dynamic model, initial values for all derivative and algebraic variables need to be provided at the initial time before the simulation starts. While power system domain specific tools first solve the set of algebraic variables of the network and then solve the set of algebraic and dynamic state variables for the differential equations of the network, Modelica compilers attempt to solve the second set of variables, setting their derivatives to zero and simultaneously solving the resulting non-linear algebraic equations [46]. Zero start values can cause singularity or convergence problems during the initialization process; thus, the initialization process should be carefully considered in Modelica models.

Initial conditions need to be provided by previous calculations of the network; in other words, power flow solution results should be provided for Modelica models to avoid initialization issues. In [47], Modelica initialization variable constructs are discussed and used, depending on the reference model used for the validation of the corresponding Modelica model. OpenIPSL models use as the ***start*** attribute for the state variables. The variability of a certain variable will explicitly indicate how to assign the start value of the variable. A variable with no variability indicates that its value will change during the simulation, so a start keyword should be indicated for assigning the initial value of the variable:

```
1.  Real P (start = x0);
```

Where **x0** is the guess value used when the Modelica compilers solve the initialization problem. If a continuous-time variable is declared with the attribute ***fixed***, it indicates that an equation to solve this variable will be added at initial time:

```
1.  Real x (start= x0, fixed= true)
```

For another kind of variability, such as the ***parameter***, it indicates that its value does not change during simulation, but it can be changed between simulations. The starting value of a variable will be assigned as an equation:

```
1.  parameter Real S = x0;
```

Dynamic models contain continuous-time variables which are initialized by the ***initial equation*** section. This model section is useful for initialized system variables in steady-state; hence, for variables that do not change with time or do not change if they are not affected by a dynamic event, a value for its derivative is provided and the equation is included in the initialization process. Given these considerations, the components from the OpenIPSL library (used in this thesis) are constructed in a way to facilitate the computation of the initial steps before the computation of the model's equations. Thus, the steady-state solutions from a power flow solution, are used to initialize the parameter values of the power grid components.

## 2.5. Software Technology for Modelling and Simulation

### 2.5.1. Development Process

According to the definition of [48], the function of the development process is to determine the order of stages involved in the development and evolution of software, as well as to establish the transition criteria for progressing from one stage to another. A development process is important because it provides guidance on the order in which a project should carry out its major tasks to achieve a certain goal. The development process, in general and by self-experience, could be divided in four main stages:

- The *Definition of Requirements* describes the tasks and use cases to fulfill certain objectives. The use cases contain information about which tools and models should be used and how the models should be validated to provide technical description of the tasks to be implemented. At this stage, it is common practice to describe the pre-conditions and post-conditions of the main tasks to be developed.
- It follows the *Design Process* of the proposed use cases, with the definition of the software architecture to implement. The design also involves the specification of parameters and the methods to be used, as well as how those methods will interact with each other and how the information will be exchanged to fit the tasks and description of each use case.

- The *Implementation Process* deals with the coding of the proposed design. In this stage, the tools, programming language, and modeling language are chosen. Thus, the implementation process not only aims to generate executable code, but it also deals with the implementation of models for later analysis and simulation.
- In the *Test & Validation Process*, the execution of the proposed implementation or the simulation of the implemented model is carried out to analyze if the results of the previous implementation match with reference models or fulfills the expected outcomes. The latest one can be obtained via user interaction with the proposed implementation, while the former one can be obtained comparing available data from different sources of information.

The results in [48] reveal some of the development process, which do not follow these basic principles, such as the code-and-fix process, or where these principles are strongly defined, such as the Waterfall model. A variant of the latest process is the V-Model [49], which defines different stages of the process model, from a general view to specific implementation and going up through the testing and integration of the specific component to the whole system, with constant relationships and communication of each stage (see Figure 6). This development model has been chosen for the implementation of the power system Modelica models in use, because there have been strong definitions of the objectives and the outcomes of those models.

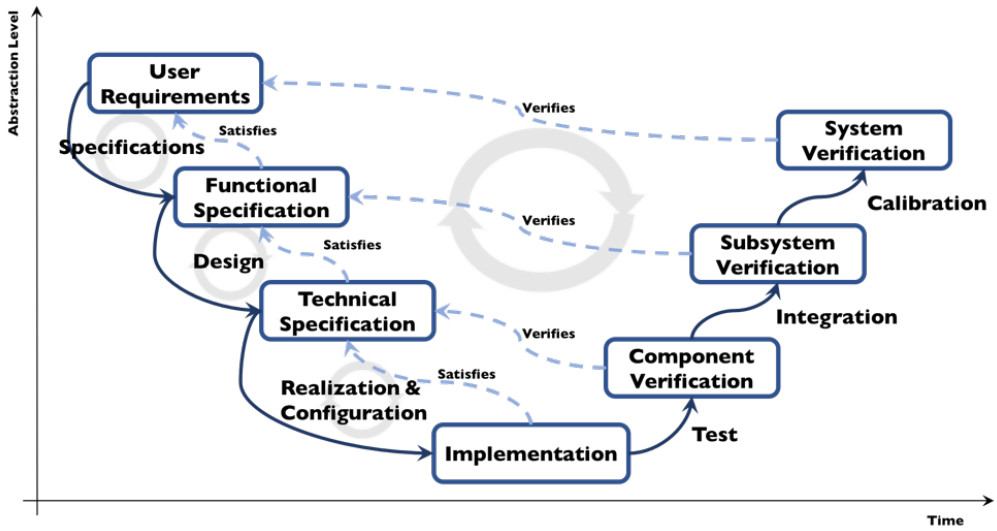


*Figure 6. Representation of the V-Model with respect to abstraction level and time*

An evolution of the previous development processes is the Spiral process model, divided into cycles for every progression in the development of a product (see Figure 7). Every cycle involves a progression from the overall functional specification down to the coding of each individual program, starting with a required definition of specifications, which considers:

- the identification of the part of the model or software tool to be elaborated;
- the identification of the alternative means of implementing those parts; and
- the identification of the constraints imposed on the application of the alternatives.

These steps lead to the identification and evaluation of uncertainties and risks of the finalized prototype. If these uncertainties have been clarified and the risk is minimum, with a robust enough proof-of-concept implementation, future evolution of the product are designed and implemented in the next cycles of the process [48].

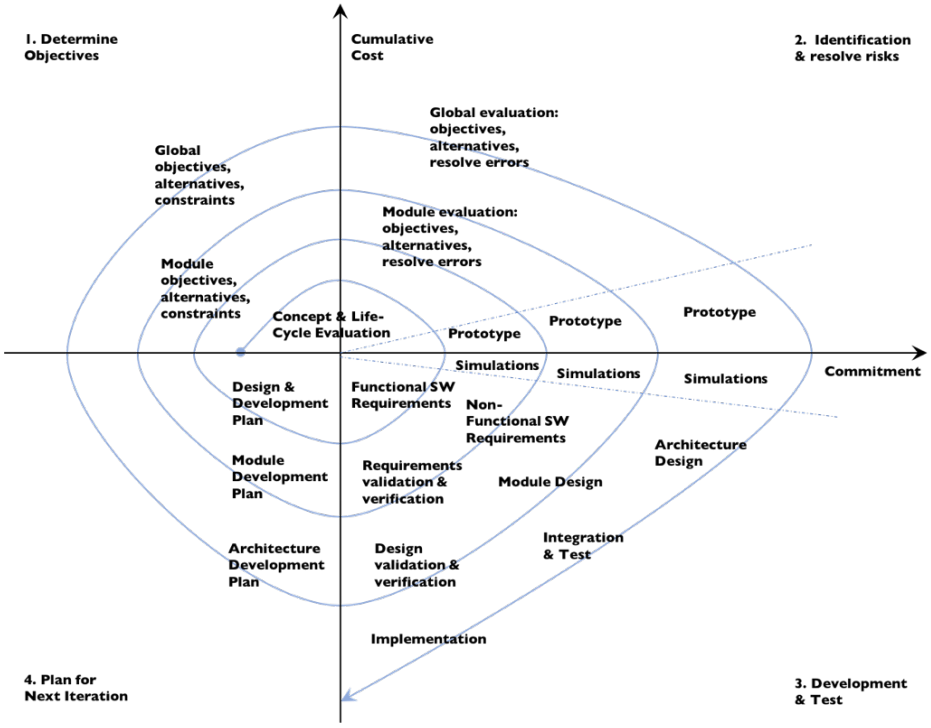


*Figure 7. Representation of the different stages of the Agile methodology for software development*

### 2.5.2. The Choice of Open-Source Software

Proprietary software, being the de facto choice in power system simulations, is perceived as well-tested and more computationally efficient with respect to other tools. Some of these software tools are powerful in few types of analysis methods with the use of specific data format and data models. However, there are several drawbacks to proprietary software. The first to consider is the license agreement restricting the use of proprietary software by imposing economic or other limitations [50]. Free/Libre and Open Source Software (FLOSS) have been developed as alternatives to the proprietary software development model. Through different experiences, the FOSS model has been shown to produce reliable, secure, and efficient code. This fact has recently drawn some attention from the power systems community, resulting in the GOOSA effort [51].

However, before this, FLOSS was more limited to education and learning activities and generally conceived for use in power system courses [52] even though they provide almost the same functionalities as commercial software. FLOSS can be exploited in the power system field to provide tools allowing engineers to change or implement new code, evaluate new techniques and algorithms, and guarantee freedom and liberty to exchange information [50]. Emerging challenges in power systems may be better approached through the FLOSS model than the traditional techniques found in proprietary software.

### 2.5.3. General Purpose Modeling Languages

The adoption of other information standards within the Power System domain can be used to extend the CIM/CGMES information models and profiles. Equation-based modeling languages for dynamic modeling and simulation can complement those information standards. The use of General Purpose Modeling Languages (GPMLs), with a focus on object-oriented modeling and development, such as the Unified Modeling Language (UML) and Systems Modeling Language (SysML), and DSML, such as Modelica, can be combined. Those GPMLs can be exploited to represent the requirements of a

modeling system or simulation system, thus representing the parametrization of components and their equations, or providing graphical representation that can be utilized in a variety of SCADA systems.

UML [53] is one of the Object Management Group (OMG) ISO Standard specification for system modeling. UML plays an important role in Model-Driven Development (MDD), providing a standardized modeling language to create models for software development. The UML consists of a set of elements to give a standard interpretation and standard representation of the properties and behaviors of the system. UML semantics are classified in two categories: structural, defining the meaning of the model's elements of an engineering domain at specific point in time, such as a Class Diagram representation, and behavioral, defining model elements that change over time, such as a state machine diagram or sequence diagram representations.

The SysML [54] is an extension of UML for modeling of system engineering applications that considers hardware, software, information, and processes. It provides semantics for engineering modeling that combines hardware, software, information, and processes. It also provides additional design principles such as requirements modeling and the reuse of such UML class diagrams as block diagrams supporting parametric modeling interoperability. Its graphical representation benefits the understanding of system constraints and the interactions between components and actors related to the system. Such graphical representations complement the description of technical specifications of simulation tools or algorithms, often described with the ambiguity of natural language.

### 2.5.4. Semantic Web Modeling Languages

The World Wide Web Consortium (W3C) provides standard Semantic Web languages to create semantic models of the real world. The Resource Description Framework (RDF) [55] is a semantic modeling language, utilized in Semantic Web for managing distributed data, which organizes the data providing an integrated description of objects in the world and their relationships. The Semantic Web refers to things from the real world such as resources; thus, this definition is extended to things from the engineering domain. Because the RDF manages the data in table cells of three values (see Table 1), the basic data construct for RDF is called a triple, where the subject is the identifier of a row, the predicate is the identifier of a column, and the object is the identifier of the value of a cell [56].

RDF is used by the CIM to implement an ontology with semantic power system information. The Web Ontology Language (OWL) [57] is another modeling language from the Semantic Web that provides classes, properties, and data values for distributed data modeling. In its version 2.0, it extends the RDF with additional modeling capabilities, such as *Restriction Class* to describe new entities with classes that could already exist in a current working model. The concept of classes and attributes plays an important role in the development of information models as well as in Model-Driven Software Engineering. The classes and attributes are utilized to create models for software development.

*Table 1 Sample of Gas Turbine Components Triples*

| ***Subject*** | ***Predicate*** | ***Object*** |
|---|---|---|
| GasTurbine | consist_of | Compressor |
| Compressor | has_property | InletDesignTemperature |
| InletDesignTemperature | has_magnitude | Temperature |

### 2.5.5. Model-Driven Development and Model Transformations

As mentioned in section 2.3, the CIM defines all the available information for the power system network and components, with their steady-state behavior and limited description for their dynamics information. Within the CIM and CGMES, the physical dynamic behavior is only represented by parameters and described within block diagrams rather than using their mathematical form, i.e., mathematical equations. CIM and CGMES do not provide a mathematical description defining the power system models – only parameters and pictorial diagrams are exchanged.

The IEC CIM for Dynamic defines relevant standard parametric models for physical dynamic components' information exchange. The abstract definition from well-defined standards, such as the UML, is not enough for commercial modelling and for simulation tool vendors to share their models in an unambiguous way. Even with the adoption of the CIM & CGMES, most power system analysis tools implement software interfaces to transform the CIM files into a tool's internal proprietary tool data format and representation, as illustrated in *Figure* 8. Those software tools provide their own internal API which is responsible for parsing the CIM model, using available code libraries that can process data implemented with the RDF/XML syntax. Those APIs parse the CIM/CGMES model and populates its data into the tool's internal data structure, for which, in most cases, its implementation is unknown.

The adoption of available modelling languages could enhance the unambiguous development and representation of power system models. Thus, the application of Model-Driven Software Engineering (MDSE) techniques has received more attention for their use in building new software analysis and simulation tools. MDSE is a subset of Model-Driven Development (MDD) that consists of a development paradigm based on models, which relies on the use of OMG standards for software and systems modelling. The software development process within the MDSE requires models and model transformations.

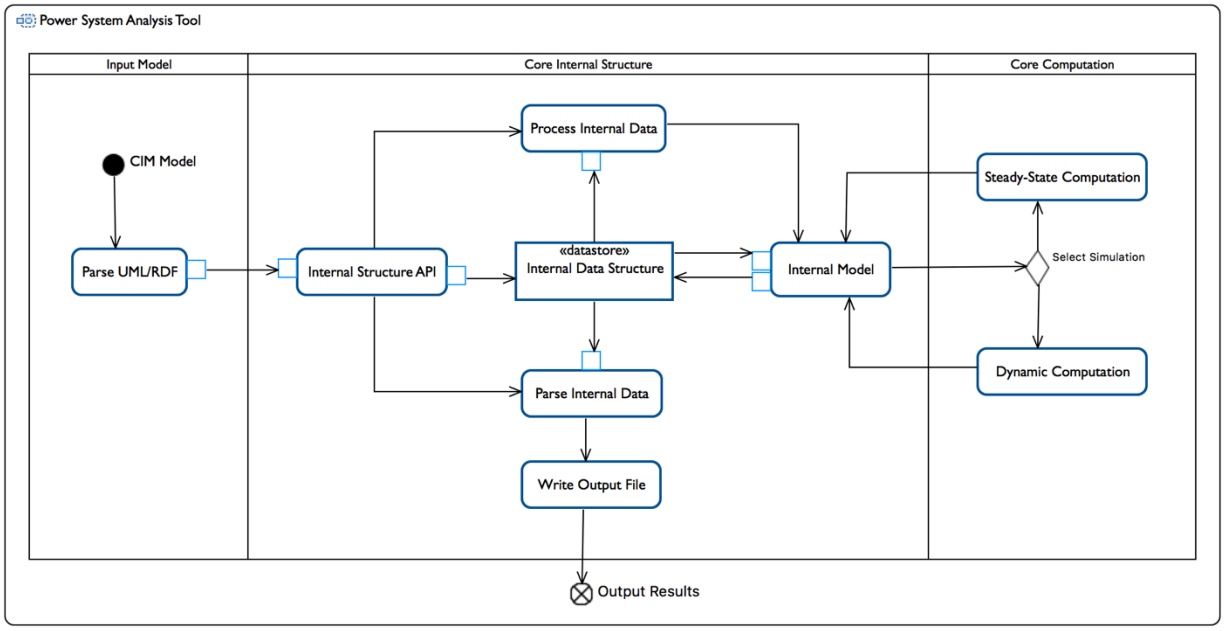


*Figure 8. UML representation of the workflow of power systems analysis tools, which have the capability to adopt the model-to-text (M2T) approach, with the adoption of the CIM/UML model representation.*

In general, the transformation process is a method that produces an output information model or data from other information models or data as input. Moreover, these transformations require the development of mappings between a source model and a target model [58]. However, such mappings are not available because they are implemented as parsers or data format converters within the API and internal structure of most of the power system tools. This leads one to consider two types of model transformation within the MDSE:

- **Text-to-Text (T2T)** deals with the transformation of origin data or input program code into documentation or program code, which can be executed by a machine.
- **Model-to-Text (M2T)** describes the process of the transformation of an input model, implemented in a modelling language, such UML, into program code or documentation. This transformation can be also interpreted as code generation, which is applied by most software vendors to enhance their tools with the development of new models. The opposite transformation process, Text-to-Model (T2M), is mainly used for reverse engineering
- **Model-to-Model (M2M)** describes the transformation of source model, defined via a modelling language, into a target model, defined in a different or the same modelling language, without compiling the target model into program code (see Figure 9).

Most power system analysis tools only implement T2T transformations, known in the power system domain as "parsers" or "data file format converters/filters." Recently, with the goal of adopting the CIM standards, some power system analysis tools have implemented software interfaces for M2T transformation, reading CIM files into the tools' internal proprietary tool data format and representation, as illustrated in Figure 8 (see [47] for a detailed example of such approach).

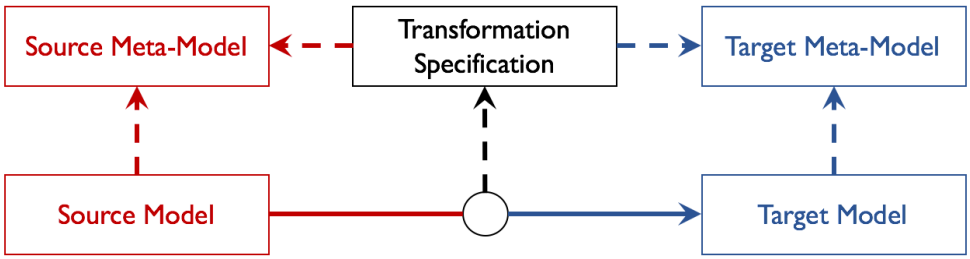


*Figure 9. Exogenous Model-to-Model transformation: different modeling languages are used for the source model and for the target model*

The UML representation of the CIM is studied to understand the terminology of a power system model in CIM: which classes and attributes represent the different power systems components and how the relationships among these classes reveal the topology of a network model. On the other hand, the meta-representation of the Modelica is also studied to understand the terminology of component modelling using equations and component blocks. Instances of these blocks must be initialized, given the correct initial conditions, and are used as variables of the high-level network model.

### 2.5.6. The Functional Mock-up Interface Standard

New technologies enhance simulation capabilities, including the ability of models to become more detailed. Consequently, a power system model could be developed by integrating equations from other domains in such a way that conventional over-simplifications are reduced. One example is the modeling and simulation of the

communication system for automatic controls and protections relays. The Functional Mock-up Interface (FMI) [59] has great potential in facilitating the data and functional interoperability of multi-domain components in a system model. FMI is particularly suitable for Cyber-Physical Systems (CPSs), where model components may represent distinct subsystems from different domains.

Via the FMI standard, the models from each domain can be exchanged using Functional Mock-Up Units (FMUs). In Section V, the feasibility of exchanging unambiguous detailed multi-domain models, i.e., a combination of gas turbine model with a power system grid model, is demonstrated using a simulation tool that supports the FMI standard. This shows the potential the FMI standard has for multi-domain model exchange that would allow manufacturers to exchange equation-based models, while at the same time protecting their intellectual property.

The FMI standard enables the translation of equation-based models into blocks or FMUs containing the model description and the model executable code. The standard provides two different interfaces:

1) The FMI for Model Exchange where only the description of the model and its equations are exchanged. This requires that the solvers from the simulation tool importing the FMU are compatible with the FMI standard and utilize the binary code from the FMU; and

2) FMI for Co-Simulation where the description, the equations and the mathematical solver from the simulation tool that creates the FMU are exchanged within the same FMU block. In this way, a master simulation tool only needs to invoke FMI standard procedures to execute the FMU.

### 2.5.7. Data Formats

While UML and SysML provide a semantic language to model any kind of system, specific information modeling semantics, such as the CIM, provide an open data format that can be used to store steady-state and dynamic simulation results. The IEC 61970-552 for CIM/XML Model Exchange Format defines the data syntax style and implementation rules of CIM-based information that is transferred by means of files. The CIM/XML implementation is based on the RDF which is an extension of the eXtensible Markup Language (XML) [60]. However, dynamic casting, multi inheritance and other factors make reading multi-million entry files very slow.

Other text data formats and binary data formats are commonly used for information exchange by power systems simulation tools. Text data formats, such as .txt and .csv, are commonly used to store data with predefined structures from the analysis tools in use. The meta-data information for processing these data is implemented within the software tool. These data formats are not self-explanatory; thus, additional meta-data should be exchanged with the user to allow him/her to understand what information can be extracted from the data. A common text-based format for power systems data is the PSSE file format [61]. It provides an advantage among other formats, because it is easy enough to read with a text-based application; moreover, it provides enough information for network topology, load flow, and short circuit-type studies.

Binary format files, such as the Matlab MAT format [62], contain meta-information and provide an advantage regarding the text-based formats, because they are self-explanatory. Thus, it is easier to process the information of those files without being attached to any software tool. However, most of those formats are proprietary data formats that require specific commercial software tools for processing the information contained in the files.

The application of an open-standard binary data formats, such as the HDF5, is considered. HDF5 data format offers a platform-independent Application Program Interface (API) to implement a well-defined hierarchical data format structure that allows the storage of any kind of data, with its corresponding meta-information, in a single binary file [63]. The HDF5 is proposed as a complement to other existing binary and data formats, to store input data for and output data from simulation tools.

## 2.6. Summary

This chapter offers an overview of available software technologies and standards related to the design and implementation of power system information modeling and simulation. The SGAM offers architecture framework and layers regarding where to include those software technologies to conceive the model development and model simulation of power system components for the study of the network grid as a complex cyber-physical power system. In the context of the SGAM Information Layer, General Purposes Modeling Languages are used within this thesis for the design and evaluation of requirements to be considered for new power system analysis tools (see more details in Chapter 3) and for the design of workflows, classes, and blocks used to represent the proposed software architecture (further explanations is given in Chapter 6).

The UML is used for the abstract and semantic model representation by the CIM/CGMES models. The SysML is used to define an abstract semantic model representation, in combination with the Modelica language, to develop and implement simulation dynamics models. Both syntax implementation of the CIM/CGMES and the syntax implementation of Modelica are considered in the development of a software tool for the automatic generation of those dynamics models, using user-defined mapping rules and model transformation process (see more details in Chapter 5). The models used and developed during this work are based on the development of the OpenIPSL Modelica library for power systems models. Some of the models reported in the references have been used as reference models for the test and validation of the software prototype developed. This software prototype consists of a software architecture with capabilities for:

1. The modeling of power system dynamic models, based on the CIM/CGMES and Modelica languages;
2. The simulation of power system models for dynamic studies, taking advantage of the computation capabilities offered by different available Modelica compilers; and
3. The use of model validation and signal validation methods for the evaluation of dynamic simulations results, and the storage of these results for further model updates or model analysis.

During the development of this thesis, the steps defined by the V-Model and the Spiral-Model software development processes have been considered. The methodology provided by the Spiral-Model has been used for the development of constants and the validation of requirements, the development of prototypes and testing and validation, until a suitable solution of the concept can be developed. That does not mean that the work is closed and finished, because, for, as the methodology suggests, the work is continuously adapting to the new requirements of actors involved in the development and use of power systems' cyber-physical systems.

# Chapter 3

# Requirements for Modeling & Simulation

## 3.1. Requirements for power systems domain-specific M&S tools

To create the model of a system, a simplified or complex representation of a physical system can be expressed in terms of mathematical equations. In case of power system, the set of devices that are typically described by continuous dynamics and discrete events, are also described with mathematical equations. Those equations are then translated into a computer program and, in the case of power system models, become suitable for steady-state and dynamic simulation studies. The simplifications and hypothesis made during the design of power system models are often driven by the numerical solution methods that are available in a specific tool to solve the given system equations, which in turn prescribe modeling specifications. Thus, the implementation of those model components is in most cases ambiguous, and often tightly coupled to both the software tool and the numerical methods used to solve steady-state or dynamic analyses.

Both models and methods are implemented in the power system tool by a specific vendor or developer. Thus, the modeling of any power system is defined by the constraints imposed by the model components or model information that is available in the simulation tool, or available from the component's vendor/developer [64]. This makes it difficult to reuse and maintain network models, because they are confined to specific computer program. Furthermore, de facto simplifications and hypothesis do not consider the uncertainties that might affect the behavior of the model. These uncertainties are often considered insignificant and simulation results are expected to be conservative [65]. Different strategies are employed to model a limited type of uncertainties of the component's dynamic behavior in the power system model.

Conventionally, model and tool development is conceived as solving either short-term or specific technical problems. Models only allow one to represent specific time-scales such as "static" (i.e. power flow), "dynamic" electro-mechanical, and electro-magnetic dynamics (i.e., fast-transient system responses). In this approach, models are described within the structure of a specific simulation solver (e.g., discretized models using the trapezoidal integration rule); thus, unambiguous modeling is left for the later stages of the tool development process if at all considered, resulting in a deficit of general requirements catering to the use of modeling and simulation standards.

### 3.1.1. Software Technologies for Requirements Formalization

The typical system approach, to implement such requirements, consists in applying random variations to parameters or by defining more complex power system models with the use of "user-defined models" in a tool's own DSL. these strategies are tool specific, lack of portability and reproducibility of results becomes cumbersome. It is common practice to express those requirements using natural language.

However, technical specifications, such as class attributes and relations, data flows from one block to another or describing the implementation of a process are not easy to explain only using natural language [66]. The interpretation of these requirements, might increase the design and implementation cost of the functionalities and data specifications

described by the natural language. This extra development costs might be produced because it is expected that the engineer interpreting those requirements would have some have experience in software development.

The requirements presented in this work help to argue in favor the separation of both modeling and simulation stages in isolation from each other. This is possible with the application of Model-Based Software Engineering principles and the use of the FMI standard used and discussed in this work. For example, UML tools such as Papyrus + Moka [67] can implement a FMI API to import FMU blocks into a UML model, or vice versa. This could lead to develop CIM-based tools that can export their information into FMU block. Another example is the use of FMI compilers or Modelica compilers with support for FMUs to execute and represent the behavior of those models. This makes possible to the user the freedom to choose and exploit the models in different computer environments.

The development process described in *Chapter 2 section 4.1.* started with an interpretation and implementation of a first set of requirements expressed in natural language. Following the development process leads to adapt principles from Software Engineering to propose a formalization of design requirements for the migration process from legacy power system analysis tools to new standard-compliant software tools. For this purpose, the Use Case Methodology based from the IEC 62559-2 [68] is followed in the sequel. The migration process considers the development of tools for modeling, and separately, tools for simulation, adding new functionalities, to the well-known power system analysis and simulation methods, which can fulfill TSOs specifications concerning the exploitation of simulation models for dynamic studies.

Those specifications can be classified as functional and non-functional requirements. The former ones represent what the modeling and simulation tools should do. The later ones capture the implementation details for the functional requirements, i.e. capture the technical languages and methods for representing how the functional requirements should be implemented. The functional and non-functional requirements make use of available standard modeling semantics, such as the CIM for power grid information modeling and Modelica for equation-based modeling. In addition, strategies for multi-domain simulation with the adoption of the FMI standard for Model Exchange and Co-Simulation, are also considered. A general use case for modeling tools and another for simulation tools is discussed. Each of them is described using parts of the use case template suggested by the IEC 62559.

## 3.2. Formalization of Requirements for Modeling Tools

This section describes the base scenario use case **for modeling tool requirements to support standard modeling languages and allow multi-domain modeling**. Following the template provided by the IEC 62559, the general description and objectives of the use case are given. The functional requirements within this base scenario use case will be described with a SysML Use Case diagram. Moreover, description of the information exchanged, relation between actors and requirements will be provided. To complete and give further details of this base scenario use case, the SysML Requirement diagrams will be used to gather and describe the non-functional requirements that could be applied for the implementation of the functional requirements.

### 3.2.1. Description and Objectives

New modeling tools could be developed as stand-alone applications or plug-ins for general purpose simulation tools. Their functionality could be developed as modules able to interact with each other. To develop more accurate system models, the adoption of

multi-domain modeling is attractive and feasible by using the well-known CIM standard in combination with semantics from the ISO 15926. A module for automatic generation of models could help the model developer to generate different model configurations. Such a module combines the information model with Modelica libraries, e.g. OpenIPSL [34] to describe the dynamic behavior of typical component models and user-defined models.

In the case of user-defined models, the use of libraries such as the OpenIPSL allow scalability and reimplementation of different parts of the model, if necessary. Modelica makes the network representation transparent to all users. But in case the model should have certain protection level, the modeling tool should be able to generate FMUs, which can be imported or exported to produce even more detailed and complex models. This would require model checking functionality would be desirable to assess the model's implementation. Table 2 shows the pre-conditions that these new modeling tools need to implement.

### 3.2.2. Use Case Actors

The use case actors define those entities that have something to do with development of models. The responsibilities for model development are classified according to which person or system is responsible for certain functionalities (see Table 3). In this use case, two actors are identified: A *Model Developer*, which is the domain expert that can work with both parameters and equations of the model, and the *Modeling Tool*, with capabilities to help the developer to define a complete model. Notice that a third actor, a *Tool Developer*, is implicit in this definition of requirements, because is the one responsible of implementing these functionalities.

### 3.2.3. Use case functional requirements

The main functional requirements, their relation with actors and other functional requirements to be developed in a modeling tool, are shown in Figure 10. Information modeling and equation-based modeling are combined to develop detailed simulation models, with mechanisms for exchange of these models. Thus, the models can be reused in different simulation environments that comply with the standards in use. Following the Use Case Methodology, the

Table 4 provide the necessary description to interpret the functional requirements from Figure 10.

*Table 2 Pre-requisites for the development on new Simulation Tools*

| **Assumptions** |
|---|
| Information modeling with CIM and ISO 15926 is applied. |
| **Pre-Requisites** |
| CIM profile definition is implemented following CIM meta-model structure definitions. |
| **Assumptions** |
| Equation-Modeling is applied to complete the dynamic information for dynamic simulation models. |
| **Pre-Requisites** |
| Modelica language class stereotypes and attribute definition is implemented. |
| **Assumptions** |
| Exchange of both information models and equation-based models. |
| **Pre-Requisites** |
| FMI standard API for FMU generation is implemented. |

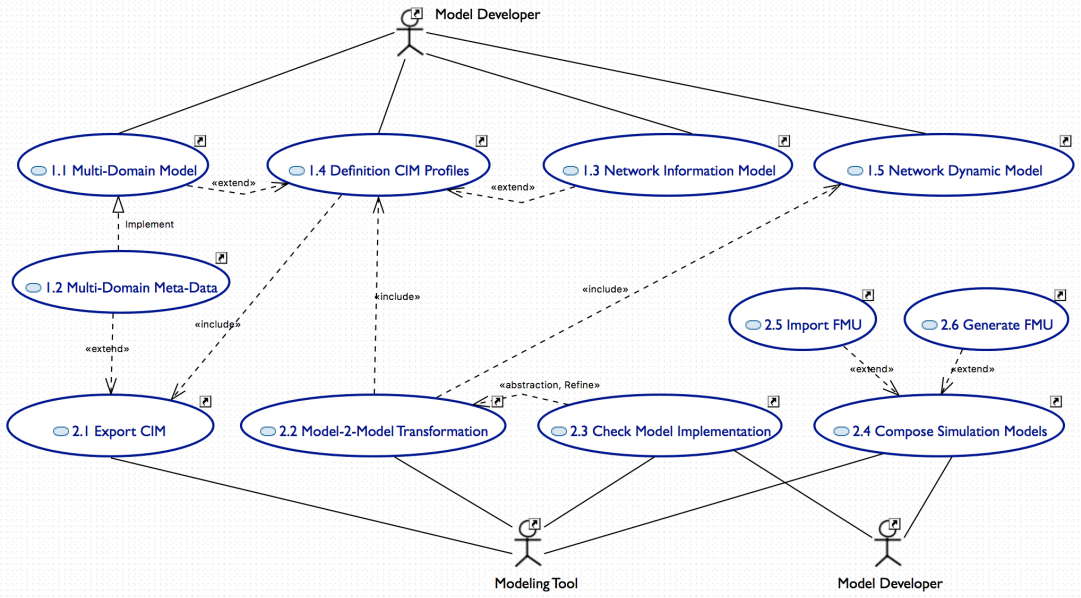


*Figure 10 Use case diagram with the main functional requirements*

*Table 3 Description of the role for the Modeling Tool Use Case Actors*

| Actor Name | Actor Type | Description |
|---|---|---|
| Model Developer | Human | Creates the definition of parameters, mathematical equations and topology of the power system model. |
| Modeling Tool | System | Is responsible to generate the resultant power system model, understand the three main standard representations: CIM, Modelica and FMU. |

*Table 4 Description of power systems specific functional requirements for modeling*

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 1.1 | Multi-Domain Model | Development and enhancement of power systems information model using the information associated with engineering, construction and operation of process plant, represented by the ISO 15926. |
| **Information Exchange** | **Relation** | **Information Description** |
| Information Model for Power Plant Systems | Associated with Model Developer<br>Implemented by Req. 1.2 | This information model defines semantic power plant components models from the ISO 15926, using classes, attributes and class relations. This information model is compatible with the IEC 61970 UML model for power systems. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 1.2 | Multi-Domain Meta-Data | Creation of an entity-relation data structure, from the ISO 15926 Information Model. |
| **Information Exchange** | **Relation** | **Information Description** |
| Meta-Data Structure | Implements Req. 1.1<br>Extends the Req. 2.1 | Provides a UML class structure from the information model specified by the standard. Contains a definition of an information schema and namespace, to identify the information from the standard, to be used within the implementation of CIM profiles. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 1.3 | Network Information Model | Development of power systems information model using semantic information provided by the CIM IEC 61970-(301,302). |
| **Information Exchange** | **Relation** | **Information Description** |
| Network canonical Model | Associated with Model Developer<br>Extends Req. 1.4 | The network model contains the definition of power systems components, modeled by IEC 61970 classes, attributes and class associations |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 1.4 | Definition of CIM Profiles | Classification of power system information, provided by the Network Model, following the definition from the IEC 61970-(452,453,456,457) CIM Profiles. |
| **Information Exchange** | **Relation** | **Information Description** |
| Network Model in CIM Profiles | Extended by Req. 1.1 and Req. 1.5<br>Included in Req. 2.1 and Req. 2.2 | The network model can be in *node-breaker* configuration (2 CIM Profiles: EQ and SSH), for steady-state analysis; or can be in *bus-breaker* configuration, (4 CIM Profiles: EQ, TP, SV and DY) for dynamic analysis. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 1.5 | Network Dynamic Model | Development of systems components with Modelica. Usage libraries for power systems, e.g. OpenIPSL, and libraries for power plant modeling, e.g. ThermoPower [69] |
| **Information Exchange** | **Relation** | **Information Description** |
| Dynamic Simulation Mathematical model | Extends Req. 1.4<br>Included in Req. 2.2 | They contain parameter, variable definition and the equations that describe the model. The parameter definition uses Modelica meta-model constructs to specify which parameter require start values. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 2.1 | Export CIM File | Implementation and storage of CIM profile information into corresponding files. |
| **Information Exchange** | **Relation** | **Information Description** |
| CIM Profile Files | Associated with Modeling Tool<br>Extended by Req. 1.2<br>Includes Req. 1.4 | XML files that contain power systems information, plus power plant information. This information is stored in corresponding profile files. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 2.2 | Model-2-Model Transformation | Automatically generated models comprised by information models' values and mathematical equations. |
| **Information Exchange** | **Relation** | **Information Description** |
| Simulation Model Modelica files | Includes Req. 1.4 and Req. 1.5 | A set of Modelica files that contain the mathematical definition of the network model. This model is defined by parameters, variables and equations defining the dynamic behavior using power system domain and other engineering domains, e.g. mechanical domain. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 2.3 | Check Model Implementation | Check that the model does not rise any compilation errors and is well-balance in number of equations and unknown variables. |
| **Information Exchange** | **Relation** | **Information Description** |
| Simulation Model Modelica files | Associated with Modeling Tool and Model Developer Refines Req. 2.2 | The Modelica files from Req. 2.2 are revised by checking the names, values and units of the model parameters, checking the correct use of the model's components, i.e., correct instantiation of components. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 2.4 | Compose Simulation Models. | Assembly of the complete simulation model. |
| **Information Exchange** | **Relation** | **Information Description** |
| Final Simulation Model | Modeling Tool and Model Developer Extended by Req. 2.5 and Req. 2.6 | Power system model compiled into Modelica file or FMU. Additionally, the model can be completed by adding external simulation models, in FMUs or other Modelica components. |
| **Req. ID** | **Req. Name** | **Req. Description** |
| 2.5 | Import FMU | Connect a FMU block, as part of the working simulation model. |
| **Information Exchange** | **Relation** | **Information Description** |
| FMU File | Extends Req. 2.4 | An FMU file with model information and compiled code from an external model, that can be used to complement the working simulation model. FMU configuration can be Model Exchanged or Co-Simulation. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 2.6 | Generate FMU | Connect a FMU block, as part of the working simulation model. |
| **Information Exchange** | **Relation** | **Information Description** |
| FMU File | Extends Req. 2.4 | An FMU file with model information and compiled code of the complete working model or a part of it. FMU configuration can be Model Exchanged or Co-Simulation. |

### 3.2.4. Requirement diagrams for Non-Functional Requirements

Figure 11 and Figure 12 show the main non-functional requirements related with the Model Developer actor. The ***REQ_1.2.1***, which gathers the semantic information from the ISO 15926 Data Model specification, satisfies the *Multi-Domain Meta-Data* (***REQ_1.2***). The former complies with the requirement to implement *CIM/RDF semantic* (***REQ_1.3.2***). Thus, the Model Developer should be able to provide information in RDF from other information standard formats, to comply with the rules provide by the CIM. Moreover, this information must comply with the semantics provided by the CIM in its UML definition (***REQ_1.3.1***) and used within the ***REQ_1.3***. To provide detailed profile information of the network models, with multi-domain information, the *REQ_1.4* is verified by *IEC 61970 and ISO 15926 Semantics and Data Structures* (***REQ_1.4.1 and REQ_1.4.2***).

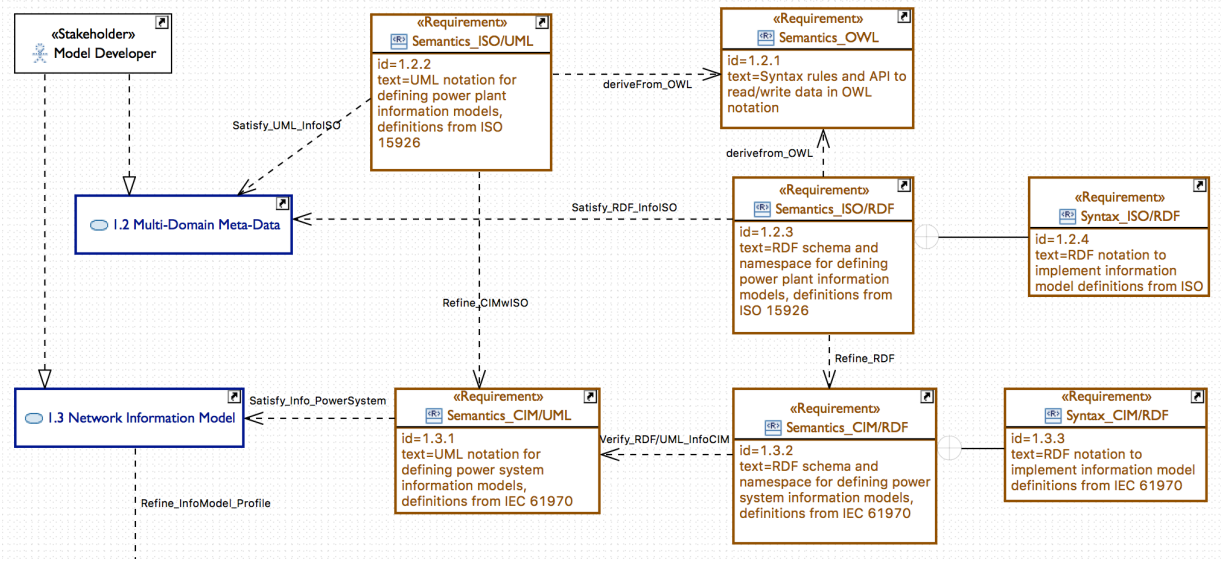


*Figure 11 Requirement specification for development of information models, related with the model developer*

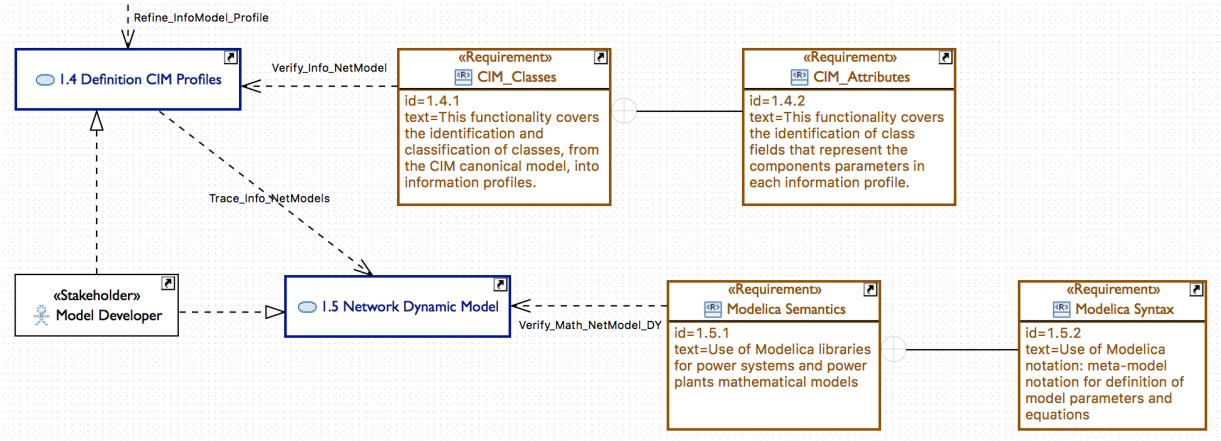


*Figure 12 Requirement specification for implementation of CIM profiles and development of equation-based dynamics models, related with the model developer*

The application of semantic information is fulfilled with the implementation of the proper language syntaxes. The Figure 13, Figure 14 and Figure 15 show the requirements related to the Modeling Tool. These requirements gather the use of the language syntaxes to represent different engineering semantics (depicted in Figure 13 and Figure 14). These syntaxes are used to write the network representations, for their further computation and execution. The functional ***REQ_2.2*** is related with the ***REQ_2.2.1*** that indicates that the model transformation process is implemented with map between information models (***REQ_2.1.1*** *and* ***REQ_2.1.2***) and equation-based models implemented in Modelica (***REQ_2.3.1***).

Figure 15 shows the technical specification related to the application of the FMI standard. The Modeling Tool should be able to provide mechanisms to import FMUs or to generate FMUs from the final model. With the former requirement, a network model can be composed by different Modelica blocks and FMU blocks. With the later, a Modelica model can be exported into an FMU so to be used as part of other simulation models, which comply with the FMI specifications.

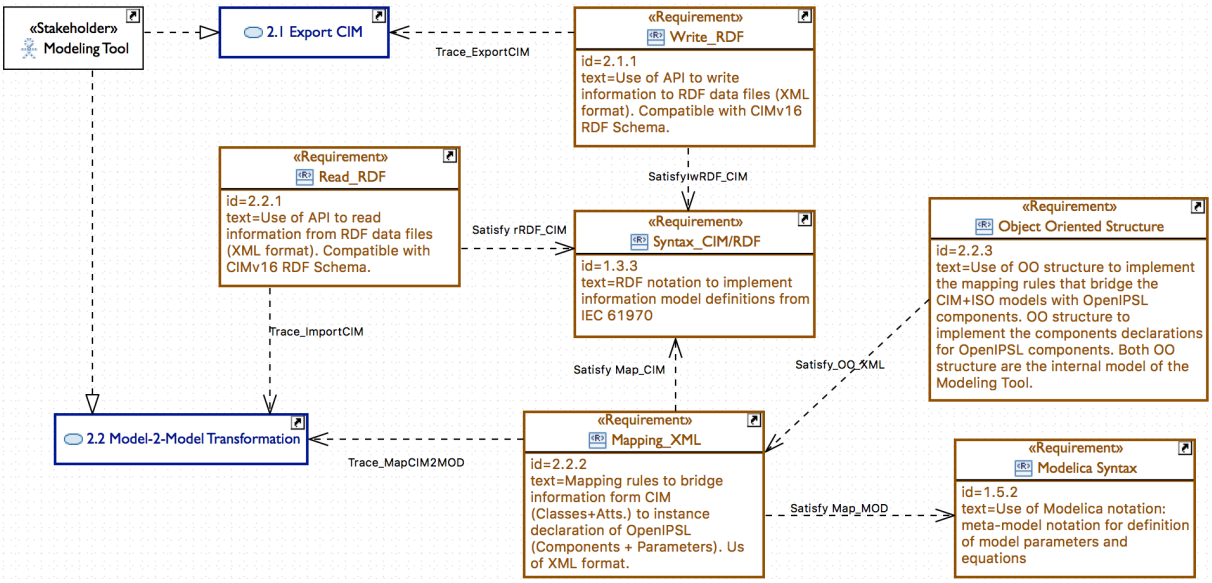


*Figure 13 Technical requirement specification for exporting information models (Req. 2.1) and automatic generation of dynamics models (Req. 2.2)*

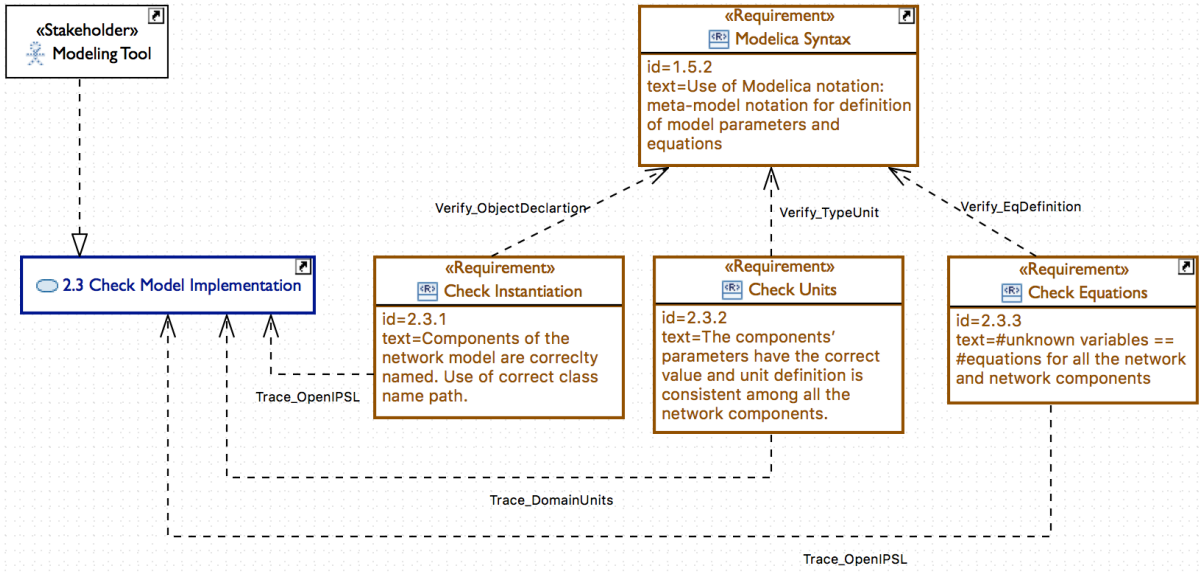


*Figure 14 Technical requirement specification related to the verification of automatic generated models.*

## 3.3. Formalization of requirement for simulation tools

This section describes the base scenario use case for **simulation tools requirements to support standard modeling languages and to allow multi-domain simulation**. The template provided by the IEC 62559 is again followed, to describe the general description and objectives. For better understanding of the requirements and functionalities, the base scenario use case is divided in two parts: Requirements involving steady-state simulations and requirements for dynamic simulations. The description of functional requirements is shown within a SysML Use Case diagram and description of the information exchange and relation between use cases. SysML Requirement diagrams will be used to gather and describe the non-functional requirements of this use case and how they are related with the functional requirements.

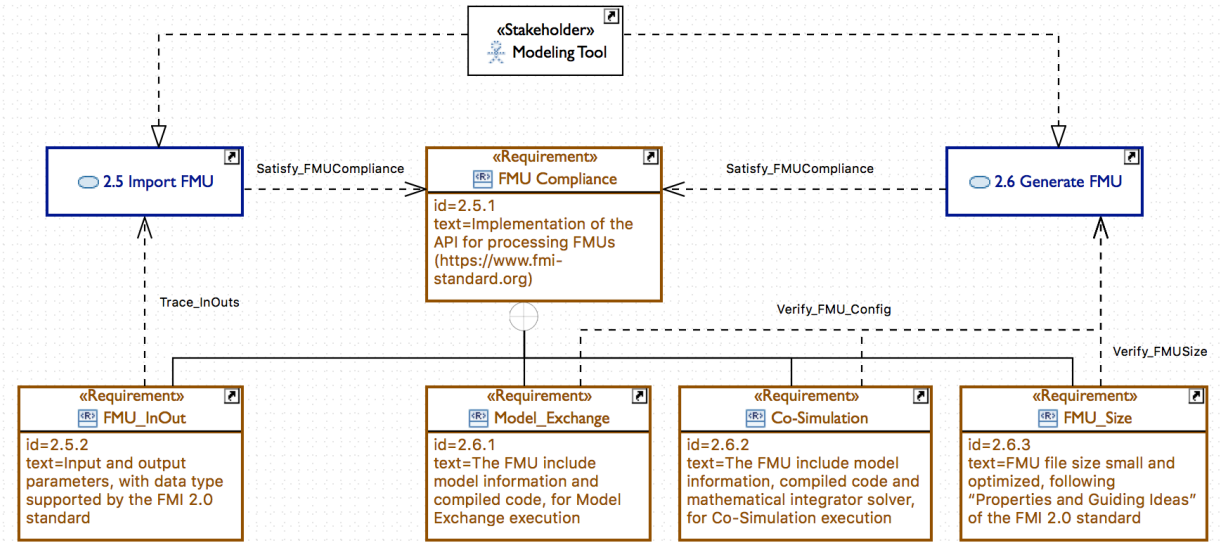


*Figure 15 Technical requirements for modeling tool, with support for FMI standard.*

### 3.3.1. Description and Objectives

This use case applies the same concept of modularity describe in Section 3.2.1. Simulation tools should be conceived as different modules that can interact with each other. The development of such tools should consider the adoption of Modelica language and the FMI standard. These main requirements generate the pre-requisites gathered in Table 5. In this way, the tool mathematical solvers could be implemented without being attached to any model representation. they also could be used to simulate model that are built from FMUs for Model Exchange (preferred), or the tool could also use the mathematical solver from the simulation model if FMUs are exchanged for Co-Simulation.

The simulation tool should include mechanism to connect FMUs [70] and define methods for steady-state analysis and time-domain. The results from these simulations could be exported to CIM/RDF formats or into binary data formats, such as the HDF5 that can facilitate the exchange for results. These specifications are designed by the diagrams in Figure 16 and Figure 17, and described in Table 7 and Table 8, respectively. These new tools should allow the monitoring and debugging of simulations. The debugging of equations can give details of status of equations at specific point on time during simulation [71]. An additional functionality to consider is the development of calibration methods to match the model parameters with available measurement data [72].

*Table 5 Pre-requisites for the development on new Simulation Tools*

| **Assumptions** |
|---|
| **Simulation Tool is able to calculate steady-state and dynamic simulations.** |
| **Pre-Requisites** |
| **A list and implementation of mathematical solvers are available, compliant with equation-based models.** |
| **Assumptions** |
| **Model compilation complies with equation-based modeling languages.** |
| **Pre-Requisites** |
| **Different Modelica compilers are available within the Simulation Tool.** |
| **Assumptions** |
| **Execution of both equation-based models and FMU-based models.** |
| **Pre-Requisites** |
| **FMI standard API for FMU execution is available.** |

### 3.3.2. Use Case Actors

The responsibilities for simulation, debugging and validation of models are identified according to the actors involved in the simulation tool: A *Test Engineer*, which is the domain expert that perform analysis on the simulation results, and the *Simulation Tool*, with capabilities to help the test engineer to execute the models. The two actors are used in Figure 16 and Figure 17 and more formal description of those actors is shown in Table 6.

*Table 6 Description of the role for the Simulation Tool Use Case Actors*

| Actor Name | Actor Type | Description |
|---|---|---|
| Test Engineer | Human | Defines the configuration of solvers to perform experiments with the models. And analyses any simulation results or validation results of the power system working model. |
| Simulation Tool | System | Implements the APIs and methods to execute the different kind of simulations with the power system working models. Implements methods for traceability of the simulations, debugging of equations or execution code of the model. And provides functionalities for validation against measurements. |

### 3.3.3. Use case functional requirements for Steady-State Simulations

The main functional requirements, their relation with actors and other functional requirements to be developed in a simulation tool, are shown in Figure 16. These requirements are designed to proposed Modelica representation and FMUs to handle steady-state simulations. Thus, requirements such as ***Req. 3.2*** and ***Req. 3.3*** gather and described functionalities related to steady-state computations. The FMI standard is also considered to use for this kind of computations, because the provide model description and binary functionalities from other power system models from other domains. Table 7 provide further details of these functional requirements.

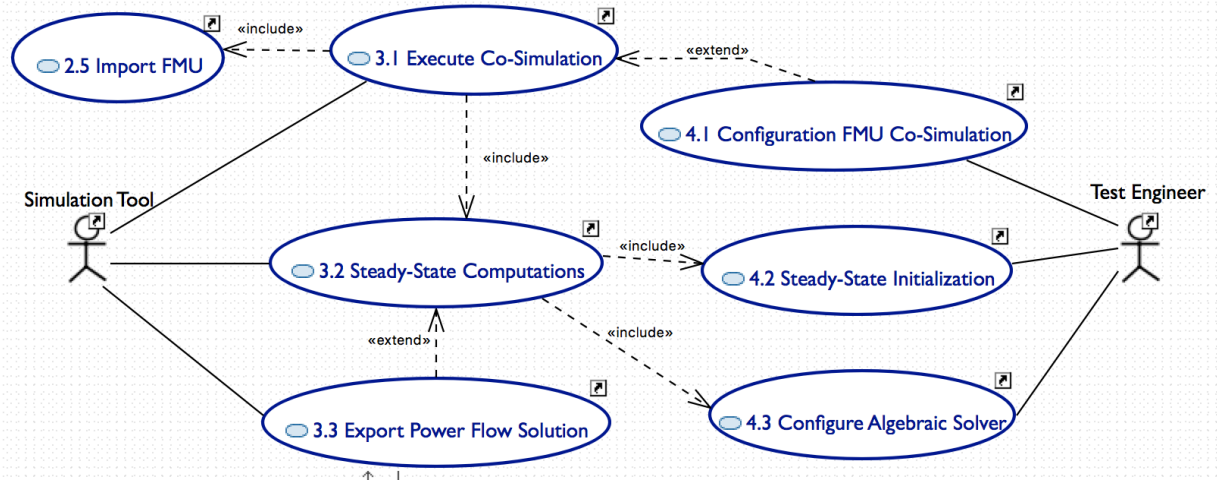


*Figure 16 Use case diagram of the main functional requirements for steady-state simulation, with FMU support.*

*Table 7 Description of power systems specific functional requirements for modeling*

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 3.1 | Execute Co-Simulation | Execution the working power system model's compiled code. The working model can be of the same modeling language or composed by different sources. |

| **Information Exchange** | **Relation** | **Information Description** |
|---|---|---|
| Simulation Results | Associated with Simulation Tool.<br>Includes Req. 2.5 and Req. 3.2<br>Extended by Req. 3.6 and Req. 4.3 | The simulation results contains the values of all the models' variables. |
| **Req. ID** | **Req. Name** | **Req. Description** |
| 3.2 | Steady-State Simulation | Execution of the mathematical solvers that compute the algebraic equations of the working power system model. |
| **Information Exchange** | **Relation** | **Information Description** |
| Working Power System model & Steady-State Results | Associated with Simulation Tool<br>Is Included in Req. 3.1<br>Includes Req. 4.1 & Req. 4.2 | Produces results of simulation represented with scalar values for all the variables and unknown variables. After execution, the simulation model updated with steady-state values of current, voltage, power and angle at each network node. |
| **Req. ID** | **Req. Name** | **Req. Description** |
| 3.3 | Export Power Flow Solution | Enhancement of power systems information model using the information associated with engineering, construction and operation of process plant, represented by the ISO 15926. |
| **Information Exchange** | **Relation** | **Information Description** |
| Power Flow results within CIM SV Profile File | Associated with Simulation Tool<br>Extends Req. 3.1, complements Req. 3.2 | The Information model is updated with the CIM State-Variable Profile complying with the IEC 61970-(301,442) CIM packages. |
| **Req. ID** | **Req. Name** | **Req. Description** |
| 4.1 | Configuration FMU Co-Simulation | Set-up of algebraic solvers and integration methods, complying with FMI. |
| **Information Exchange** | **Relation** | **Information Description** |
| FMU Solver configuration | Associated with Test Engineer<br>Extends Req. 3.2 | Configuration of the solver parameters for steady-state or dynamic simulation, provided by the solvers embedded in FMUs that Co-simulation configuration. |
| **Req. ID** | **Req. Name** | **Req. Description** |
| 4.2 | Steady-State Initialization | Initial guess values provided to the network components. |
| **Information Exchange** | **Relation** | **Information Description** |
| CIM SSH Profile | Associated Test Engineer and Modeling Tool<br>Included in Req. 3.2 | These values are provided by the CIM SSH Profile. They refer to initial guess values prior a power flow calculation (or similar steady-state method). Test engineer modified those values in the working simulation model. |
| **Req. ID** | **Req. Name** | **Req. Description** |
| 4.3 | Configure Algebraic Solver | Provide the solver parameters to compute the algebraic equations of the working power system model. |
| **Information Exchange** | **Relation** | **Information Description** |
| Tool solver configuration | Associated with Test Engineer<br>Extends Req. 3.2 | Configuration of parameters for steady-state solvers, which can interact with Modelica models. |

### 3.3.4. Use case functional requirements for Dynamic Simulations

Time-domain simulations (***Req_3.4***) are the main dynamic analysis considered in this study. For the time-domain model simulation scenario (see Figure 17 and Figure 18), simulation tools must provide means for visualization. *Debugging a simulation* (***Req_3.6***) help to trace possible errors in the prescription of the model equations. *Parameter Calibration* (***Req_3.8***) with measurements is useful for the tuning of parameters that might affect the behavior of the model and that could be fitted to better represent the real system behavior. Notice that the *Initialization of Dynamic Variables requirement* (***Req_2.7***) has been associated with Test Engineer and Modeling Tool actors. This is because the initialization of the simulation models' variables can be solved, also, from a CIM-compliant modeling perspective [73]. Table 8 and Table 9 show the description and information exchange for the functional requirements from Figure 17 and Figure 18.

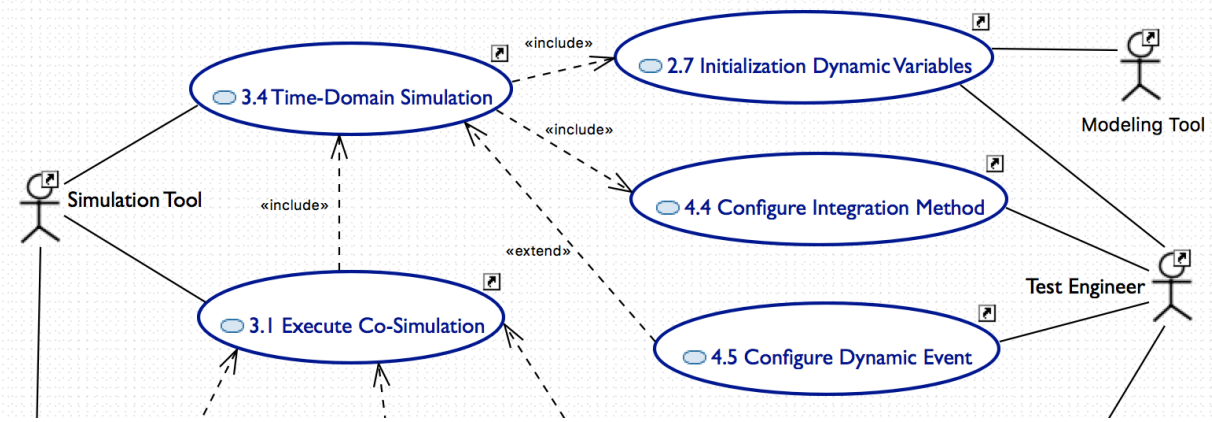


*Figure 17 Use case diagram with the main functional requirements for dynamic simulation, with FMU support. Req. 2.5 and Req. 4.1 from the Figure 2 and Figure 7 have not been included in this diagram to avoid redundancy.*

*Table 8 Description and information description of the functional requirements from Figure 8*

| **Req. ID** | **Req. Name** | **Req. Description** |
|---|---|---|
| 2.7 | Initialization Dynamic Variables | Initial guess values provided to the network components. |
| **Information Exchange** | **Relation** | **Information Description** |
| CIM DY Profile | Associated with Modeling Tool & Test Engineer<br>Is Included in Req. 3.3 | These values are provided by the CIM DY Profile. They refer to initial guess values of the components' dynamic parameters. Test engineer can modify those values in the working simulation model. |

| **Req. ID** | **Req. Name** | **Req. Description** |
|---|---|---|
| 3.4 | Time-Domain Simulation | Execution of the mathematical integrator methods that compute the DAEs of the working power system model. Simulates the behavior of the model's components. |
| **Information Exchange** | **Relation** | **Information Description** |
| Working Power System model & Time-Domain results | Associated with Simulation Tool<br>Is Included in Req. 3.1<br>Includes Req. 2.7, Req. 4.4 and Req 4.5 | The working dynamic simulation model, updated with a fault model or parameter changes that affect the model dynamic behavior. Produces results of simulation represented with vector or array of values (signals) for all the variables and unknown variables. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 4.4 | Configure Integration Method | Provide the integrator method parameters to compute the DAEs equations of the working power system model. |
| **Information Exchange** | **Relation** | **Information Description** |
| Tool solver configuration | Associated with Test Engineer<br>Extends Req. 3.2 | Configuration of parameters for time-domain integration solvers, which can interact with Modelica models. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 4.5 | Configure Dynamic Event | Add to the model extra equations or extra component that models a discrete event affecting the dynamic behavior of the system. |
| **Information Exchange** | **Relation** | **Information Description** |
| Model of a discrete event. | Associated with Test Engineer<br>Extends Req. 3.3 | The discrete event model can represent a change on the set point of regulating equipment, injection of noise in components (for responses on stochastic behavior), or the model of a fault produced in lines or connection points. |

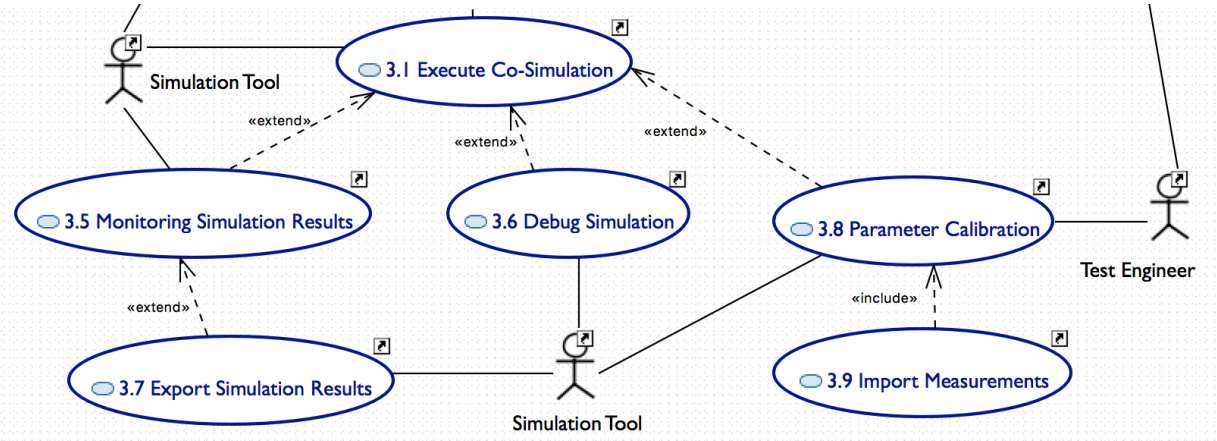


*Figure 18 Use case diagram with the main functional requirements for traceability of dynamics simulations and model validation, with FMU support. Req. 3.2 and Req. 3.3 from the Figure 7 and Figure 8 have not been included to avoid redundancy.*

*Table 9 Description and information description of the functional requirements from Figure 9*

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 3.5 | Monitoring Simulation Results | Create graphical or text reports showing the results form executing the models both steady-state and dynamic simulations. |
| **Information Exchange** | **Relation** | **Information Description** |
| Simulation Results files. | Associated with Simulation Tool<br>Extends Req. 3.1<br>Extended by Req. 3.7 | Results in scalar or array form (signals) that can be displayed in plots or text reports.<br>Results can be presented off-line (after simulations), or on-line (during simulations) |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 3.6 | Debug simulation | Check status of equation performance at any step of the simulation, as well as the execution time performance and robustness in initialization process. |

| Information Exchange | Relation | Information Description |
|---|---|---|
| Models' equations | Associated with Simulation Tool<br>Extends Req. 3.1 | The model's equations are traced by the compiler, which provides, at each simulation time step, the equation and parameters that are being solved or executed |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 3.7 | Export Simulation Results | Store the results form executing the models both steady-state and dynamic simulations, using a standard data format. |
| **Information Exchange** | **Relation** | **Information Description** |
| Simulation results | Associated with Simulation Tool<br>Extends Req. 3.5 | Results in scalar or array form (signals). All the variables and parametres are stored in files. The information is classified by component. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 3.8 | Parameter Calibration | Modification of the dynamic simulation model's parameters, for correct mathcing with reference measurements. |
| **Information Exchange** | **Relation** | **Information Description** |
| Model information | Associated with Simulation Tool<br>Extends Req. 3.1<br>Includes Req. 3.9 | The parameters and variables of all the components of the model. They can be modified to fit the behavior of the reference model. Reference model information from other simulation tools or measurements. |

| Req. ID | Req. Name | Req. Description |
|---|---|---|
| 3.9 | Import Measurements | Signal measurements from measurement units connected to the real power system. |
| **Information Exchange** | **Relation** | **Information Description** |
| Reference model results or physical network measurements | Associated with Test Engineer<br>Is Included in Req. 3.8 | Time-domain simulation results from a reference model or measurements taken from the physical network. To calibrate the working model parameters of the dynamic components and match the real behavior of the model. |

### 3.3.5. Requirement diagrams for Non-Functional Requirements

New simulation tool must provide the basic functionalities used for power system studies, either steady-state simulations (*REQ_3.2*) or time-domain simulations (*REQ_3.3*), with capabilities to support Co-Simulation Execution (*REQ_3.1*). Thus, new simulation tools with FMI support should be able to execute the basic power system simulations and comply with their requirements. The SysML diagram on Figure 19 and Figure 20 show state-of-the-art functionalities that might seem trivial from the power systems analysis perspective, but are important to consider for the development of power system analysis tools based on the modeling and simulation standards studied. Thus, it is important to consider the base values of the model, and the unit system utilized, as stated with ***REQ_3.2.2*** and ***REQ_3.2.3***.

To support multi-domain simulation, it is necessary to provide support for FMI so that a model can be composed by either power system elements and components from other domains. The technical specifications related to FMI support for Co-Simulation are synthesized in Figure 21, which shows basic state-of-the-art functionalities as

requirements, such as ***REQ_3.4.1*** for model parameter display. These non-functional requirement is related with traceability of model behavior, i.e. graphical representation of the model behavior. Moreover, the FMU and equation based modeling allow the application of functionalities such as the ***REQ_3.5.2*** that affects the debugging of the model execution.

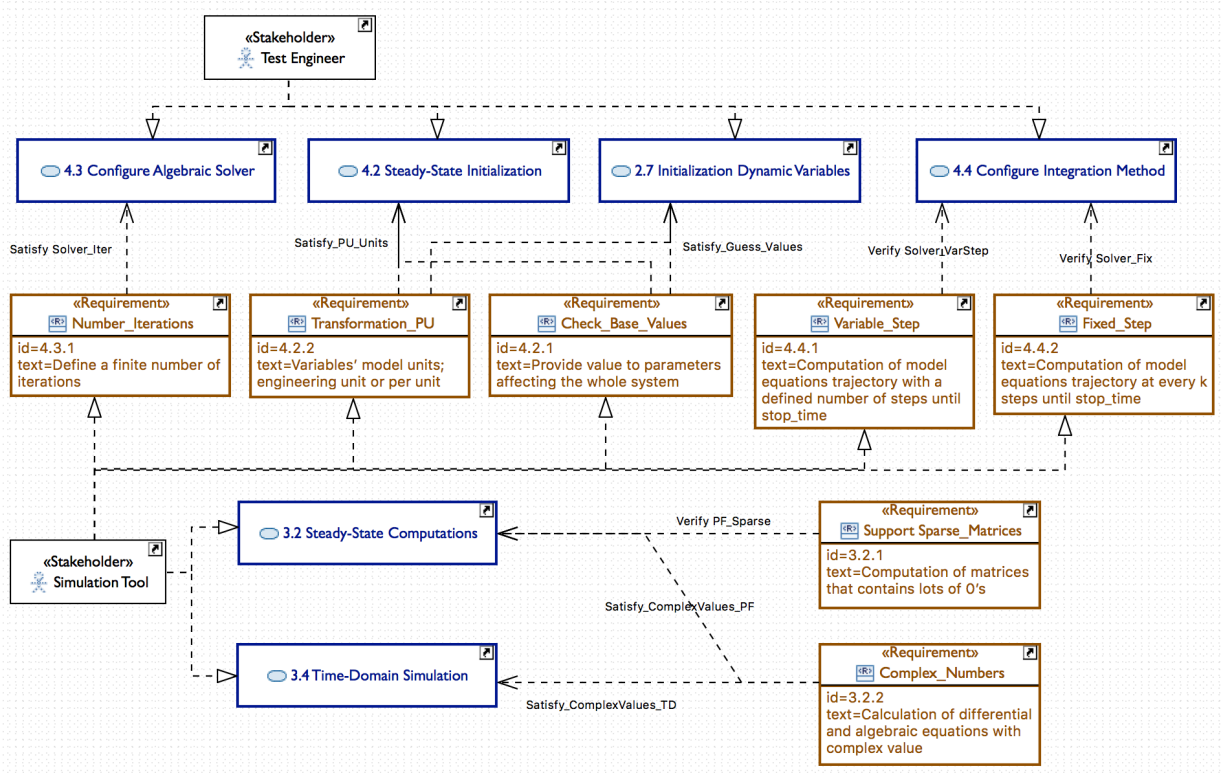


*Figure 19 Relation of functional and non-functional requirements for power system simulation tools.*

## 3.4. Summary

With the adoption of Requirements Engineering (RE), better interoperability of tools and models could also be achieved. First, the use of UML or SysML does not depend on how the system is implemented. But they define how the information, behavior to be developed to support the available standards, without strictly specify which programming language should be used for the tools implementation. Second, further application of the RE with SysML and Modelica could help to define different restrictions within the same model, provide more specific model details and improve their interoperability. Models developed in Modelica are translated into executable code by an equation-based compiler. Thus, all the equations and restrictions of the model could be able to interpret in different simulation environments.

The formal definition requirements for modeling and simulation tools with the SysML has been addressed, where functional (use cases) and non-functional (technical) requirements have been designed. The requirements have been divided into two main scenarios: a base scenario for modeling tools and a base scenario for simulation tools. The formalization of this requirements has been documented with technical specifications, following the Use Case Methodology for documentation of systems and tools specification. Part of the templated provided by this methodology is considered and it has been extended with the use of SysML Requirements diagrams as a support for the description of the requirements of each use case, which the base Use Case Methodology does not provide.

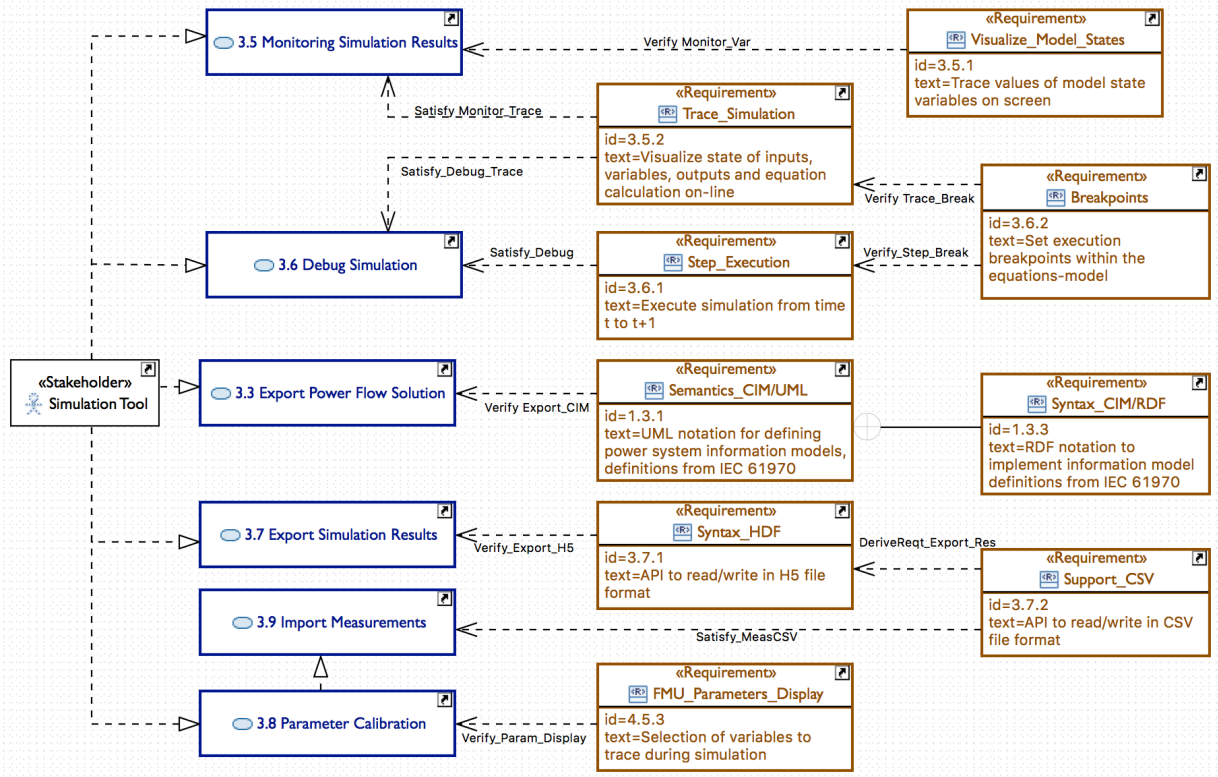


*Figure 20 Relationship of Use Cases and Requirements for power systems simulation.*

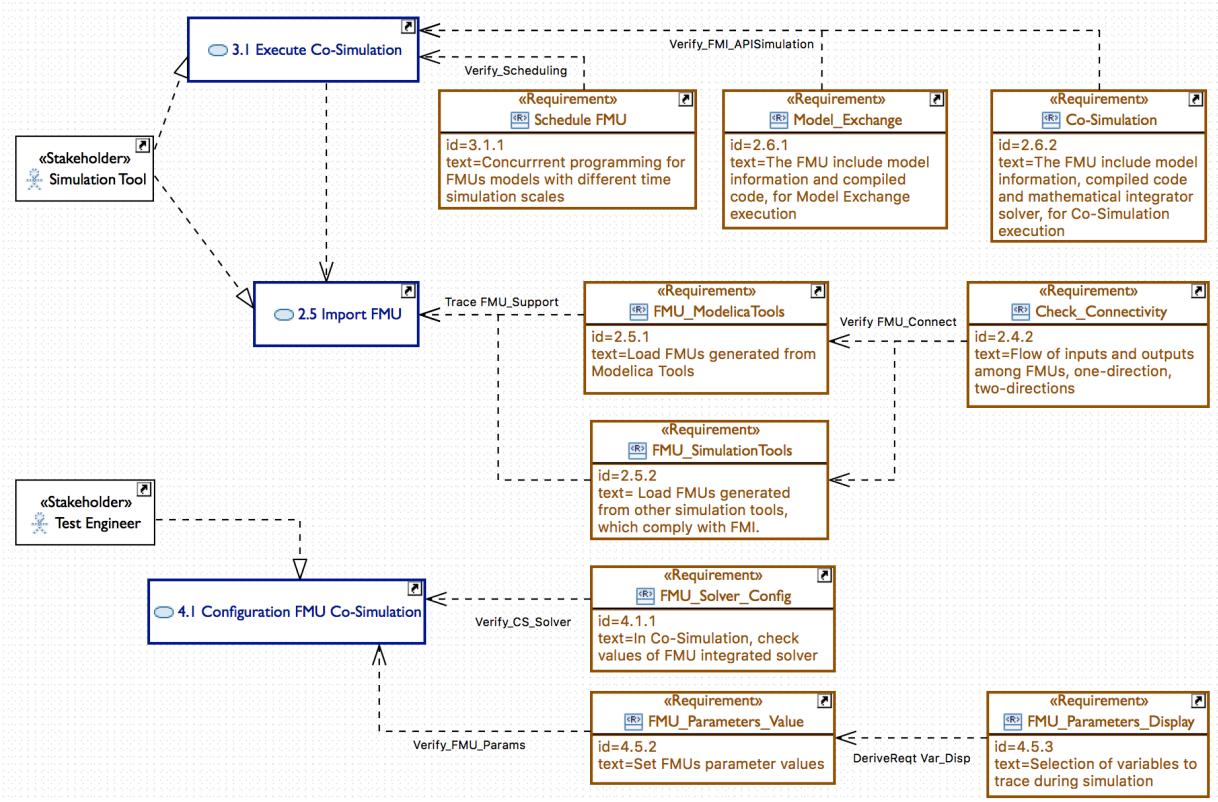


*Figure 21 Relation of functional and non-functional requirements for simulation tools supporting FMI.*

From the modeling perspective, it is necessary to provide complete information and equation-based modeling to better represent the behavior of such complex systems. In this case, it has been considered the adoption of the CIM/CGMES, which can be complemented with more detailed information from other standards, such as the ISO 15296 for power plant operation. Moreover, equation-based modeling languages have been shown as a good complement, providing the mathematical behavior that an information modeling language cannot provide on its own.

From the dynamic simulation perspective, the Modelica language (and Modelica compilers) and the FMI standard for model exchange and co-simulation have been considered as main technologies for modeling of power system dynamics models and dynamic simulation. These technologies can also motivate existing commercial tool vendors to support the definition of Modelica (equation-based) user defined models, that could replace the much simpler and ambiguous block diagram representations. Moreover, the Modelica language can be used to defined all available standard models and create additional validation models to verify the behavior of different tools.

The challenge expressed in this chapter is to encourage the development of power system tools to integrate advances from the field of M&S in system-of-systems engineering to fully comply with standard modeling languages, allowing multi-domain modeling and co-simulation of cyber-physical power systems. Further work is focused on a formalization for information-based modeling and development of new profiles. The adoption of the FMI standard implies that mathematical solvers need to adapt to handle co-simulation, parallelization of simulations, etc., which may require the use of different solver tolerances and simulation time-steps.

Finally, with the emergence of the SSP standard [70], the requirements described in this paper can be expanded to enhance the interoperability of the future cyber-physical power system tools. The SysML model of the requirements proposed in this section are available on-line at the *https://github.com/fran-jo/sysml.powersystems.framework*. This SysML model can be modified with the Eclipse Papyrus Modeling tool, and its aim is to facilitate access and so that it can be modified and expanded by other researchers in the future.

# Chapter 4

# Multi-Domain Modeling & Simulation

## 4.1. Extending the CIM Modeling Context for Multi-Domain Support

Multi-domain studies are getting more and more relevant and there is a need for modelling languages that can support and represent different static and dynamic behavior of multidomain physical systems. To comply with possible industry interoperability requirements, regarding the simulation of power system dynamic models, this work considers modeling and simulation, and comparison, of power system domain (GGOV1-based) models vs. multi-domain models of gas turbines and grid, using the Modelica equation-based modeling language. Following the non-functional requirements *Req 1.2.1* and *Req. 1.2.2* from Chapter 3 section 3.2.4, the ISO 15926 semantic information for power plant operation is studied.

From the ontological information provided by this standard, implementation designs are proposed herein. The first proposal covers a basic UML design for a more detailed description of gas turbines, from the available OWL representation, which can be integrated to the UML-based semantics proposed by the CIM standard. The second design proposal involves a full adoption of RDF semantics for its similarities with the ISO 15926 OWL's representation. It requires that the semantics subsets from the concepts of the CIM IEC 61970 and the ISO 15926 could be expressed in terms of the RDF language, by the definition's extension given by the standards IEC 61970-501. The modeling proposal of this chapter complies with non-functional requirement *Req 1.2.3* and *Req 1.2.4*. The CIM standard should be harmonized with the ISO 15926 to make it possible to make studies that utilize both information sets. Thus, the ISO 15926 information could also be parsed with the RDF schema rules proposed by the CIM.

The current CIM Profiles are considered and, in parallel, those profiles can be extended with multi-domain information. With the inclusion of the ISO 15296 information within the CIM Dynamics profile, the standard power system models can be designed using SysML. This could be used to express complete information about components, their constrains and inputs/output ports. The SysML representation can enhance the information models with more physical information. The same implementation of CIM files, with RDF format, is considered for the incorporation of new data. Thus, the implementation of new information within a power system model could benefit from the already approved CIM implementation rules, defined with the IEC 61970-552 package.

The development of power system models using the CIM is based on message construction and profiling methodology, which includes contextual modeling rules and a modeling framework. This framework is comprised by three main layers:

- The Information Model layer contains the whole CIM information with UML representation.
- The Contextual Model layer encapsulates the profile definition.
- The Syntactical Layer applies message assembly rules and implementation for information exchange [74].

In Figure 22 an extension for the CIM Canonical and CIM Layer Architecture is proposed. The extension consists in providing more detailed information of non-electric power and energy systems. The aim is to develop better representations of the dynamic component models of primary energy sources, e.g. Gas Turbines, within the power system domain.

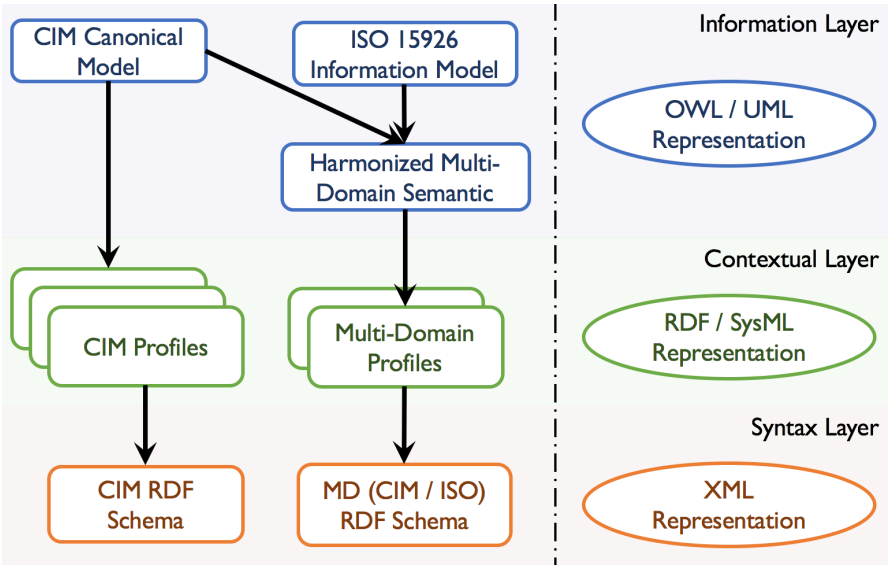


*Figure 22 Representation of the different layers for developing CIM models.*

### 4.1.1. CIM Gas Turbine Examples Models

Existing gas turbine models, such as GGOV1 [76], IEEE [77] and Rowen [78], have different levels of complexity and accuracy. Gas turbine simple models were initially preferred, primarily due to computer power and data availability limitations of the time when they were proposed (i.e. early 1990s). However, the aforementioned widely-accepted models do not employ a detailed physical representation of the gas turbine dynamics. Instead, their representation is based on abstractions in the form of logic and transfer functions [79], which are approximations of the physics governing the thermo-mechanical process, not strictly the physical law's in mathematical form.

It has been shown in [16] that more detailed models are required to incorporate the frequency dependency feature of gas turbines with the aim of performing power system stability studies during abnormal system frequency behavior; however, transfer functions approximations have shown to be insufficient for this purpose [80]. Although the CIM is currently addressing the requirement of dynamic information exchange through IEC-61970-302 and IEC 61970-457, it might lead to the exchange of ambiguous models and might not incorporate the necessary information to completely represent power plant operations and planning, whose characteristics can provide relevant impact on power systems studies.

General models such as the GGOV1 from Figure 23, is represented in CIM by a set of classes and attributes (see Figure 24), which are used by current power system modeling and analysis tools. However, the information utilized by the CIM representation could be extended if data from another engineering domain is used. Stated by the requirements *Req 1.1* and *Req 1.2* in section 3.2.3, the use of other information standard that can be used multi-domain modeling is proposed. In this case, multi-domain information considers physical characteristics from other engineering domains into the power system domain.

## 4.2. UML Extensions to Support Multi-Domain in CIM – Information Layer

The Information Layer or Information Model Layer defines an interoperability framework for semantic understanding from the different information standards that could be used for the definition of models. To enhance the CIM representation of dynamic control devices such as Turbine-Governors models, CIM semantics are taken as a reference to include new turbo-machinery data.

To illustrate this idea the GovCT1 model constitutes the entire turbine governor group of the GGOV1-based governor models is taken as an example. However, GovCT1 class attributes are restricted to only governor parameters with a simple representation of the turbine. This simple representation consists on a combination of transfer functions and limiter equation that model the turbine part of model. To create a harmonized multi-domain semantic representation of gas turbine models, some of the physical data types defined by the CIM Domain package can be reutilized. Furthermore, the CIM Domain package can be extended with ISO-based units, referred by this standard as "Property" classes (see Table 10).

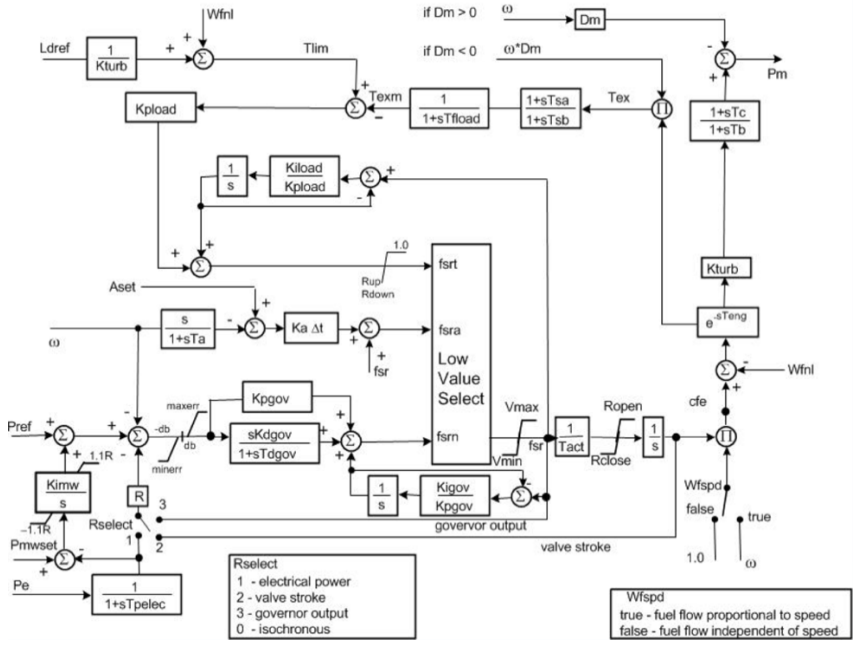


*Figure 23 General model for any prime mover with a PID governor, used primarily for combustion turbine and combined cycle units (GGOV1 Model)*

*Table 10 Combining CIM Domain with ISO Properties Information*

| Class | Unit | Source Standard |
|---|---|---|
| PU | | CIM |
| Boolean | | CIM |
| Float | | CIM |
| Seconds | | CIM |
| Rotation Speed | rad/s | CIM |
| Area | m2 | CIM |
| Absolute Pressure | Pa | ISO |
| Absolute Temperature | K | ISO |
| Density | kg/m3 | ISO |
| Mass Flow Rate | kg/s | ISO |
| Heat Capacity | J/K | ISO |

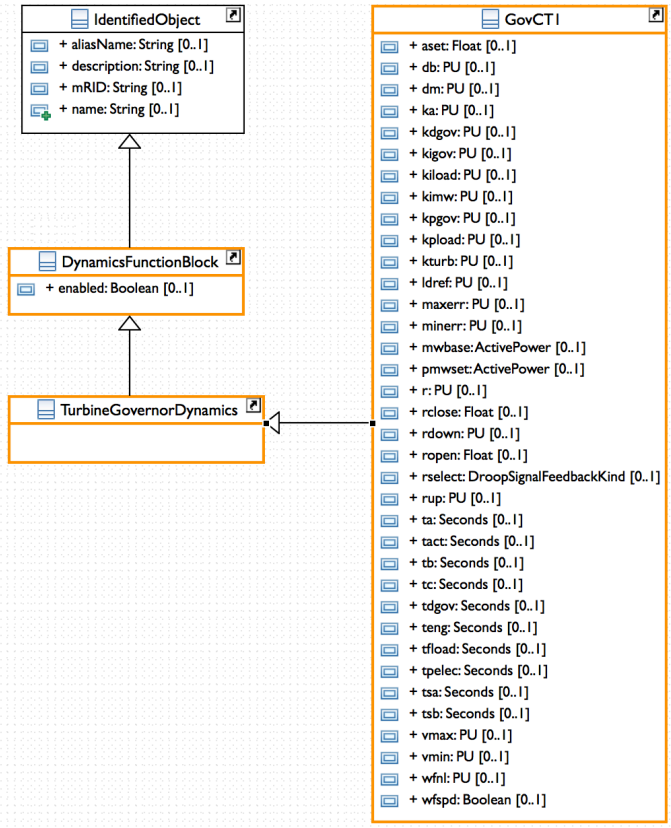


*Figure 24 UML representation of the TurbineGovernor GovCT1 model, GGOV1-based model*

The proposed ISO-UML representation has the *PhysicalObject* entity as the super-class for all physical concepts. For naming and identification purpose, this class directly inherits the attributes from the CIM *IdentifiedObject* class. Instances of other thermo-mechanical concepts and classes are thus identified by their inheritance from the *PhysicalObject*. A *TMachineryResource* is created to distinguish only those *PhysicalObject* instances used to model a gas turbine elements with flanges. In this way, classes such as *Fluid* and *Compound* are isolated from other dynamic components classes without removing their direct inheritance from *PhysicalObject* (see Figure 25 and Figure 26).

The *TMachineryResource* class inherits from the *PhysicalObject* class, instead of directly inheriting from the *IdentifyObject* class following the definition provided by the ISO standard for *PhysicalFunctionObjects* as *PhysicalObjects*. Removing the transfer function and limiter parameters from the whole *TurbineGovernor* model that represent the thermos-mechanical behavior of the turbine, we get the parameters of the *Governor* part (see Figure 27). Thus, the corresponding parameters of the Turbine model can be provided by a better parametrization within its UML model with the proposed ISO UML design (see Figure 28).

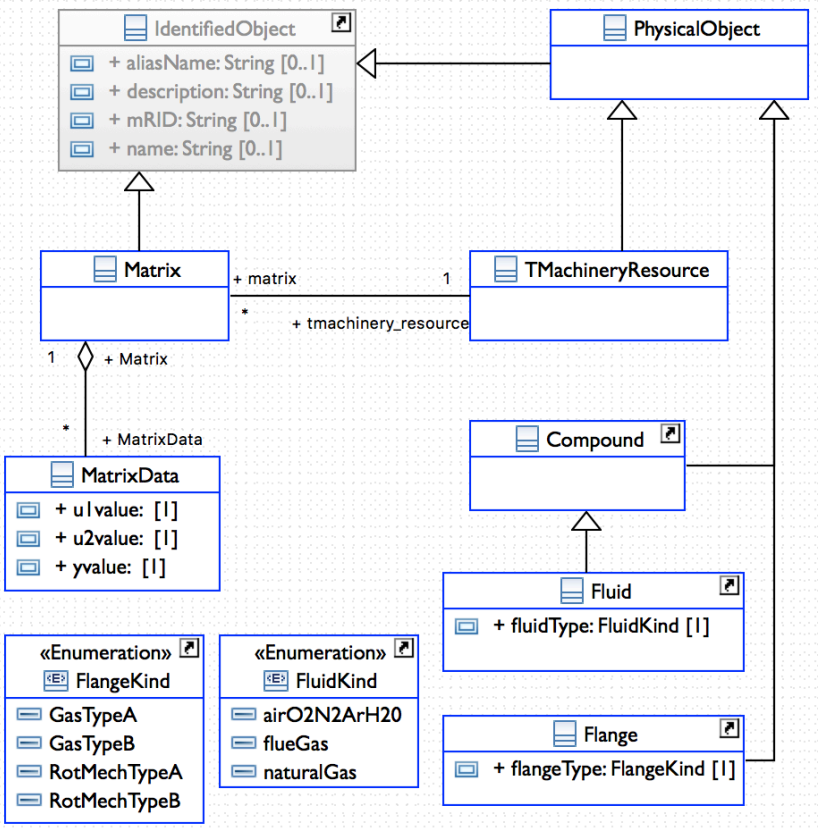


*Figure 25 UML representation of objects and data types identified from the ISO standard, which can be classified as Basic objects.*

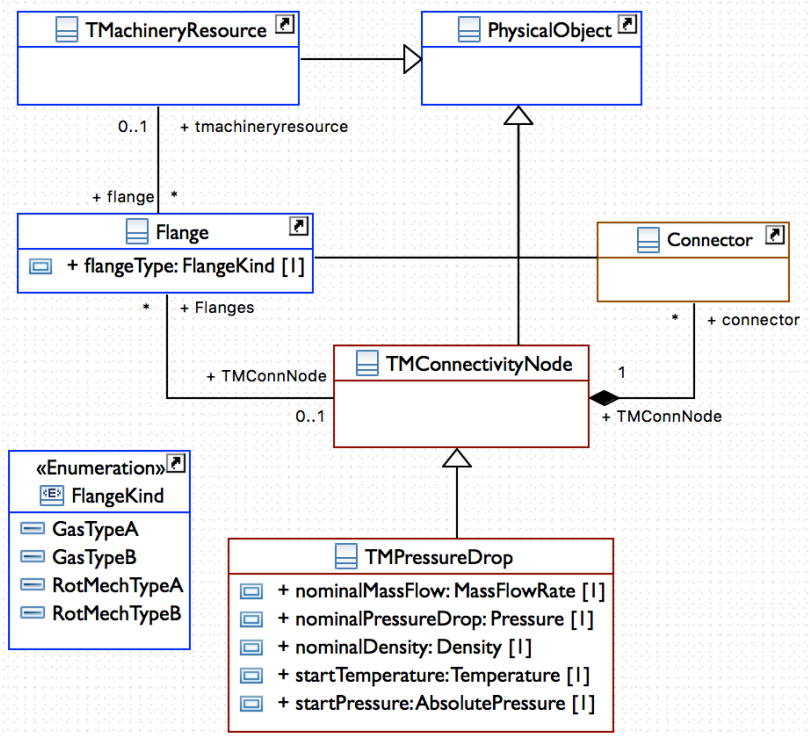


*Figure 26 UML representation of objects to model the topology of Turbo-Mechanical connections, which can be classified as Topology objects.*

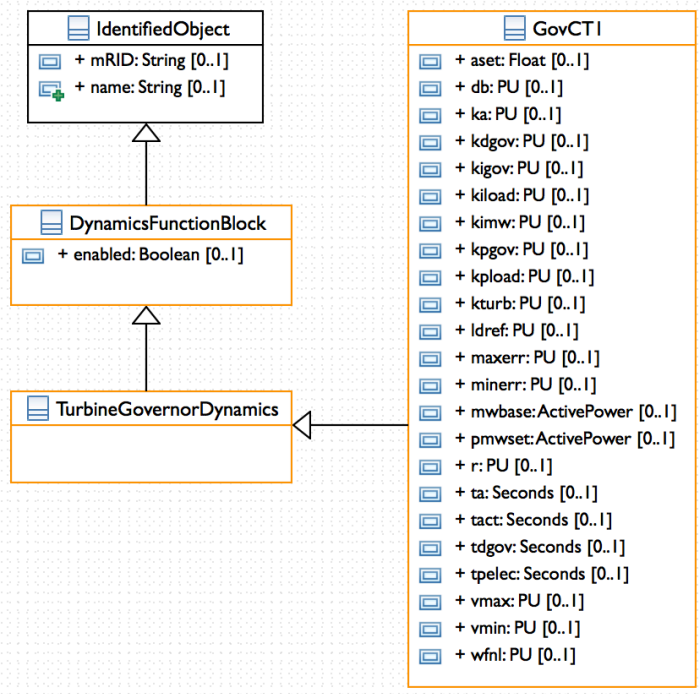


*Figure 27 UML class diagram of the CIM gas turbine model without the data related to the turbine.*

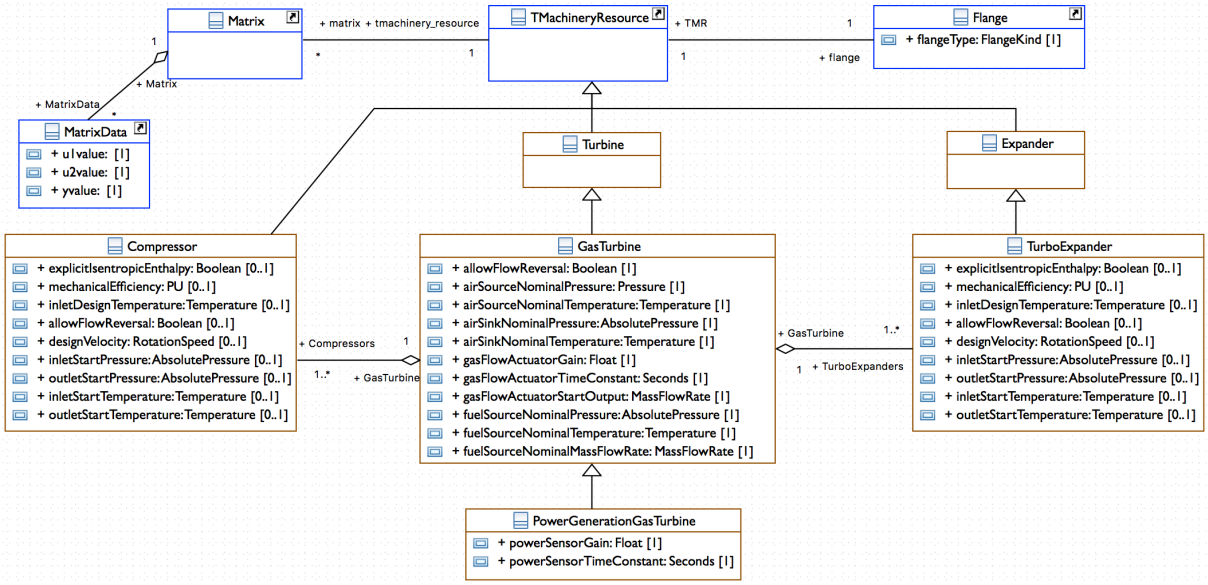


*Figure 28 UML class diagram concept for the modeling of a turbine, with data specification from the ISO 15926*

## 4.3. SysML to Support Multi-Domain Modeling in CIM - Contextual Layer

The Contextual or Business Layer is where UML methods are used to define the classification of the CIM canonical model and the information exchange requirements between applications. It is usually merged with the Message assembly layer, because the information exchanged is implemented as a message with specific information for specific power system study. However, the UML representation has some limitations for the physical representation of models and the relation that model components have with each other.

The CIM profiles organize this canonical model into subsets of classes and packages which they represent the information for a particular study. A profiling tool extracts the required information from the CIM/UML database classes and attributes and creates the reports that explain the subset of classes composing the profile of interest. Those reports are the resulting profile files. The UML representation provides a set of rules about the classes, their attributes and their relations. These rules conform a useful data base with model information.

Conventional profiling tools are able to create complete CIM profiles using UML definitions and their internal components' model implementations [80]. The final representation of those profiles is assembled with the CIM/XML/RDF schema. The RDF represent the triples of data that incorporates mechanisms to represent in better detail the power system models. However, because of the restrictions of UML for physical representation, a profiling tool that process RDF files encounters two challenges:

1) The profiling tool need to implement the physical information that constitute a power system component or network model, e.g. the most relevant parts of the ISO 15926 can be integrated into the CIM Profiles to create new data models for study the impact of gas and steam in the electricity production should be implemented implicitly within the profiling tool,
2) The profiling tool cannot represent graphical those physical characteristics using simple UML class and attribute representation, e.g. UML representation cannot define the physical connection and physical values that connects two different equipment.

The use of SysML for the graphical physical representation of systems could solve the graphical representation of the physical characteristics of components or networks models, using different engineering domain semantics, without losing information. With SysML a physical representation of any power system model could be achieved instead of a pure information model. SysML Block diagrams could represent the parameters of the CIM, with an addition of providing physical constraints and the equations that define the behavior of the model components. The physical constraints can be designed in requirements diagrams and those requirements can be related with the corresponding block, which also contains the attributes defined by the CIM classes.

Thus, those requirements can be also exchanged within the model, so they could be assembled into a RDF message and be implemented in any simulation tool. This solution could lead to the creation of new CIM profiles and the message assembly can benefit from the SysML representation by the application of Model Driven Development principles, processing each component and physical models as a complex object. This comply with requirement gathered in *Req 2.2.3* from section 3.2.4, facilitating the implementation of information assembly and model transformations.

Moreover, the SysML could be used to represent those physical details that cannot be represented by UML. Thus, new formats and technologies could be integrated within the Syntax Layer. Relevant to the definition of *Req 1.5* from section 3.2.3., software technologies such as the FMI standard, and Modelica libraries, such as the The OpenIPSL library [34] or the ThermoPower library [69], could be used to enhance the CIM for Dynamics profiles with equation-based and multi-domain models details. For example, Figure 29 shows a physical representation of the GGOV1 model in SysML. This multi-domain model combine models of typical network components with the governor model and the turbine model from the turbo-machinery domain.

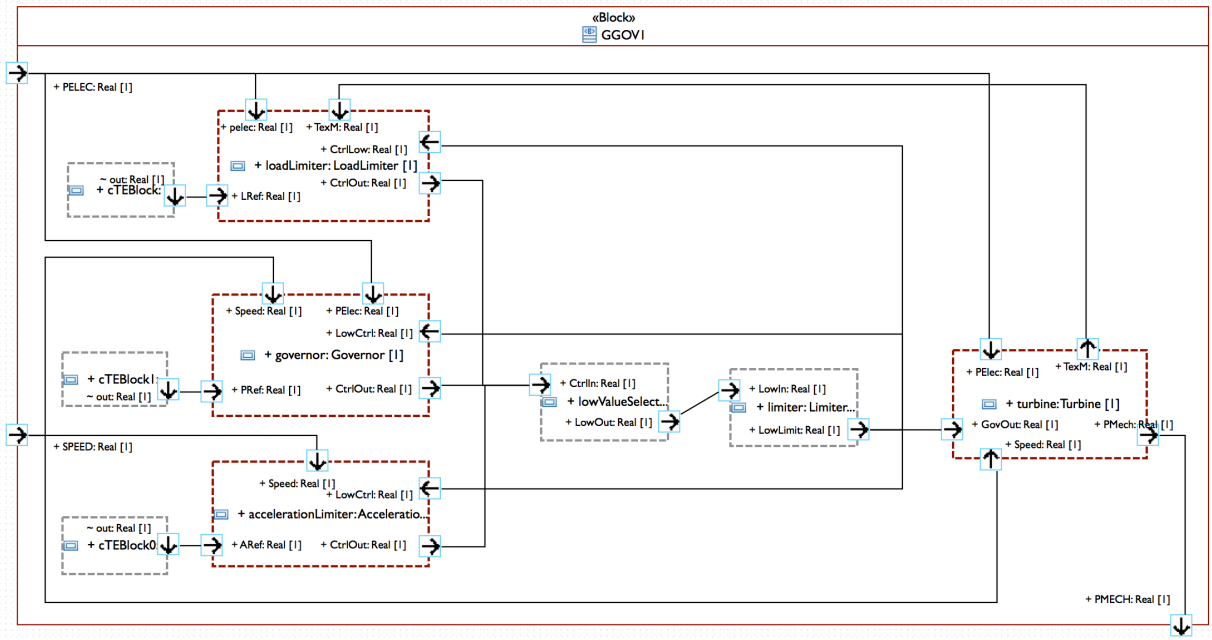


*Figure 29 Refined SysML GGOV1 governor model, where control logics and the turbine model are split in different blocks.*

## 4.4. RDF Schema to Support Multi-Domain in CIM – Syntax Layer

The CIM provides two methods for transmitting the CIM data using the XML language: the CIM/RDF/XML format for transferring the full CIM model of a power system or for transferring changes in the CIM model; and the CIM/XML format for transferring simple changes in the CIM model or add new data, as meter readings. Because the interest on creating better detailed power system models, the focus is on the first method. Its implementation rules are defined within the IEC 61970-552 package, which have been gathered into the *Req. 1.3.3* from Chapter 3 section 3.2.4.

Following the class definitions from the section Figure 25, Figure 26 and Figure 28, serialization of these classes has been implemented with the Apache JENA library [82] for developing a JAVA library, equivalent to the new ISO-UML representation of the ISO 15296 semantic information. This implementation is used to create the RDF-triple prototype model representation, shown in a graph-like representation Figure 30, which represents part of the basic definition of a Gas Turbine model.

To comply with RDF implementation of new classes under the CIM definitions, an extra package has been created with a vocabulary definition both IEC 61970 concepts and the new concepts extracted from ISO 15926 (see Figure 10). RDF and CIM namespaces have been used to guarantee that the propose enhancements comply with their naming and data structure definition. An additional namespace has been created which points to the proposed new semantic information. This namespace is used to classify the new components, which are to be designed with detailed information from the turbo-machinery domain. The resulting RDF model shows how the implementation follows the CIM object definition and the definition of unique identifiers for each object (see Figure 32). A total of six packages have been created to classify a set of new classes to be used for the proposed semantic modeling approach (see Figure 33).

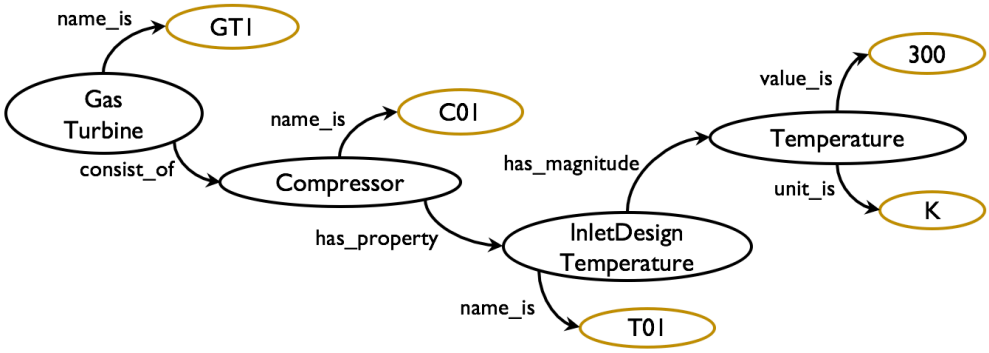


*Figure 30 Subset of the graph for the compressor model of the gas turbine corresponding to the RDF representation*

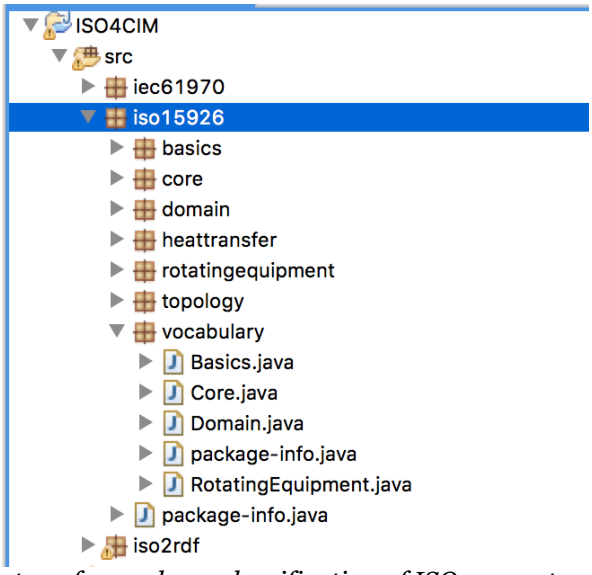


*Figure 31 Prototype for package classification of ISO concepts and vocabulary*

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cim="http://iec.ch/TC57/2013/CIM-schema-cim16#"
    xmlns:iso15="https://alsetlab.github.io/iso/15926/ISO_CIM-schema_cim16#">
  <iso15:GasTurbine rdf:about="_7c1ae276-2bb8-11b2-80e9-8acfb8612c06">
    <iso15:TMachineryResource.Compresor rdf:resource="#_7c1ae277-2bb8-11b2-80e9-
8acfb8612c06"/>
    <cim:IdentifiedObject.aliasName>GT01</cim:IdentifiedObject.aliasName>
  </iso15:GasTurbine>
  <iso15:Compresor rdf:about="_7c1ae277-2bb8-11b2-80e9-8acfb8612c06">
    <iso15:Compresor.InletDesignTemperature rdf:resource="#_7c1ae278-2bb8-11b2-80e9-
8acfb8612c06"/>
    <cim:IdentifiedObject.aliasName>CP01</cim:IdentifiedObject.aliasName>
  </iso15:Compresor>
  <iso15:Temperature rdf:about="_7c1ae278-2bb8-11b2-80e9-8acfb8612c06">
    <iso15:Temperature.unit>K</iso15:Temperature.unit>
    <iso15:Temperature.value>300.0</iso15:Temperature.value>
    <cim:IdentifiedObject.aliasName>T01</cim:IdentifiedObject.aliasName>
  </iso15:Temperature>
</rdf:RDF>
```

*Figure 32 Detail of RDF model, as a prototype for a multi-domain model combining CIM information and ISO 15926 information.*

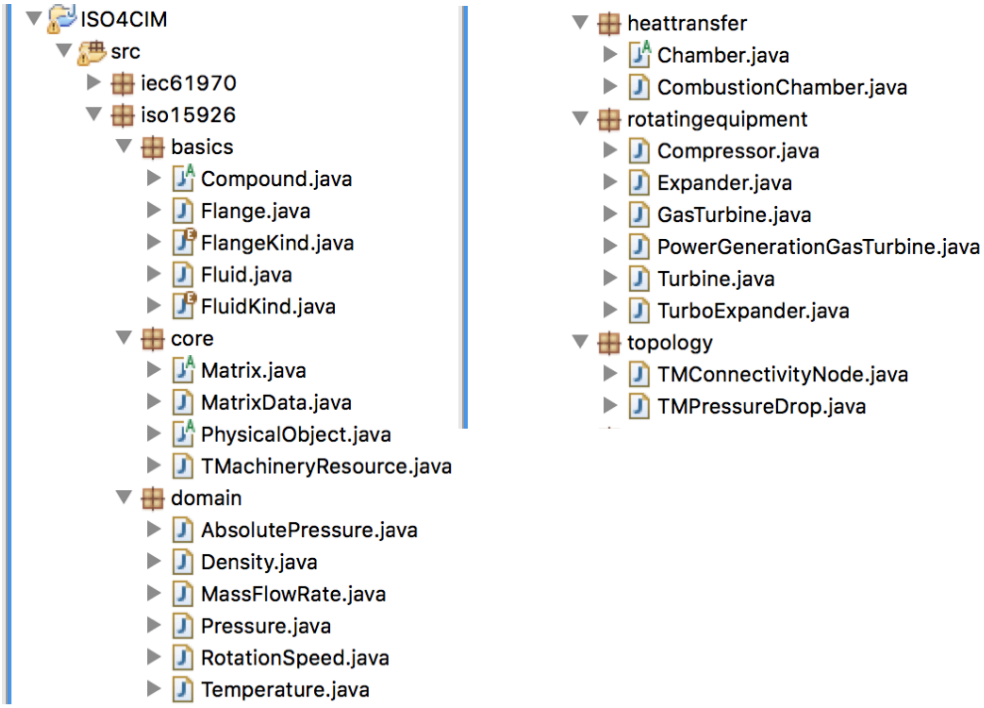


*Figure 33 Prototype implementation for package classification of ISO concepts following an Object-Oriented approach.*

## 4.5. Multi-Domain Equation-Based Modeling

This section addresses the equation-based modeling and simulation of multi-domain systems, which are pre-requisite to apply an automated model transformation method to transform the model from CIM to Modelica. Such a model transformation is conceived to have a full representation of the model. And both semantics can be used within the Syntax Layer because they are compatible with the XML representation.

In the power system domain, common model of the turbine and governor of a gas turbine is tipically described using the GGOV1 model, where the turbine part is combined with the Governor part and modeled from a power systems perspective. A better representation, in Modelica, of the GGOV1 model can be found in the OpenIPSL library, under the */OpenIPSL/Electrial/Control/PSSE/TG/GGOV1/GGOV1.mo* file. The turbine model from the ThermoPower library is used as a separate model, providing more detailed equations and parameters from the turbo-machinery perspective.

To this end, a new design of the GGOV1 model, from the OpenIPSL GGOV1 Component, was created separating each of the three controls logics forming the GGOV1 model (see Figure 34): (1) A block for the load limiter; (2) a block for the acceleration limiter; and (3) a block for the main governor part. Its control logics and turbine model are separated in different blocks. A forth block was created to represent only the turbine. Thus, the new GGOV1 model design is better represented from an Object Oriented (OO) approach, obtaining a convenient way to re-use the models when a certain study requires only the turbine and the governor, instead of having an all-in-one-simplified model with turbine controls and protections.

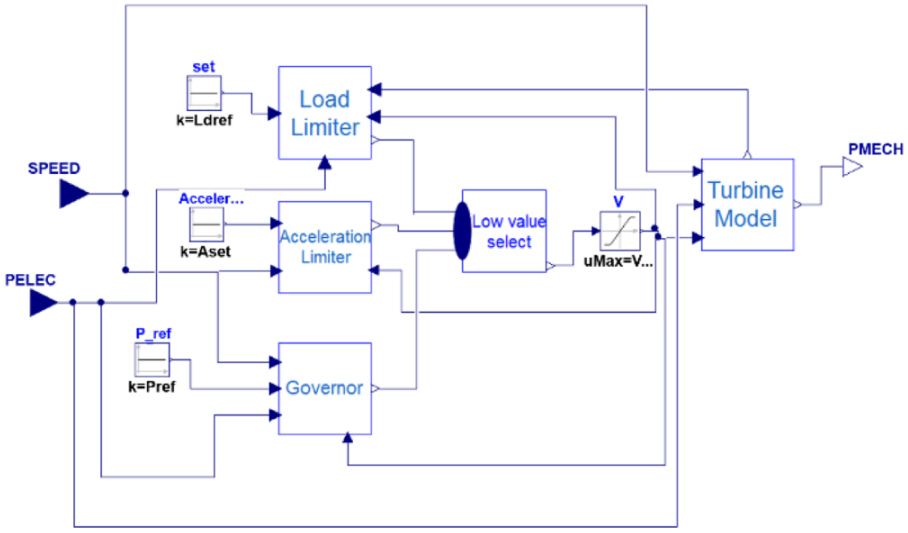


*Figure 34 Refactored OpenIPSL GGOV1 governor model, where control logics and the turbine model are split in different blocks.*

An additional interface block (see Figure 36) was created to allow the connection between the electro-mechanical generator model and the detailed gas turbine model. It provides an implementation of the *TMConnectivityNode* class using the Flange and Connector classes of the proposed topology information model presented in Section 4.4. This connection relates the rotational mechanics (flange internal variables) of the gas turbine model with the generator mechanical power and speed, as shown in the code section of Figure 36.

```
1.  model TM2EPConverter "Interface between OpenIPSL generators and ThermoPower
    gas turbine models"
2.      import Modelica.Constants.pi;
3.      outer OpenIPSL.Electrical.SystemBase SysData;
4.      parameter Integer Np= 2;
5.      parameter OpenIPSL.Types.ApparentPowerMega S_b= SysData.S_b "System base
    power";
6.      Modelica.Mechanics.Rotational.Interfaces.Flange_a shaft;
7.      Modelica.Blocks.Interfaces.RealOutput PMECH;
8.      Modelica. Blocks.Interfaces.RealInput SPEED;
9.      Real omega_e;
10. equation
11.     omega_e= der(shaft.phi) * Np;
12.     SPEED= omega_e / (100 * pi) - 1;
13.     PMECH= der(shaft.phi) * shaft.tau / (S_b * 1e6);
14. end TM2EPConverter;
```

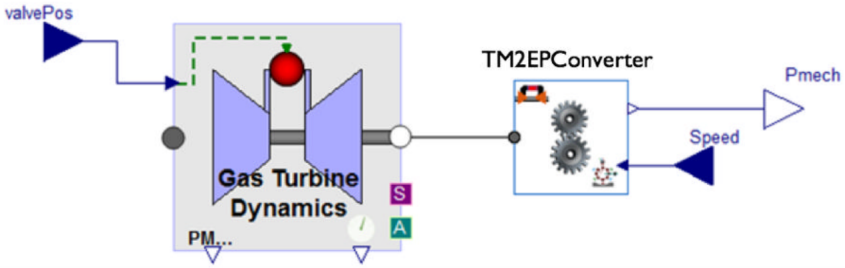


*Figure 35 Modelica graphical representation of the Generation group depicted in Figure 35*

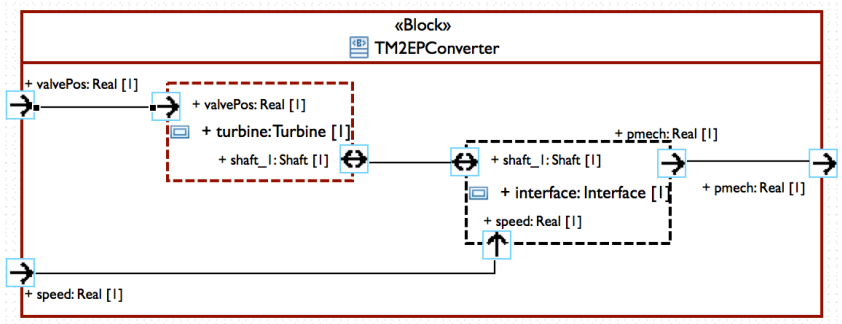


*Figure 36 Gas Turbine model from ThermoPower library, connected to the interface block. Its code is in the top of the figure.*

### 4.5.1. Simulation Results

The multi-domain model consists in Figure 37 of a "generation group" template (which it can be used for different sets of component configurations) connected to the grid model, the ThermoPower turbine and a governor block model from the power system domain. The governor block uses the implementation from Figure 34, thus its turbine model is substituted by and connected with the physical model of a gas turbine from the ThermoPower library (see Figure 35).

The multi-domain benchmark model in Figure 37 is simulated to study the dynamic behavior of the grid model when subject to different modeling detail of the turbine. The time-domain responses are to those from the same model described using components only of the power system domain. The responses are obtained for a load change event. A simulation of 100 seconds was performed on both the multi-domain and power system-only models. The active power of the load was increased by 0.2 p.u. after 30 seconds of simulation, and was set back again to the original value after 20 seconds.

Due to the mechanical model of the turbine, the multi-domain model represents better the dynamic behavior of the mechanical power, i.e. the transient response of the mechanical power is more accurate than the simulation from the equivalent power system-only model (see Figure 38). The responses of the frequency (see Figure 39) and the electrical power (see Figure 40), show the same period but with lower amplitude, which indicates a faster stabilization of the both signals, using the multi-domain model. The presence of mechanical-domain model parameters and equations within the model give more insight information of the physical behavior of the model components.

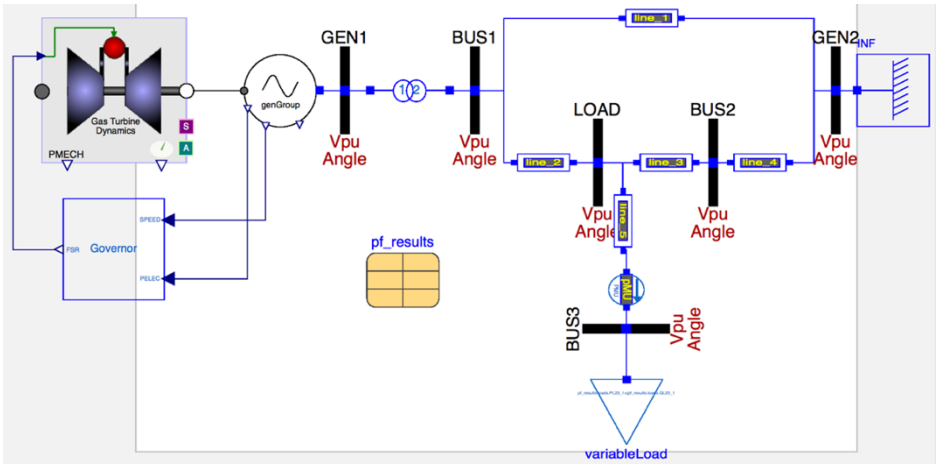


*Figure 37 Multi-domain SMIB model with GGOV1-based governor model and a detailed Gas Turbine model.*

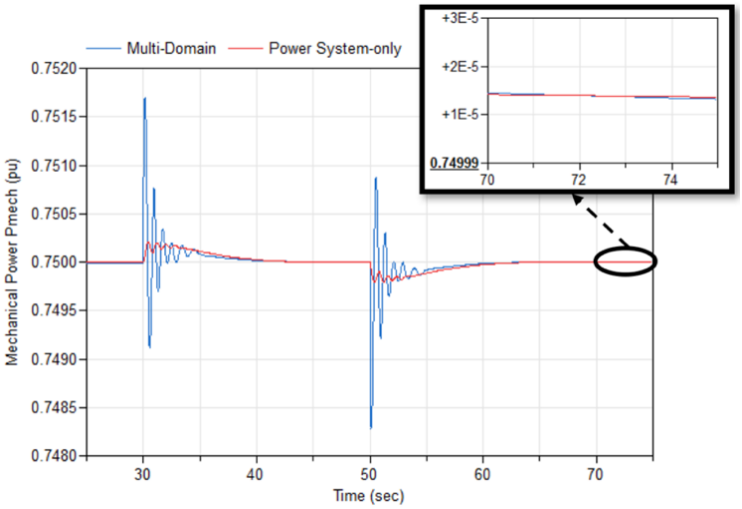


*Figure 38 Comparison of the mechanical power response for both multi-domain and power systems-only model.*

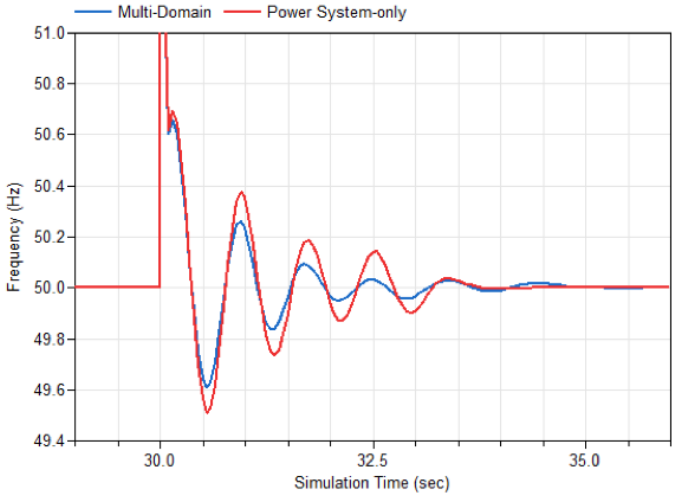


*Figure 39 Comparison of the frequency response for both multi-domain and power systems-only model.*

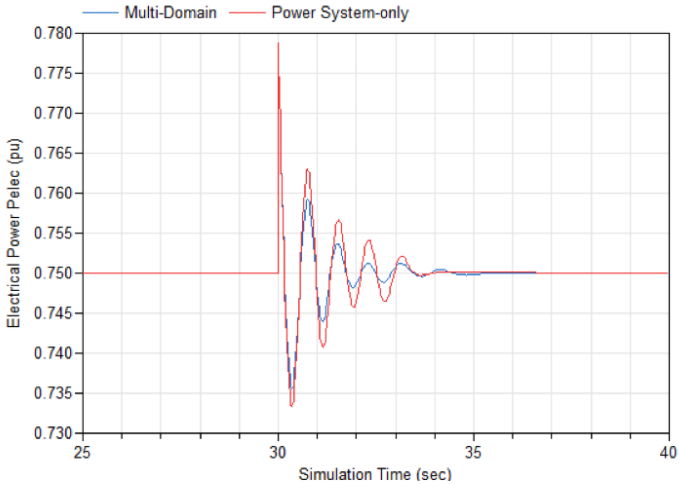


*Figure 40 Comparison of the electrical power response for both multi-domain and power systems-only model.*

## 4.6. Summary

This chapter describes a proof-of-concept on the use of Modelica libraries to include turbomachinery features in multi-domain power systems for both the semantic information modeling and dynamic simulation of these models. The value of this work is that, while the first part results in making available all necessary semantic modeling information of multi-domain representation in CIM; the second part provides all necessary equation-based models in Modelica. Thus, the approach in [73] can be rapidly applied for multi-domain model transformation from an extended CIM model to a Modelica-based simulation model.

Semantic information and data provided by the ISO 15926 has been analyzed with the aim of proposing a SysML and RDF information modeling approach that can cope the lack of dynamic information (i.e. parameters), within the CIM/CGMES, while being able to refer to dynamic models of physical systems, such as Turbines. A modification of the context layer for CIM modeling has been proposed, to incorporate further modeling resources, thereby expanding the capabilities of CIM. The SysML can represent in more detail the characteristics and physical connections of components, while preserving certain level of protection for the user-defined models, to make it more compatible with current needs of model information exchange.

To prove that combining semantic information from different domains within a unique simulation model could be suitable for power systems dynamic studies, a multi-domain model has been derived, to allow detailed representations of gas turbines within power grid simulations. Although, open source software libraries (e.g. the OpenIPSL and the ThermoPower) are not widely used in the industry, it is still possible to appreciate certain differences in the responses of typical variables of interest when compared to those of a commonly used model for power system analysis. For instance, the GGOV1-based turbine model is not dependent on the shaft speed and therefore, the changes on the mechanical power are due to the governor response.

A simple network model has been developed as a proof of concept for this kind of modeling and simulation methodology. Although, the models are simple (due to the lack of publically available information), the turbine modeling approach provides a framework for future studies with multi-domain models in power systems. Thus, a better model that includes among other things the valves dynamics is desired as it would allow a better modeling of the fuel mass flow rate behavior. These kind of turbo-mechanical dynamics is not present in power system models, such as the GGOV1 turbine governor model.

# Chapter 5

# Model-to-Model Transformation

## 5.1. Why do we need Model Transformation?

In Chapter 2 section 2.4.5, the problem of dynamic information exchange is introduced. Current practices on sharing power system dynamic models are based on parameter exchange and block diagrams. Thus, interoperability can only be unassailable though the comparison of several implementation results. The main issue with this approach is that the exchange of such information cannot guarantee unambiguous model exchange between different simulation tools [19]. The dynamic model exchange approach has two main back draws. First, dynamic models for different components are not consistent through platforms due to simplifications, modelling philosophy and assumptions. Second, conventional block diagram modelling forces users to share only parameters of models with predetermined structure, the model's mathematical representation is therefore not shared explicitly. This leaves open to interpretation how the actual implementation of the models is carried out in each application. Therefore, two different model implementations of the same specification and parameters/block diagram will be inconsistent and will produce inconsistent simulation results.

Relevant to the lack of mathematical description on the current definition of the CIM and CGMES, the concept of model transformation is introduced, so it can be used to bridge gap between information modelling and equation-based modelling [73]. In this chapter, a formal description of a Model-to-Model (M2M) transformation process, which is used to automatically generated power system dynamic simulation models, is presented. This process covers the functional requirements *Req. 2.2* and *Req. 2.7* defined in Chapter 3 section 3.2.3 and section 3.3.4. Following these requirements specification, modeling tools should be able to produce a complete simulation model with available APIs modeling standards. An example of model transformation addressing this issue can be find in [74], where Modelica is linked with the Building Information Models (BIM) for Building Energy Performance Simulation.

The M2M transformation process has the advantage to automatically generate complex network models. It is implemented for the use of CIM & CGMES definitions, as the source information models, to obtain the target equation-based definition of network models, in Modelica, which provide a standard interpretation and implementation of the dynamic simulation model. Mapping rules between two modelling languages are defined and they are implemented with a meta-model structure that processes the most relevant keywords from those languages. By the definition of the non-functional requirements *Req. 1.3.3* (from Chapter 3 section 3.2.4) the syntax from the CIM semantics is processed to read the relevant values from the source information model. By the non-functional requirement *Req. 1.5.2* (from Chapter 3 section 3.2.4) The syntax from the Modelica language and the OpenIPSL library is used to create the instances of the OpenIPSL components.

### 5.1.1. The problem of Initialization of Power System Models

Components models within the OpenIPSL are used to build dynamic network models which contain model variables or states that evolve over time. Not assigning an initial value to algebraic states or using default values (0 or no value) may result in divergences of the

initialization algorithm. These values should either be written manually when designing the model or computed through an external routine. These considerations are specified by the requirements *Req. 2.7* and *Req. 3.4* in Chapter 3 section 3.3.4.

To initialize a Modelica model build by OpenIPSL components, initial guess for all algebraic, continuous and discrete variables need to be provided by a power flow solution results. The current values in buses, lines and loads are set to zero and are determined by the static analyzer at the initial step of the simulation. Once the model is built with all the starting values, the first step of the Modelica compiler is to correctly determine the initial steady state without initial values for currents. Dymola and OpenModelica manipulate symbolically the initialization problem and generate analytic Jacobians for non-linear problems.

From the CIM representation of a power system, Modelica code can be generated and the model variables can be properly initialized with the power flow solution stored in the corresponding CIM model. The M2M transformation process is adopted to address the problem on initialization of power system models in Modelica (see section 2.3.2), to give the initial conditions of the power grid model for the dynamic simulation from the CIM Information Model, with the minimum participation of an engineer. The M2M transformation process is applied for automatically generating power system models and to satisfy the functional requirement *Req. 2.7* for initialization of dynamic values.

## 5.2. CIM and Modelica Syntaxes for Mapping rules

Exogenous model transformations, i.e. model transformation where the source modeling language is different from the target modeling languages [58], is implemented by the development of mapping rules. These rules gather the key attributes from both modeling languages' syntaxes considered in the model transformation process. In this section, the relevant features of both CIM and Modelica are studied, for their use on defining the mapping rules between OpenIPSL components and their equivalent CIM models.

### 5.2.1. CIM RDF Schema

The data base of components and connections represented by the CIM data model are stored in XML files, and its syntax is implemented, with the Resource Description Framework (RDF) language, by the specifications from the IEC 61970-501 standard [74]. Within the RDF schema, objects and relations between them are stored in graph-like data structures (see Figure 41 and Figure 42). This kind of representation makes it easy to understand the parameters and the associations that CIM classes have between them. Moreover, serialization, checking and making files for storing topology changes are easier to interpret by CIM-based XML processing tools, such as the CIM Spy [84] that can read graph-like structures, or the CIMTool, for validation of CIM models and profiles [85].

In case of Figure 41, the CIM class ***SvPowerFlow*** has properties p and q and a third property that represents an association with a CIM class ***Terminal***. The *SvPowerFlow* class is used to store the power flow solution for P and Q and the connection terminal to which these values are associated with. In its syntax, the CIM uses the ***rdf:ID*** parameter that contains a unique ID, used to identify the objects from the CIM model. CIM classes use the ***rdf:resource*** as a pointer to another object within the CIM model. Its value is another rfd:ID used to identify the other class related to the current class. Those two attributes are defined with the standard RDF schema definition. To identify the electrical semantic names, the CIM uses its own RDF schema, defined by a namespace of the current version of the CIM, e.g., http://iec.ch/TC57/2013/CIM-schema-cim16#.

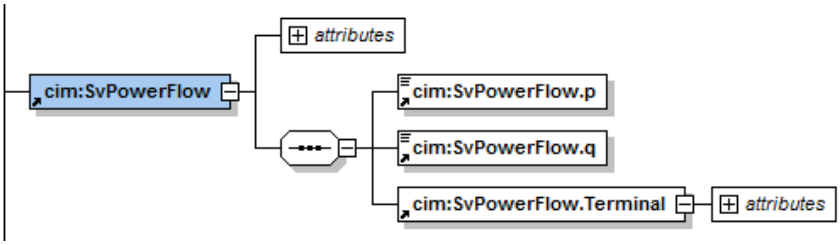


*Figure 41 Graph structure of the attributes for the CIM class SvPowerFlow.*


*Figure 42 Graph structure of the attributes of the CIM class SynchronousMachine.*

All the standard CIM names are identify by the prefix **cim:**, e.g., the rated apparent power of a synchronous machine is identified by the attribute cim:SynchronousMachine.ratedS.

### 5.2.2. Modelica Code Representation

A model implemented in the Modelica language is comprised of three parts:

- First part, a specialization keyword, indicating which kind of class should be represented, and the name of the class open the model declaration. From the Modelica definition, in the mapping we consider three different class stereotypes: ***Model, Class*** and ***Connector***.
- Second part, there is the variable section where all the parameters of the model are defined, with their corresponding attributes that define the parameter's characteristics: ***visibility, variability, name, value, comments*** and ***annotations***. The *variability* indicates the kind of variable; and *visibility* indicates the level of protection of the variable.
- Third part of the model is the equation section where the mathematical model is represented.

The following lines of code represent the implementation of a Pi-equivalent model for the OpenIPSL Line model, with its attributes and equations:

```
1.  class Line "Pi-Line"
2.     PowerSystems.Connectors.PwPin p;
3.     PowerSystems.Connectors.PwPin n;
4.     parameter Real R "Resistance p.u.";
5.     parameter Real X "Reactance p.u.";
6.     parameter Real G "Shunt half conductance p.u.";
7.     parameter Real B "Shunt half susceptance p.u.";
8.  equation
9.     R * (n.ir - G * n.vr + B * n.vi) - X * (n.ii - B * n.vr - G * n.vi) = n.vr - p.vr;
10.    R * (n.ii - B * n.vr - G * n.vi) + X * (n.ir - G * n.vr + B * n.vi) = n.vi - p.vi;
11.    R * (p.ir - G * p.vr + B * p.vi) - X * (p.ii - B * p.vr - G * p.vi) = p.vr - n.vr;
12.    R * (p.ii - B * p.vr - G * p.vi) + X * (p.ir - G * p.vr + B * p.vi) = p.vi - n.vi;
13. end Line;
```

## 5.3. Mapping rules and Mapping Meta-Model

The mapping rules are defined within a **Mapping Class Structure** which contains the attributes from both CIM and Modelica semantic syntaxes. The workflow for the creation of this meta-model class structure based on mapping rules is shown in Figure 43. The mapping matches the CIM classes and attributes names with the component names and their parameter names from the OpenIPSL library component models. The application of the workflow from Figure 43 can be updated and modified for creating the mapping rules for different modeling languages.

In the first step identified by the action ***Identify Common Parameters***, a model developer defines the matching criteria of CIM classes and OpenIPSL components. The design of the mapping considers different features from the Modelica language, and is considered within the second step of the workflow, ***Create Mapping Rules***: A ***ComponentMap*** element have attributes *rdf_id* and *rdf_resource*, which are used by the M2M transformation to facilitate data search. The attributes ***cim_name*** and ***name*** of the rules, maps the CIM name of a device, the OpenIPSL package, for the correct declaration of components within the target network model, and the attribute ***stereotype*** maps the kind of Modelica object.

The *ComponentMap* (in Figure 44 the name *pwPinMap*) element contains an ***AttributeMap*** element, which maps the CIM class' attributes as OpenIPSL component parameters. Thus, the attributes from CIM classes are mapped with ***parameter*** variability when doing the translation into Modelica.

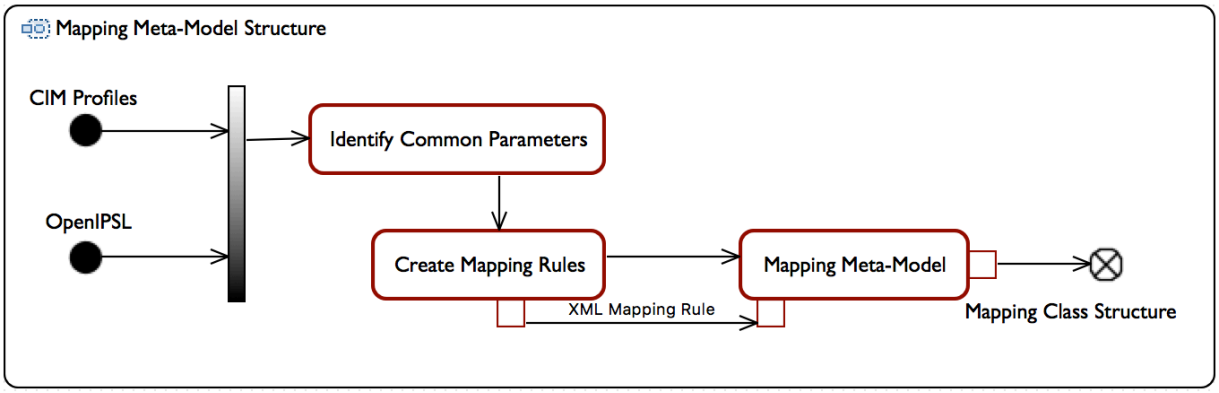


*Figure 43 Workflow for creating the mapping rules and the class structure implementing those rules*

```
1.  <?xml version= "1.0" encoding= "UTF8"?>
2.  <!DOCTYPE model SYSTEM "openipsl_component_map.dtd">
3.  <pwPinMap cim_name= "class_name" rfd_id="" rfd_resource=""
    package="openipsl_package_path" name="openipsl_component_name"
    stereotype="connector">
4.  <attributeMap cim_name= "attribute_cim_name" name="openipsl_parameter_name"
    datatype="Real" variability="parameter" visibility="public" flow="0"> value
    </attributeMap>
5.  <attributeMap cim_name= "attribute_cim_name" name="openipsl_parameter_name"
    datatype= "Real" variability="none" visibility="public" flow="1"> value
    </attributeMap>
6.  </pwPinMap>
```

*Figure 44 Example of mapping rules implementation, in XML. It covers the mapping of a CIM Terminal and an OpenIPSL PwPin component.*

Furthermore, The *AttributeMap* element is composed by a *cim_name* and *name*, a *datatype* for the data type of the CIM attribute, and *variability* and *visibility* of the parameter. A sample of the XML code with the mapping rules between CIM and the OpenIPSL Modelica Library. The ruels are implemented in XML format to facilitate its design simplicity.

The third step within the workflow is the action ***Mapping Meta-Model***. A specialized JAVA parser code for XML [86] is used to read the XML file, of the mapping rule, and its associated XML schema and creates a *JAVAX* class for the corresponding OpenIPSL component (see in Figure 45, the class ***GENROUmap***). The resulting *Mapping Class Structure* is composed by a set of classes which contain the matching criteria between the CIM components models and their equivalent Modelica models (Detailed information on the creation of this class structure can be found in Appendix A).

The ***Mapping Class Structure*** from Figure 43 is used to populate the values from the source CIM model into memory. The classes *ComponentMap* and *AttributeMap* contain the key attributes to identify the CIM classes and CIM attributes to be read from the CIM model, and their correspondent key attributes in the Modelica component. This class structure is manually enhanced with an additional class, ***ConnectionMap*** class, which contains the references of the CIM classes Terminal, *ConductingEquipment* and *TopologicalNode*. This *ConnectionClass* is be used to populate the references of these classes and to create the connections between components. The application of the specialized code to transform the mapping rules into the class structure creates an additional *ObjectFactory* class, which has been manually modified to create specialized Factory classes for each type of components mapping classes (i.e. class *SynchMachMapFactory* from Figure 45).

## 5.4. Meta-Model for building model components

The second stage of the M2M transformation process is to create the target model, using the syntax rules of modeling language to be used, in this case, the Modelica syntax. The main features of the power systems model are the model topology, the components and their connections. Thus, the Modelica language provides different object stereotypes that will be used to automatically create the target model.

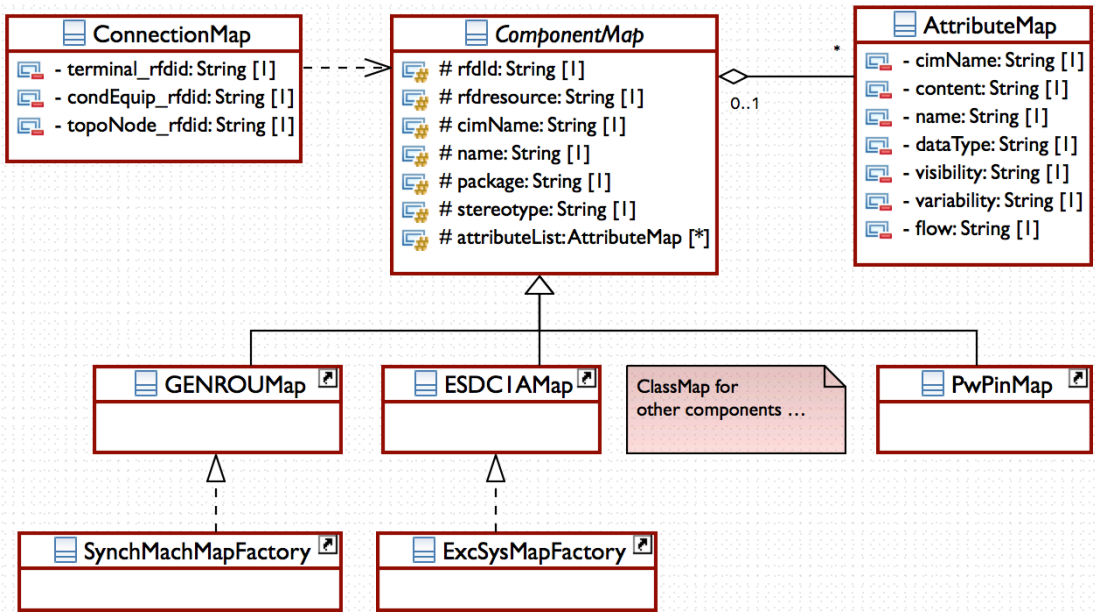


*Figure 45 UML class diagram, shows the design of a meta-model structure of the mapping, containing either both CIM classes attributes and Modelica language keywords and OpenIPSL names.*

A second workflow shows the process of identification of the target language syntaxes to be used to create the instances of the power system components models. The workflow for this second stage is in Figure 47, which considers the Modelica class stereotypes ***model***, ***class*** and ***connector*** (see Figure 46), to differentiate which component should be created and define its place into the model hierarchy for a network model. Thus, the resulting ***Network Class Structure*** resulting from this workflow is created by a set of actions that gather the identification of the relevant Modelica keywords and creation of different classes for automated model creation.

The *Network Class Structure*, considers the model stereotype to define the high-level model, e.g. the network grid; the class stereotype for medium-level models, e.g. a component, and the connector stereotype for low-level models, e.g. a connector. Its implementation is enhanced with additional classes to store the remaining features of a Modelica model.

The *Network Class Structure* is defined as general as possible (see Figure 48), covering the main language features for building models. The attributes defined in each class gives information about the most important keywords from the Modelica language. However, their semantic meaning can be interpreted and use for other equation-based modelling language, i.e., the *moAttribute* class' attributes can be used for other language parameter definition.

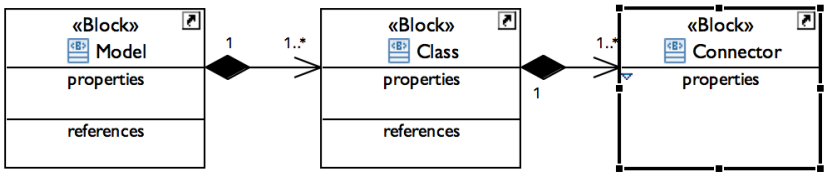


*Figure 46 This block diagram represents the three Modelica class stereotypes defined as blocks.*

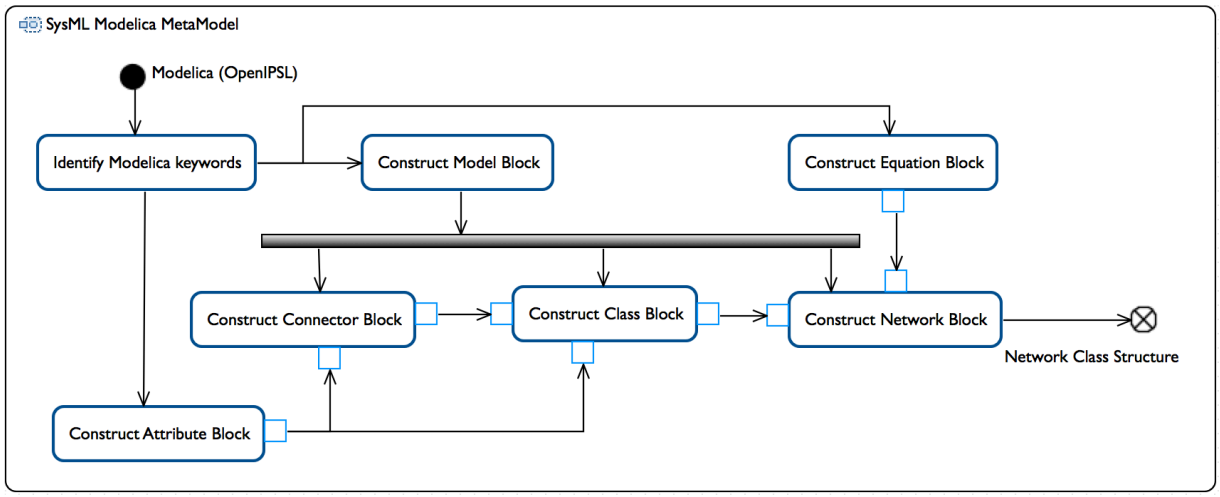


*Figure 47 Workflow for creating a class structure based on Modelica syntax and SysML semantics.*

1. Classes ***moModel***, ***moClass*** and ***moConnector*** correspond the Modelica stereotypes model, class, and connector. The *moModel* implements the model stereotype and is used to define the high-level models. The *moClass* implements the class stereotype, to define the medium-level models. The *moConnector* implements the connector stereotype, to define the low-level models.

2. The meta-model structure includes two additional classes for defining high-level models within a network model: The class ***moPlant*** encapsulates the components that are included in the model of a plant, e.g. machines and regulator components. And the ***moNetwork*** plant encapsulates plant objects and component objects and defines the whole network structure

3. ***moClass*** and ***moConnector*** are used with values from the mapping to create the instances of the OpenIPSL objects. The class ***moAttribute*** stores the values of the parameters of each component

4. The ***moEquationConnect*** is an additional class to the structure, that is used to build the Modelica connect equations. These equations define the final topology for the high-level network.

Because the words model, class and connector might be reserved keywords in different programming languages, the prefix mo has been added at the beginin of the name of the classes.

## 5.5. Main workflow implementation

The complete actions within the main workflow of the M2M transformation tool are depicted in Figure 49. It starts with an input CIM model that contains classes and attributes with Equipment (EQ), Topology (TP), State Variable (SV) and Dynamic (DY) Profile information of the network. The next action of the workflow, ***Identify CIM Profiles***, identifies the CIM classes and attributes, and parses the CIM/RDF implementation into memory. Next, for each CIM class of interest, the correspondent mapping rules are loaded and are used by the ***Populate Mapping Objects*** action to populates values from the CIM model into the *Mapping Meta-Model* structure.

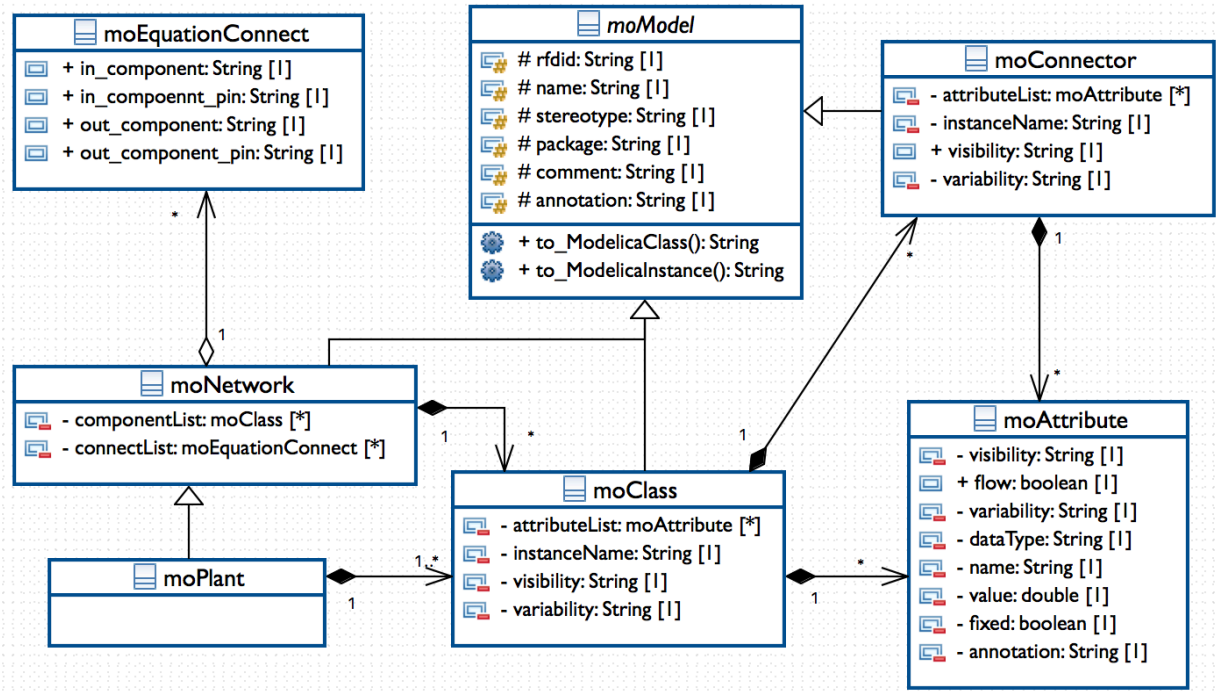


*Figure 48 UML class diagram, shows the design of the proposed internal structure for creating the Modelica code. It defines Modelica names and keywords that will be used for generating the final network model.*

Last step is to use the values from the *Mapping-Meta-Model* objects to create the instances of the corresponding OpenIPSL components. The action ***Instantiate Network Objects*** creates the network structure adding the component objects and connections to the model. The main classes and methods for the implementation of the transformation algorithm are depicted in Figure 50. Based on CIM profiles, a modular implementation is used, defining a class to process each of the available profiles. This modular implementation allows to extend the tool considering other profiles, such as the CIM *DiagramLayout* profile, for CAD application.

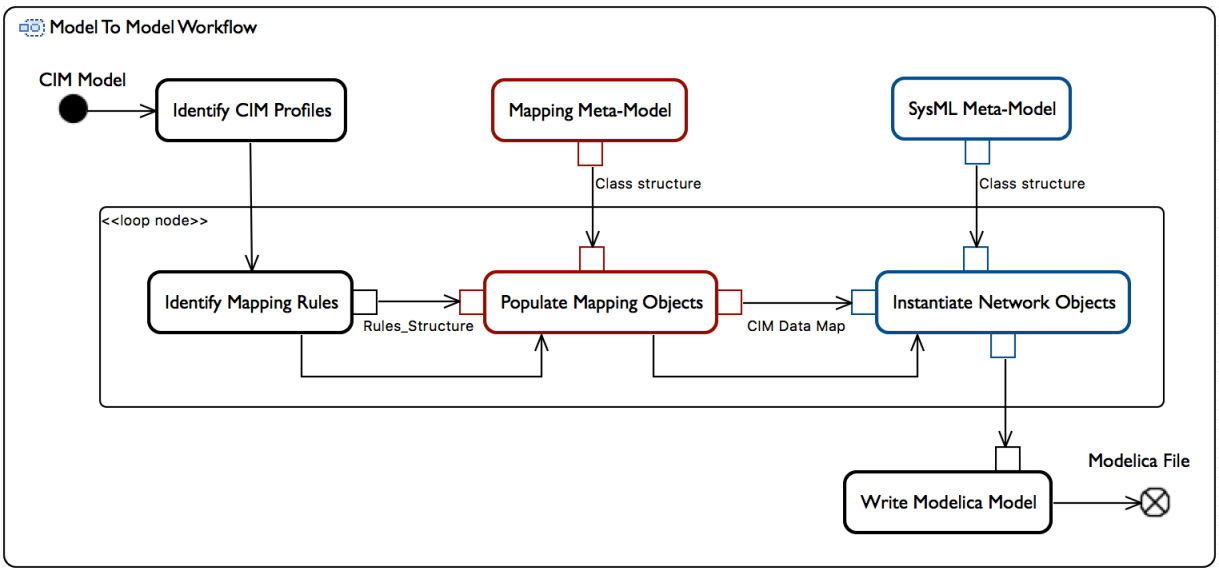


*Figure 49 Main workflow with the main functional modules for this use case*

To design the methods to control the workflow from Figure 49, the specialized classes ***ModelDesigner*** and ***ModelBuilder*** are implemented. The *ModelDesigner* class is responsible to implement the Populate Mapping Objects and the *ModelBuilder* class is responsible to implement the *Instantiate Network Objects*. With this separation of responsibilities, the *ModelDesigner* class with the *Mapping Meta-Model* structure can be used separately from the *ModelBuilder* class with the *Network Meta-Model* structure (see Figure 50)

1. The *ModelDesigner* class acts as a controller object to *unmarshall* the mapping rules and populate the values from the source model into the Mapping Meta-Model structure. It aggregates different control classes that implement an API for reading CIM values from the available CIM profile files. The *ModelDesigner* class uses factory objects, such as the *SynchMachMapFactory* to instantiate the adequate JAVAX class according to the mapping rule loaded.

2. The *ModelBuilder* class acts as a controller object to create the instances of the *SysML Meta-Model* structure. It uses the mapping classes containing the source model values as input parameters, to create the component instances and connections forming the target network model.

3. The *CIM2MOD* class acts as the main controller object. This class implements the main logic of the transformation tool. It uses the API for reading the EQ CIM profile, to navigate through the Terminals and related *ConductingEquipment* and *TopologicalNode* RDF resources and load their values, located within the other profiles.

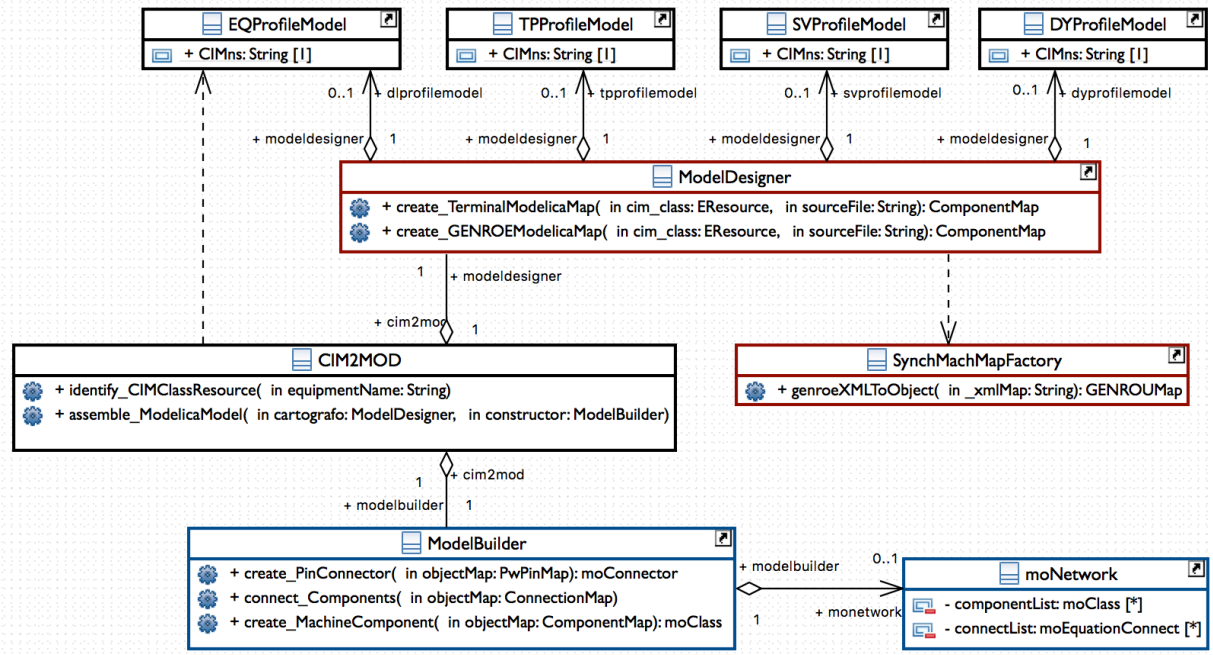


*Figure 50 Class structure representation of the main controller classes, which implement the main logic of the M2M transformation tool. The figure only represents a subset of methods for each class.*

### 5.5.1. Structure of an input CIM model

The current version of the proposed M2M transformation algorithm supports an input CIM bus-branch model containing the CIM classes depicted in Figure 51. Those are the main CIM classes with information from the EQ, TP, SV, DY profiles. The values from the TP and SV profiles are used to assign these values as *"initial guess"* values of *active power, reactive power, voltage,* and *angle* for the OpenIPSL component's variables that need to be solved for during initialization by a Modelica tool.

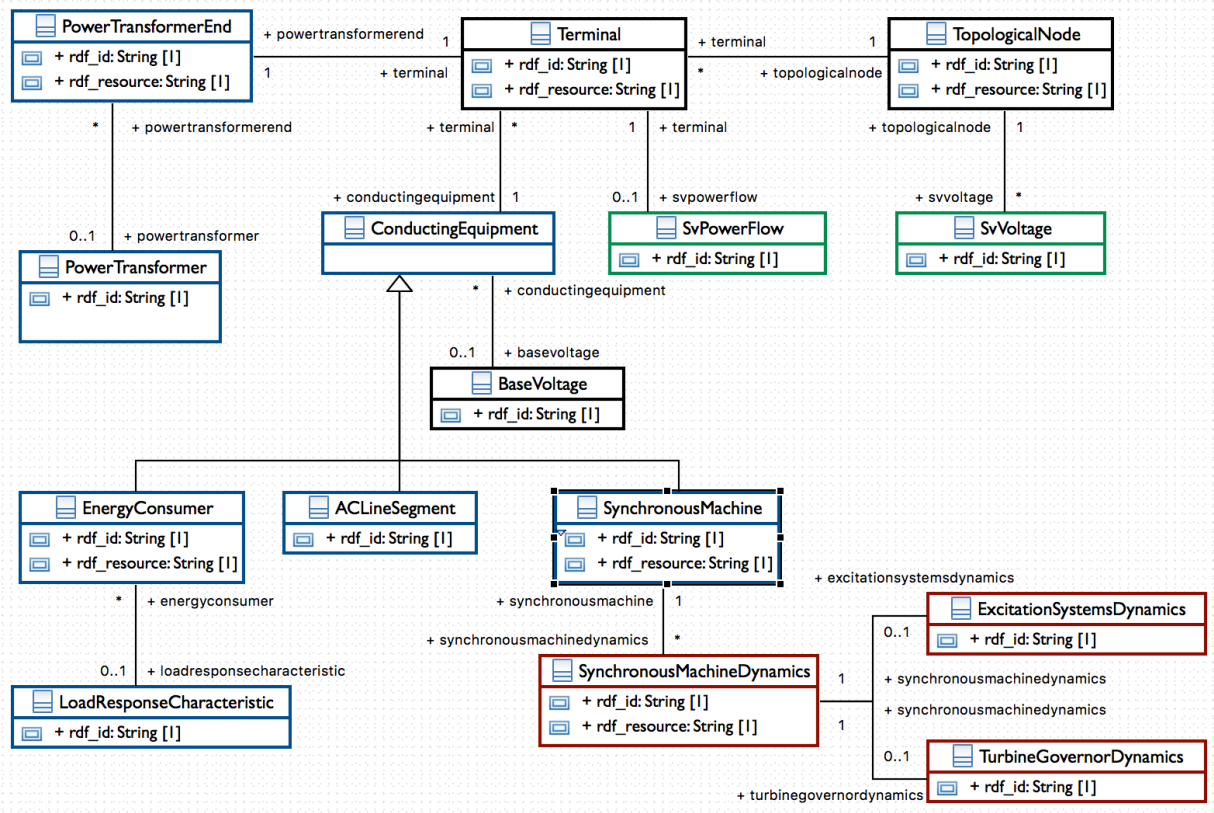


*Figure 51 CIM objects representing the main information to be processed by the model transformation tool.*

### 5.5.2. Simulation results

The definition of classes described in the previous sections have been used to implement a algorithm for the M2M transformation, which is described in the Appendix B. To test the proposed M2M transformation process, the IEEE 9 Bus model is used [87]. The system is composed by 3 generators operating at 275 MVA and 16.5 kV in Generator 1; 320 MVA and 18 kV at Generator 2; 300 MVA and 13 kV at Generator 3, with a system base power of 100 MVA. The M2M transformation results of the test model are shown in Figure 52 for simulation purposes; a fault has been added to the model to show better the dynamic behavior of the model. The fault is applied at ***Bus 8*** in the systems (see Figure 53 and Figure 54).

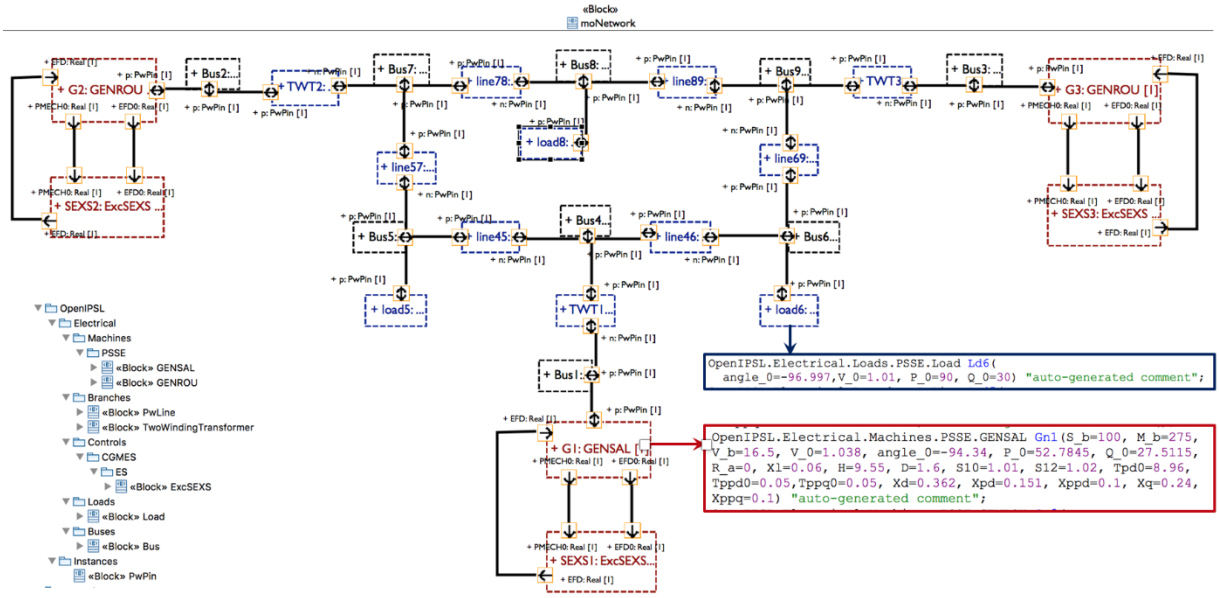


*Figure 52 SysML diagram of the 9-Bus system. In the bottom left corner there is the hierarchy of OpenIPSL*

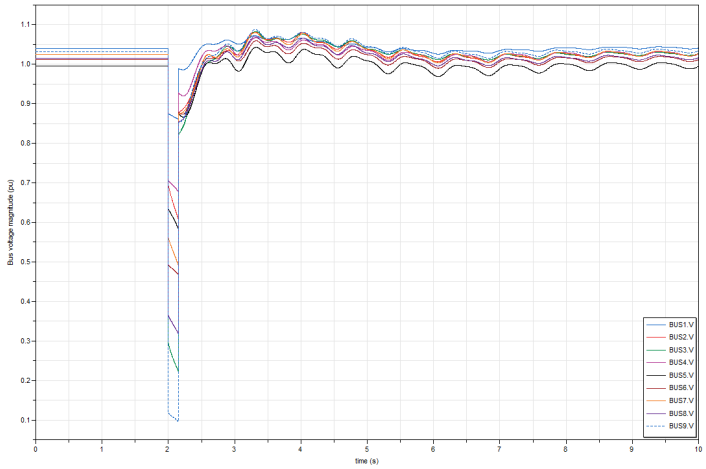


*Figure 53 Simulation results of the simulation generated model, IEEE 9 Bus model, top: Voltages (p.u.) at each bus*

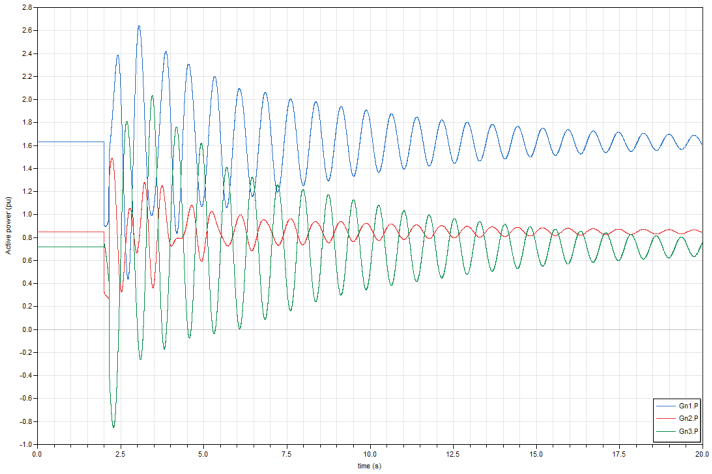


*Figure 54 Simulation results of the simulation generated model, IEEE 9 Bus model, top: Active Power (p.u.) at each Generator*

Using the Modelica compiler within the OpenModelica editor, the initialization calculation can be deactivated, forcing the solver to use the initial conditions from the power flow computation. This simulation configuration ensures the use a power flow solution before any time-domain simulation. Using the Modelica compiler in the Dymola editor, there is no direct method for deactivating the initialization phase in the solver. But the use of guess values for all variables but current values improved the computation speed of the initialization phase.

## 5.6. Summary

The complete process of binding the CIM syntax and Modelica syntax presented in this paper gives a mechanism to use mathematical modeling language as a complement for the CIM Dynamics information exchange profile, and its CGMES extension. With this solution, dynamic analysis such as time-domain simulations, can be performed using a model defined in Modelica and simulated by utilizing Modelica compilers that have proved to be applicable for this kind of analysis in different fields [88] and are showing promise for continental-level power systems simulations [89].

The mapping rules and mapping structures presented in this work are implemented following OOP principles to ensure scalability and maintenance to support new equipment mapping rules. The reason to split the mapping into two different structures follows a modular implementation of the proposed methodology. The *Mapping Class Structure* of Section 5.3 can be reused to convert the CIM into another modeling language, while the *Network Class Structure* in Section 5.4 can be reused to generate Modelica code from a different modeling language, such as SysML. Moreover, this structure itself can be used to build the network model for other simulation tools. This implementation guarantees the scalability, reusability and modularity of the proposed methodology.

The use of Modelica for the modeling of power systems dynamic behavior allows to comply with the application of EU rules for security, interoperability and transparency established by European regulations [34]. A rich model description provided by the IEC Common Information Model, is complemented by the mathematical descriptions in Modelica, providing the simulations needs for dynamic analysis. To show the dynamic behavior of the network with the presence of a fault i.e. to comply with requirement *Req. 4.5* (Chapter 3 section 3.3.4) for building a completed dynamic network model with discrete events, fault models have been implemented manually into the output Modelica models. To automatically add events, such as line trips, other CIM profiles [90] need to be supported in the proposed workflow.

The results of this workflow show how to build a power systems network model with the use of CIM information and OpenIPSL power system components. This chapter gives a proof of concept of how to comply with Appendix F in the new CGMES standard guidelines [91] to use Modelica models in a power network description and serves as an example of the feasibility of the approach recommended by this Appendix F.

# Chapter 6

# Engine-Based Software Architecture

## 6.1. Challenges in Software Tools for Modeling and Simulation of Cyber-Physical Systems

The concept of a cyber-physical system makes possible the adoption of a wide range of traditional power system planning and operational scenarios, which require the application of open standards for information exchange, modelling, and standard software technologies for co-simulation. New cyber-physical systems incorporate discrete-event formalisms of computational process and continuous-time process of the physical world [92]. This makes possible the integration of software engineering disciplines with electrical engineering and power systems engineering at the semantic level. However, the complexity of modeling and simulation tools is increased by the integration of different semantics, making it necessary to use General Purpose Modeling Language (GPML) to create some sort of theory for system integration to provide an understanding of the design challenges faced by the interaction of different engineering disciplines [93].

The currently available technologies for modeling, simulation, and analysis, most of which have been explained in Chapter 2, lead to the development of a horizontal integration framework, design composer methods, library of components, and integration mechanisms to compose and simulate new physical models with semantic languages and information from different engineering domains. Hence, the proposed architecture herein is elaborated by following the concept of cyber-physical power systems, which conceives the whole electrical grid as a cyber-physical system that has come into being as a multi-disciplinary and multi-domain system [94].

In many proprietary and open-source power systems, simulation software integrates the solution algorithms strongly connected to internal (and discretized) models. This approach has several back draws, such as limitations and extensibility and portability of the model for execution in other simulation engines. Moreover, the computational representation of these models is tightly integrated with the mathematical solvers of those simulation engines. Thus, it is not possible to identify the physical representation of models from the computational core of software tools [95], making further integration of other engineering disciplines difficult.

The adoption of the equation-based object-oriented Modelica language offers a strict separation between the model and the solver, as well as model portability thanks to its standardized language definition [37]. Moreover, the usability of the SysML for the design of system integration offers a solution for a better understanding of the interaction between computational systems and the physical systems. This chapter brings a brief description of a prototype software architecture based on computational engines, as an integrated framework where different components, tools, methods, and data formats can collaborate to meet most of the requirements gathered in Chapter III and leveraging the results from the other chapters in monograph. The proposed design of the software architecture tries to gather a variety of functionalities and concepts that can be implemented with state-of-the-art modeling and computational programming languages.

## 6.2. Software Architecture Based on Engines: Proof of Concept Design

The design of software architecture based on engines is presented herein and provides a continuation to the work from [20]. The design is conceived to cope with the technical specifications, such as the integration of the CIM, Modelica and FMI standards, formalized as functional and non-functional requirements presented in Chapter III. The architecture design offers modularity and scalability, so as to develop a set of modules or engines that can interact with each other with the exchange of a power system model and simulation data. Design patterns from Software Engineering domain, such as *Abstract Factory*, are applied in a way that those engines could be developed to be used independently from each other:

1) The ***Data Model Engine (DME)*** is used to help with the design and implementation of new information and mathematical models. The description of the main functionalities of this engine is covered in both Chapter 4 and Chapter 5. Thus, no further explanation will be added within this chapter.

2) The objective of the ***Steady-State Solver Engine (SSE)*** is to compute the steady-state solution of the power system model and update its initial conditions. It works with the information from the CIM profiles.

3) The aim of the ***Model Execution Engine (MEE)*** is to provide a set of scripts for executing dynamic analysis methods, e.g., time-domain simulations, with different simulation compilers.

4) The ***Measurement Analysis Engine (MAE)*** gathers the different functions to analyze real measurements and compare them with the results from the simulations.

5) The ***Intelligence Engine (IntE)*** is conceived to provide analysis methods wherein a model's variables should be calibrated to improve the simulation results to better match with the real measurements available.

6) A database management system provides the available methods to store those data of the particular interest for the whole framework. The ***Data Manager Engine (DBE)*** provides methods to handle these data.

7) The purpose of the ***Process Manager Engine (PME)*** is to provide easy-to-use commands as an Application Programming Interface (API), so they can be integrated in different human-machine interfaces (HMIs)

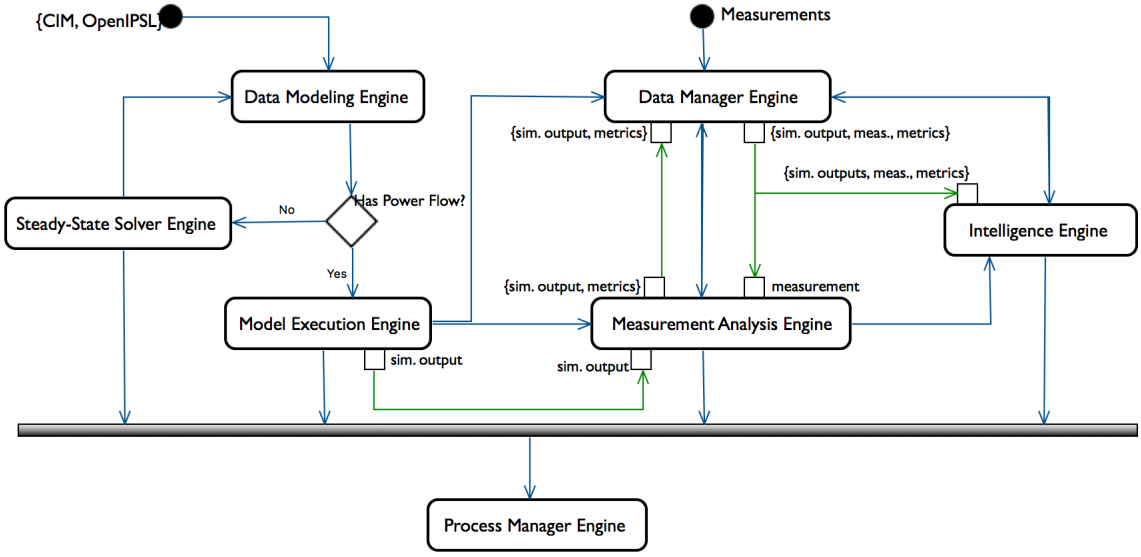


*Figure 55. Main workflow and general view of the software architecture.*

The development of this architecture was first proposed as a prototype for the integration of the standards studied. A list of few specifications was analyzed in [20], included within the first cycle of the development process described in Chapter 2 section 2.4.1. The continuous work on the different engines in subsequent cycles of development process help to rethink and enhance the technical specifications for the architecture. After the work on the formalization of requirements, the original structure of the architecture has been modified and enhanced. This has led to the analysis and design of a new structure of the architecture to integrate further work on a multi-domain meta model (described in Chapter 4) and on the implementation of mappings for Model-2-Model transformations (explained in Chapter 5).

The SysML Activity diagram shown in Figure 55 is used to depict an enhanced design of this prototype from the one presented in [20]. This diagram shows the proposed workflow design, with control arrows between engines (in blue color) and object flow arrows (in green color). The control arrows are used to visualize typical workflows for power flow analysis, from the modeling stage until analysis and reporting. In addition, object arrows visualize the exchange of data between engines. This data exchange can contain outputs from simulations, real measurements obtained from measurement devices, or analysis metrics, which are the result of the possible different analysis that could be carried out with the model object with the data objects. The same type of SysML Activity diagrams and UML Class diagrams are used in the following subsections to represent the proposed workflow and main structure of the framework.

### 6.2.1. Model Execution Engine

The Model Execution Engine (MEE) is conceived as implementing the requirements of running time-domain simulations with different simulation compilers. Time-Domain simulations can be performed using non-proprietary compilers: the OpenModelica Compiler (OMC) [38] and JModelica [40]. However, the design of the engine allows the use of other compilers as plug-ins for the engine. Thus, other commercial compilers such as Dymola [96] could be also integrated in the architecture. Thus, Req. 3.1 and Req. 3.4 from section 3.3, could be fulfilled. Those requirements state that a software tool could be able to simulate a network grid composed by the same modeling language (in this case, Modelica) or by different component parts (which are implemented within FMUs).

Figure 56 depicts an internal workflow of the engine. It should get as input data the model to be simulated and the solver configuration to be used (its configuration data could be stored in an internal file from previous executions). Before executing the analysis, the engine should allow one to check if any event is connected to the model, e.g., in terms of a fault or a load change variation, as stated in *Req. 4.5* from Section 3.3.4. The next step is to set up the configuration of the integrator method from a list of solvers available in the Modelica compilers within the *Setup Method* action. In case the FMU imported has the implementation of the Co-Simulation interface, its solver configuration is also done by the same action.

The dynamic analysis mainly involves the execution of time-domain simulation, which follows *Req. 3.4*. However, the decision node in Figure 2 represents the choice in solving the behavior of the model or performing an analysis of the equations of the model. This makes the implementation of the engine scalable to other type of dynamic analyses, e.g., for an ambient analysis of a power system [107].

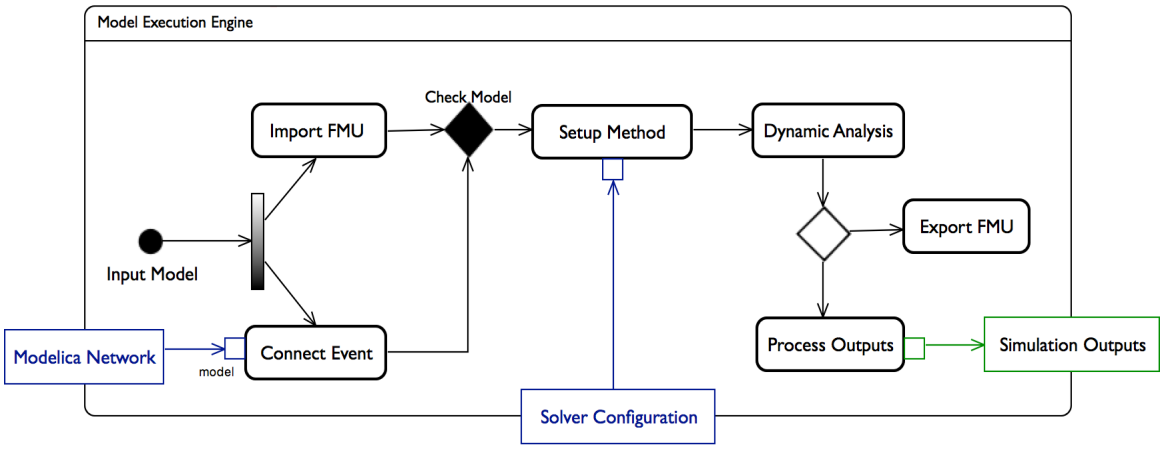


*Figure 56. Workflow design for the model execution engine.*

From the specification Req. 3.7 from section 3.3.4 about standard data formats, an internal process within the engine stores the outcomes of the simulation within the HDF5 format that constitutes the main data format of the whole framework. In this sense, new software tools based on this architecture could take benefit of the implementation of the standard HDF5 APIs [97] to access and handle these output data.

### 6.2.2. Steady-State Execution Engine

The Steady-State Execution Engine (SSEE) is created from the idea of computing the corresponding initial conditions of dynamic simulation models, and it tries to satisfy the requirements *Req 3.2, Req 3.3, Req. 4.2* and *Req 4.3* about the execution of steady-state analysis, from section 3.3.3. Its objective is to provide the proper initial conditions for the initialization of Modelica-based network models, as suggested by the work in [47]. Thus, the engine is designed to gather different methods for computing power flow solutions, on either CIM information models or Modelica models. Figure 57 shows a design of the main workflow for this engine.

On the one hand, the engine could accept as input data CIM-based information models in the form of Node-Breaker models {TP}, within Equipment and Steady-State Hypothesis profiles {EQ, SSH}. In this case, topology processing techniques, such as in [98], could be implemented for Topology processing and power flow computation. The result is an extended information network model, following the Bus-Branch topology, represented by the CIM profiles of Equipment, Topology and State-Variable {EQ, TP, SV}. Those outcomes are used within the DME and apply the M2M transformation technique to obtain a complete OpenIPSL network model.

On the other hand, the engine could accept Modelica models as input data. In this case, Modelica-based methods could be applied to check that all the model's parameters and variables have their proper values to execute dynamic simulations. This proposal of working with Modelica-based network models leave the door open to the development of mathematical methods for power flow computations, based on the Modelica language.

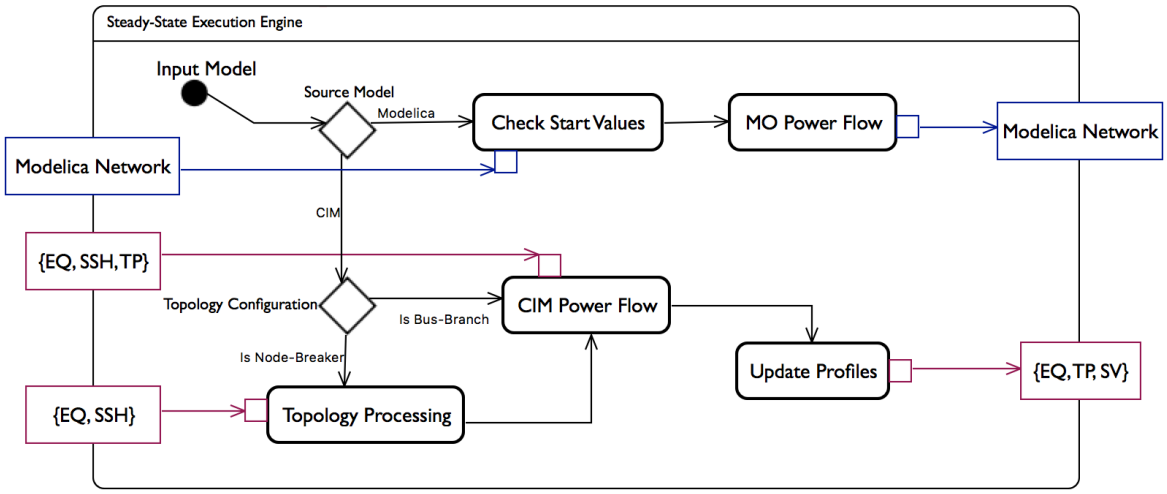


*Figure 57. Workflow design for the steady-state execution engine.*

### 6.2.3. Measurement Analysis Engine

The design of the engine proposes a simple workflow of picking signals, executing an analysis method on those signals and storing the signals, the metrics, and methods chosen for the analysis (see Figure 58). Thus, a class structure such as that represented in Figure 59 – based on the pattern design Abstract Factory [99] – could be used to create instances of these methods and perform the analysis automatically. The actions *Ambient Analysis* and *Ringdown Analysis* are used as examples to depict the action of choosing an available implemented method by its name.

As an input, the engine works with the outcomes produced by the MEE and the measurements (if available) of the network being simulated. The class structure in Figure 59 gives an idea of the proposed implementation of analysis methods. The methods should be adapted to analyze simulation results and compare them with real measurements. Different statistics are calculated and stored in a Measurement DB, where the simulation outputs, measurements, and the metrics that the engine has calculated are stored.

The proposed design of the engine workflow allows for the execution of signal analysis based on ambient data – signals with presence of noise, signals with little oscillation and without any sudden change of set point, etc. This also allows to perform ringdown analysis, which is based on small portions of the model's response. Those segments show the transient and sub-transient behavior of models, due to sudden variations in electrical voltage, current or frequency that happen in a very short period.

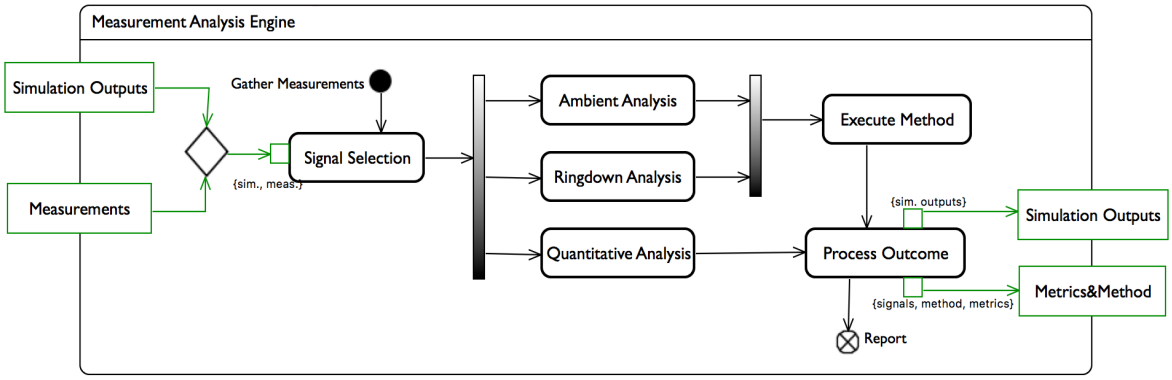


*Figure 58. Workflow design for the measurement analysis engine.*

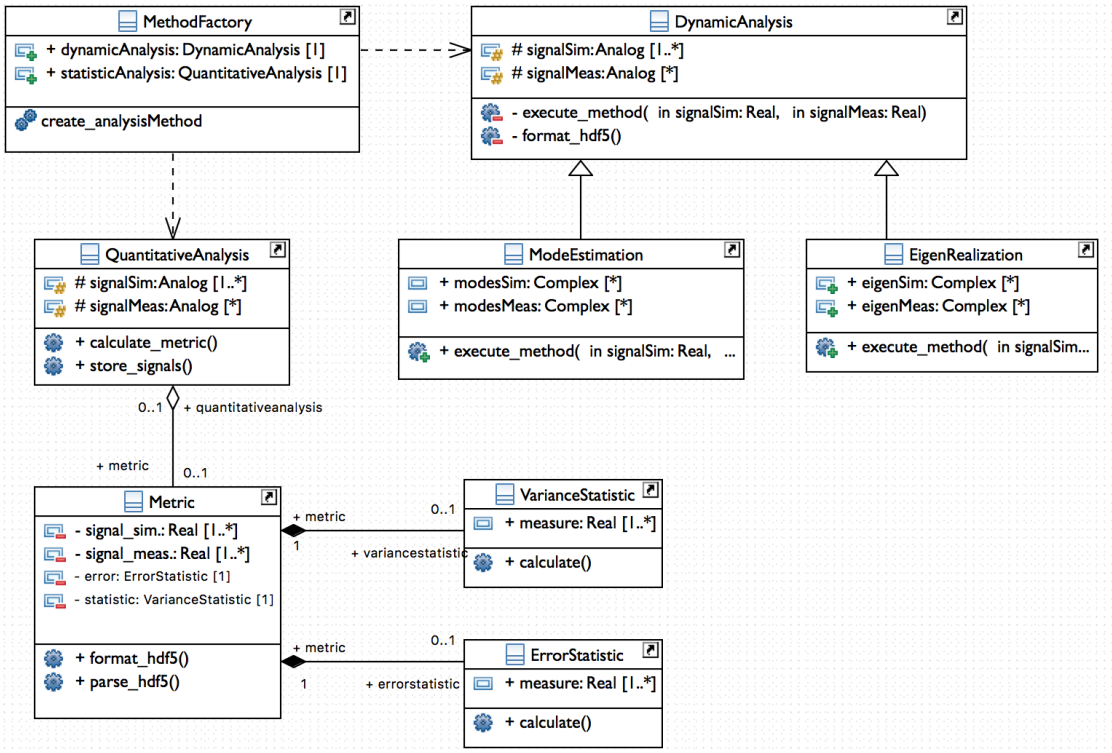


*Figure 59. Proposal of class structure for a Measurement Analysis Engine Design, based on Factory Method.*

The separation of signal analysis in ambient and ringdown provides an advantage in separating which model parameters should be identified for further calibration (as further explained in the next section, depicted in Figure 67). Quantitative analysis refers to the computation of statistics that can help to measure how exact are the simulation signals from their equivalent measurements.

### 6.2.4. Model Calibration Engine

This engine has an important role in the whole power system analysis workflow, because it provides the mechanisms for automatically adjusting the model parameters to match the equivalent real measurements. This adjustment can be provided by statistical analysis of the mean, std. deviation, and variance for the parameters being identified – depicted by the *Quantitative Analysis* action in Figure 60. The Optimal Calibration decides whether to continue with the calibration process by the computation of statistics of error measurement, such as the Root Mean Square Error (RMSE) [100].

The engine can work manually, with simulation signals and measurements, directly from a recent simulation. For this purpose, it should accept only data from simulation signals and equivalent measurement signals. It can also work with the metrics computed from the MAE, so it can automatically execute the calibration process just on the signals of interest previously analyzed. In this case, it can accept the *Metrics* computed in the *Measurement Analysis*. To operate those cases, the configuration method should be provided to the engine from the available calibration methods implemented. These actions (depicted in Figure 60) complement the definition of *Req 3.8* from section 3.3.4.

This engine should produce three different outcomes. First, there is the result of the Quantitative Analysis performed by the calibration process and the method used for calibration. This result could be useful to repeat, reproducing this process in different simulation environments, such as the RaPId environment [72]. Second, the FMU model is

used with the new parameter values. Thus, it can be directly exchanged with other simulation tools that can also adopt the FMI standard. Third, the final parameter values should be stored in a CIM-based database or model. This new information model could be used for further M2M transformation processes to produce a better equivalent Modelica model.

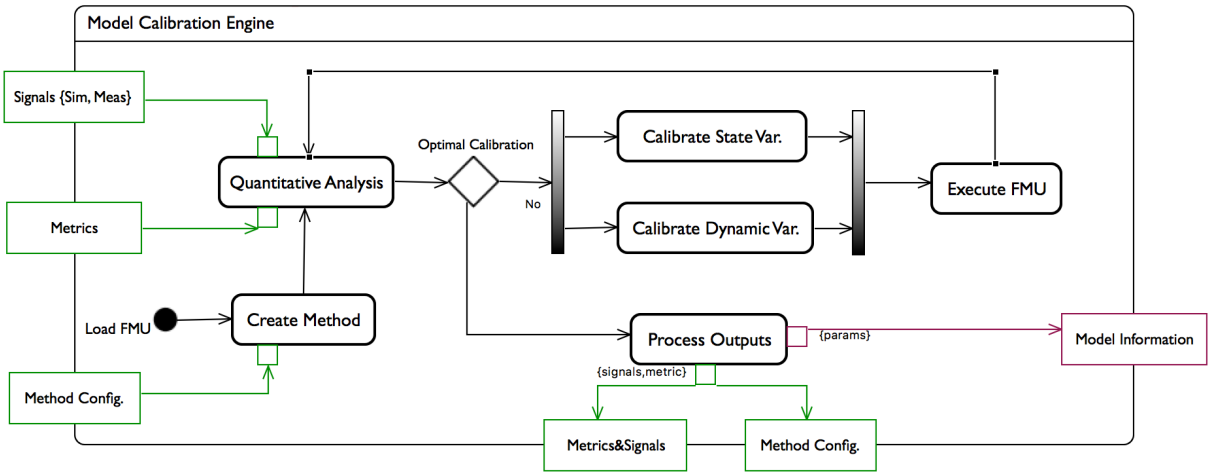


*Figure 60. Workflow design for the intelligence engine, concerning calibration of model.*

### 6.2.5. Data Manager Engine

Conventional research on software architecture proposals for power system tools, modeling, and simulation in general do not put much emphasis on the internal data format. Work such as [101] or [102] are good examples of good architecture designs and functionalities without mentioning how to store the data. In the power system domain, this decision is up to the simulation tool vendor, which implements its own internal data base structure with its own data format [61][62]. Simulation tools often implement mechanisms to export their internal data structure to other standard data formats.

The design of this engine is proposed from the perspective that it can be used in isolation from the framework proposed, and the proposed data structure could be adapted to work with other tools. Following the definition of *Req 3.7* and *Req. 3.7.1* from section 3.3.5, the use of the HDF5 format is proposed. A set of classes has been developed to manage all the data needed (see Figure 61 and Figure 62). HDF5 has been proven to be a good candidate to store time series from dynamic simulation results [102]. Within this engine, this data format is also exploited to store small amounts of information and meta-information provided by the execution of signal analysis methods.

The HDF5 format provides an API that groups data into datasets. This offers an Object-Oriented view of the information exchanged by the engines, which is easy to read, interpret, and implement by high-level programing languages. Thus, it is not necessary to depend on the implementation or adoption of a specialized data management system. This implementation is derived from *Req 2.2.3* from section 3.2.4, offering leverage for decoupling the functionality of the engines and building scalable and modular software. The engine is designed to implement a common data format for either parsing results from the Modelica compilers or storing the outcomes of the different analysis methods implemented. HDF5 offers a common HDF5 API (see the examples section). The HDF5 API will help to store data as proposed in our conceptual model in a well-organized structure within datasets and groups of data [97].

A *GeneralParser* class could be implemented as an interface for different data format parsers (see Figure 61). The data to be stored in the HDF5 files is organized following the CIM-based class hierarchy for measurements and values (see Figure 62). The classes *StreamCIMH5* and *StreamH5CIM* offer a common API to organize the simulation outputs within a standard class hierarchy. Thus, the most common outputs in power system analysis – Active Power (P), Reactive Power (Q), Voltage (V), Current (I) and angles – could be stored with additional meta-information from the components producing those outputs. Nevertheless, the design of this class structure allows the storage of other kinds of data measurements.

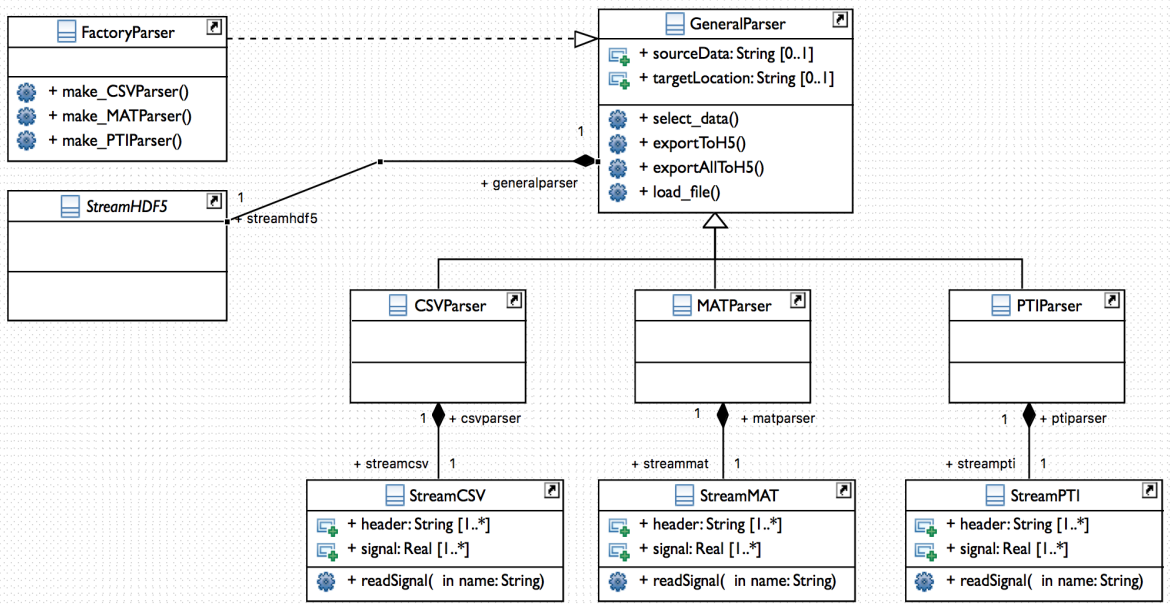


*Figure 61. Proposal of class structure, based on the Factory Method, to implement different data parsers.*

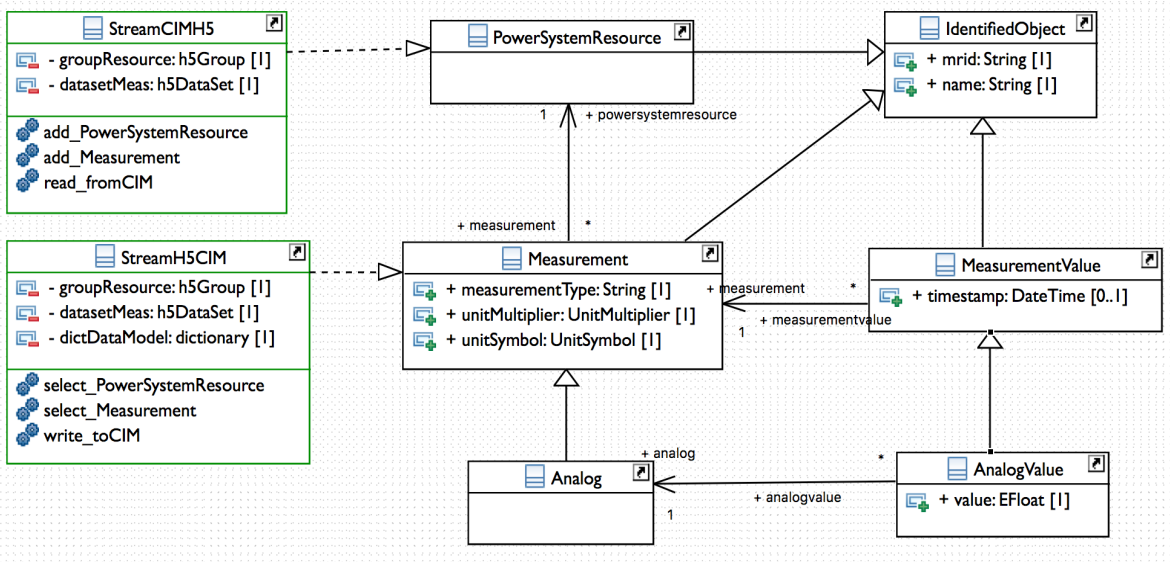


*Figure 62. Proposal of class structure to handle the conversion from CIM data format to HDF5 data format.*

### 6.2.6. Process Manager Engine

The purpose of the Process Manager (PM) is to provide an easy-to-use Human-Machine Interface (HMI) to handle all the functionalities described in the requirements, provided by the previous workflows. The HMI is designed to perform configuration tasks, execution tasks, and visualization tasks for the engines it manages. This engine is separated from the other engines and does not include any power system-related functionality. Its role is to implement generic methods to be used as external tools for the user to play with the functionalities of each engine.

Figure 63 shows the proposed design of the main functionalities of such an engine, providing an easy-to-use human-machine interface, whether it is implemented as a command line application or Graphical User Interface (GUI). The workflow shows the main types of information it manages: Information Model data, Equation-Models, Measurement data, and Analysis data. The proposed design in Figure 63, the engine itself only requires input information because it has no capability of generating its own results. Those outcomes are produced by the execution of the previous engines.

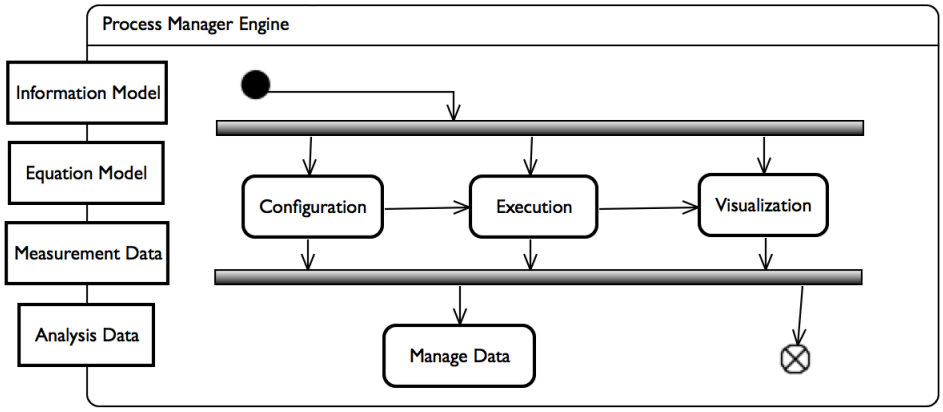


*Figure 63. Workflow design for the Process Manager Engine, with interaction with HMI to handle the functionalities provided by the architecture engines.*

## 6.3. Illustrative Examples

### 6.3.1. Model Execution Engine

A simple representation of a Single Machine Infinite Bus (SMIB) is used for testing the engine. This representation is composed by one generator connected to an infinite bus through a transmission line, and two buses will be used (see Figure 64). For this experiment, the initial conditions of the model have been set manually, directly into the model, from a power flow solution calculated from an equivalent model implemented in PSAT [104]. The resultant outputs are stored in the HDF5 format. From JModelica, the storage is straightforward using the JModelica API to obtain results. From OpenModelica ModelicaRes, API was used to parse the results into the HDF5 file.

Different Python scripts have been implemented for simulating the same model for each Modelica compilers to be tested. The scripts implement the workflow detailed in Figure 56. On the one hand, the OMC and the Dymola compliers simulate directly the SMIB model from a Modelica file (.mo). While the JModelica first translates the model into an FMU and then executes the simulation, the JModelica compiler is also capable of simulating an FMU model which has been previously translated by another tool. This test is used to prove that the results from open-source compilers are acceptable and comparable to the results from a commercial compiler such as Dymola (see Figure 65).

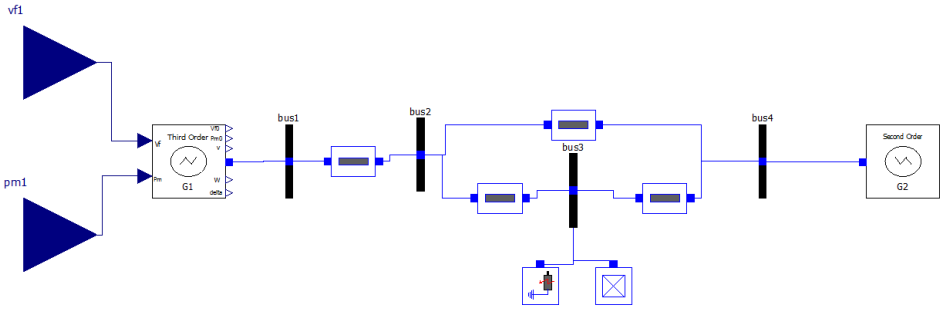


*Figure 64. Single-Machine Infinite Bus Modelica representation, with fault model at Bus 3.*

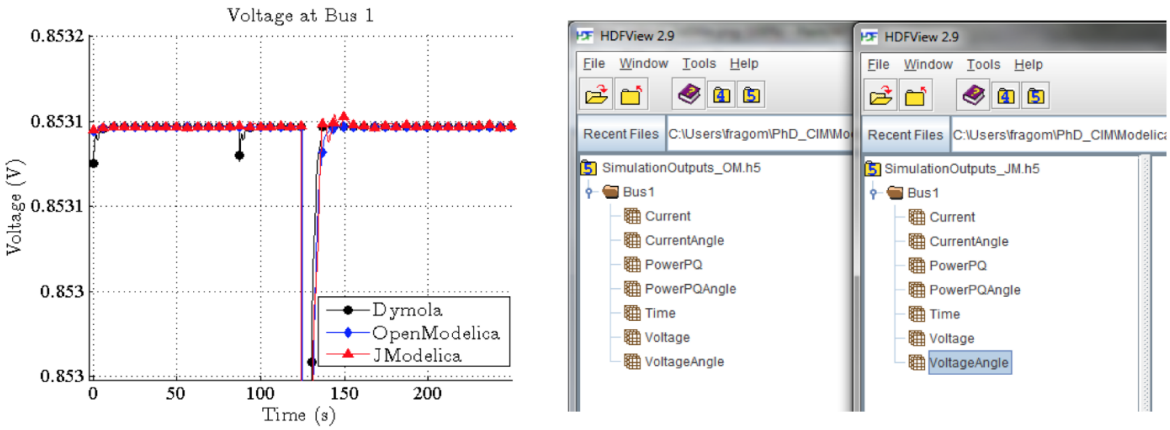


*Figure 65. (on the left) A comparison between the simulation results from JModelica, OpenModelica and Dymola. (on the right) Storage of the simulation outcomes within HDF5 format.*

Figure 65 also shows an example result of applying the DME class structure and API (Figure 7 and Figure 8) to store simulation outputs. The selection of JModelica outcomes is straightforward because its API provides functions to handle data directly. However, managing OpenModelica and Dymola outputs is more challenging, because it requires extra programming to extract values from its resulting .mat files. In this case, the ModelicaRes [105] project – an open-source tool developed in Python – provides a good API for managing the outputs produced by OpenModelica and Dymola result files.

### 6.3.2. Measurement Analysis Engine

The engine allows one to perform signal analysis on simulation signals only or with measurement signals together. For this example, the results from simulating the *Klein-Rogers-Kundur Two-Area power network* has been used [106]. In this representation, a sinusoidal white noise has been injected in the load 7 (see Figure 66). In this way, ambient noise with little oscillation could be modeled within the network to represent random load changes at an aggregated level. Therefore, this simulation provided enough data to try the designed implementation of the engine.

The purpose of this example is to show that is possible to call an external implementation of a method from this engine. In this case, a mode estimation method based on ambient data [107], implemented in Matlab, has been used. For simplicity (and for the unavailability of real measurements), two simulations have been used and labeled *Simulation Signal* and *Reference Signal* (see Figure 67).

The method is executed from a Python script that gets, as inputs, 1) the simulation signals from the model, 2) the model order for fitting the measured system response from the model [107], and 3) the reference signals from the measurement, to execute the ambient analysis method. Both simulation signals and reference signals are stored in their corresponding .h5 file. The Python script first shows the available signals from both files to allow the user to select which signals will be analyzed. Next, the script calls the Matlab method (in this case) and executes it in the background. When the execution is finished, the script shows the results of the mode estimation method and plots the signals used for this analysis (see Figure 67).

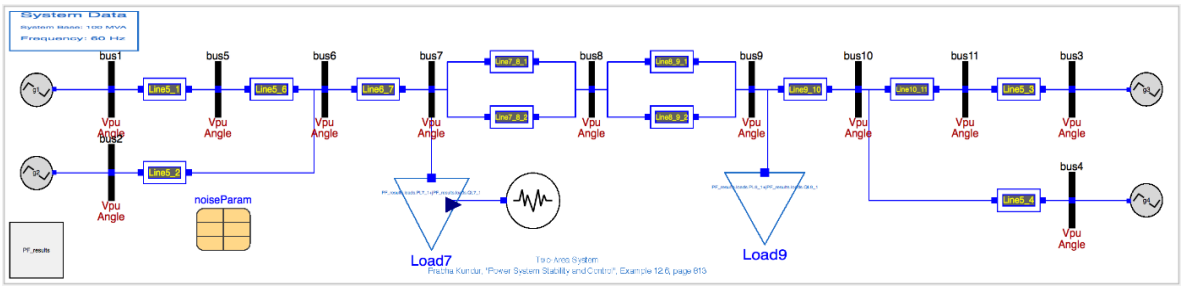


*Figure 66. Power grid Modelica representation of a two-area power system.*

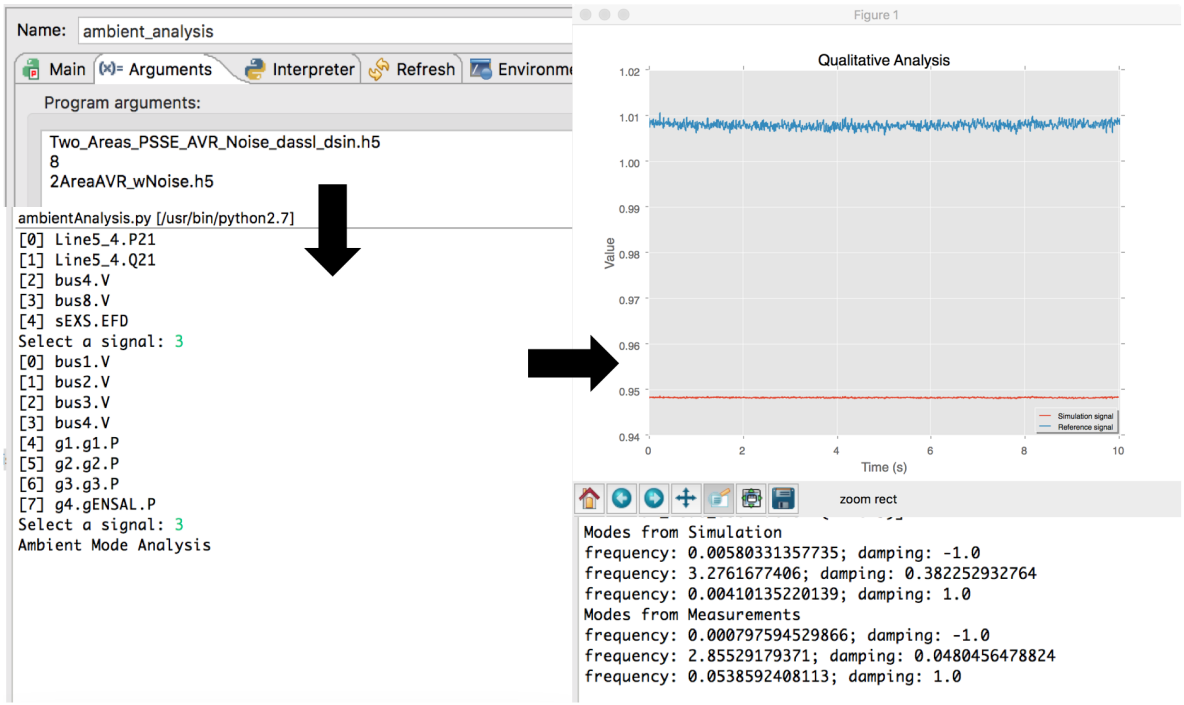


*Figure 67. On the left, the script interface that call a Matlab function. In the center, Eclipse environment to call the Python script with 3 parameters. On the right, the signals that have been analyzed by the ambient analysis method. At the bottom, the modes for frequency and damping of each signal.*

### 6.3.3. Data Manager Engine

The example in Figure 68 shows the results of implementation of the data structures depicted in Figure 61 – for handling the execution of signal and model analysis methods – and Figure 62 – for handling the access and storage of simulation results and measurements. For this example, the default view for HDF5 format is used. The results from the execution engine are stored in the file *Two_Areas_PSSE_AVR_dassl_sim.h5* and its data set structure follows the CIM class convention shown in Figure 62: *Line5_4* is

the name of the (*IdentifiedObject*) Measurement and *P2_1* is the identifier of the *Analog* object. The dataset *AnalogValue* stores the signal corresponding to that measurement. The same structure is applied for the references signals or measurements. The results of the analysis are stored following the hierarchy of method, signal, and metric (depicted in Figure 68 in under the file *results_modeEstimation_ord38_2AreaAVR_wNoise.h5*).

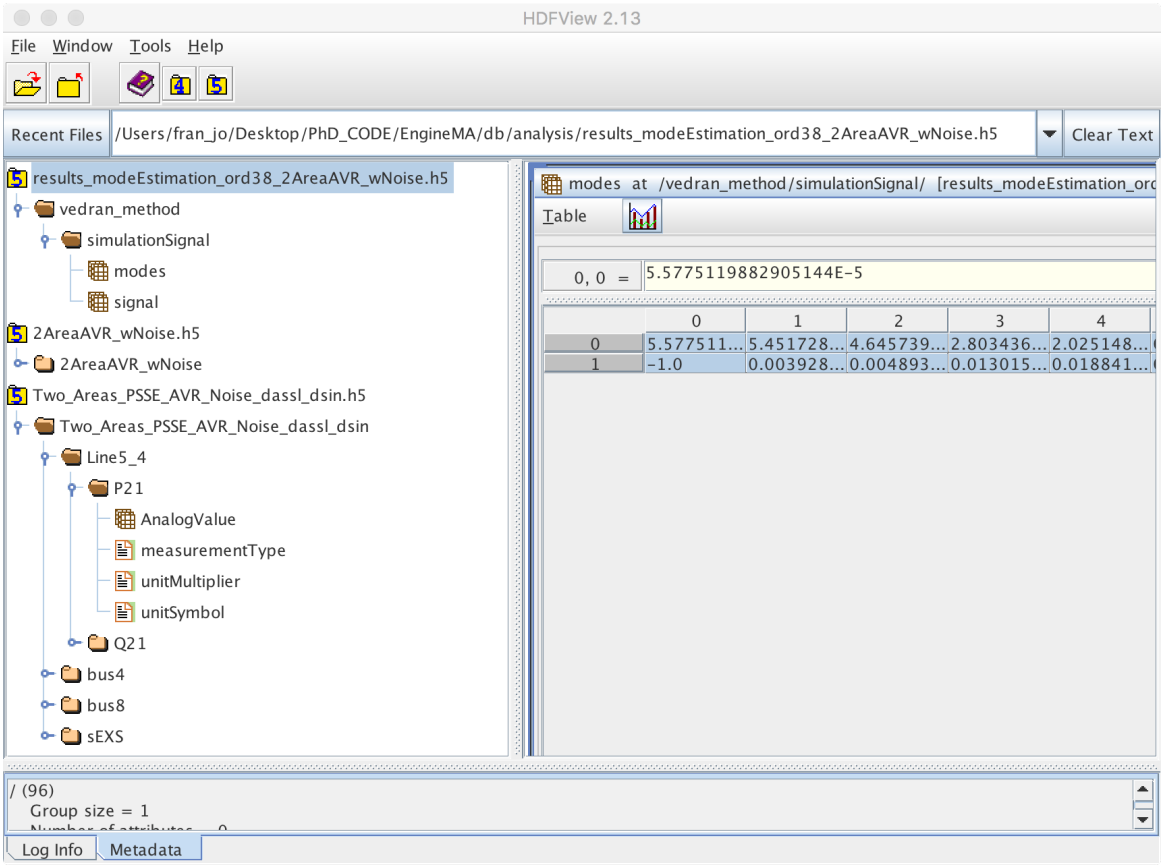


*Figure 68. Detail of internal database structure in HDF5. The table shows the results of a signal analysis. The tree structure on the left also shows how signals are organized following a CIM-class organization.*

## 6.4. Summary

The different computational engines were developed in such a way that they require minimum knowledge of the software standards discussed herein, such as the Modelica, the FMI, or the HDF5. The engines developed - until this stage of the whole development process (see Chapter 2 section 2.4.1) – provide a simple API that make use of each engine independent from each other. Towards the implementation of this architecture, examples of the Data Modeling Engine (see Chapter 5), the Model Execution Engine, the Measurement Analysis Engine, and the Data Manager Engine have been presented.

The implementation of the MEE was first addressed at the beginning of the architecture development process, to prove that Modelica compilers are suitable for power system simulations. With the use of Python scripts, OpenModelica, JModelica and Dymola compilers are integrated into the engine. The OpenModelica compiler is suitable for simulating Modelica models and the JModelica compiler allows one to simulate Modelica models or FMU models translated from other simulation tools. The Dymola compiler was also implemented to prove that open-source compilers could match the outcomes produced by commercial tools [20] [45].

The development of the Data Modeling Engine was addressed in a second step of the development process, and it constitutes the main core of this thesis. Its objective is to address the automatic and consistent provision of initial conditions, from power flow solution – a steady-state snapshot of the systems – to obtain valid simulation outputs, from a well-defined dynamic simulation models. The work of [73] explores the UML representation of the CIM standard in translating CIM data models into a Modelica model, which could be represented in more detail using SysML representation as it has been discussed in Chapter 4 section 4.3.

In the following steps of the process, the HDF5 format was included for storing simulation outcomes and real measurements. The internal data model implementation of the MAE's and DME's functionalities was based on the utilization of this data format. Thus, the notation of group and dataset from the HDF5 was adapted to store data following the CIM-based data structure notation of *PowerSystemResource* and *Measurement*. This prevents the need to integrate a complete database system based on query language.

# Chapter 7

# Conclusions & Future Work

Most of the research focused on power systems engineering are concerned with the development of mathematical computation algorithms either for steady-state analysis or for dynamic analysis. Therefore, the development of network models is still strictly attached to the development of mathematical algorithms specific to the electrical engineering discipline. In contrast, the integration of software engineering methodology and semantics enables power systems research to better cover the needs from multi-domain disciplines and system integration, which are necessary to cope with complex use cases involving cyber-physical concepts and principles.

This is where the Model-Driven Development (MDD) and Model-Based Software Engineering (MBSE) perspectives can be of great value. Nevertheless, only a few studies have been carried out on the development of new software tools supporting new software semantics and technologies for power systems analysis. Such is the case with the integration of the FMI standard for the implementation of model-based analysis tools. However, most of these examples are utilized within an academic environment, or they are reported as proofs-of-concept within technical reports. Thus, most of the implementation of efficient research methods in power system engineering continues with the utilization of the few commercial analysis tools available in the market and even fewer OSs, which have been used within the industry for decades.

From the point of view of MDD and MBSE, this thesis is not so different from the available research on new development tools, but it provides a theoretical background for these software-engineering concepts and methods to become applicable to the development of new models and tools for the power systems field. The first goal of this thesis was to develop a link to existing information exchange standards from Modelica, as a good candidate for developing power system dynamic models, without ambiguities. This means that those kinds of representations could be developed most efficiently, using a dedicated modeling language that is not related to any computational analysis algorithm, i.e. Modelica. This could leverage the integration and reusability of those models without depending on any proprietary data structure or any programming language.

This thesis is composed of six chapters with introduction, development of ideas and conclusions, a library of references and proofs-of-concept, for each of the topics described within. This organization comes from the recompilation of publications finished during the development of this thesis. Introductory remarks and the literature review of the problems addressed by this thesis can be found in Chapter 1 and Chapter 2. The idea behind Chapter 3 is to exploit an abstract level of requirements to gather different elements from the software and electrical engineering disciplines and put them together for a description of how to address the development of new tools and what would be required to report when research on these topics is carried out. From the work carried out during the ITEA 3 OpenCPS EU Project, this chapter exploits the concept of Requirements Engineering to establish a common ground for the integration of new software technologies.

Chapter 4 explores the development of the extension of power systems information semantics with semantics from other disciplines. The aim of this chapter is to add more value to power system CIM models with enhancements to the CIM information semantics with the development of UML-based ISO 15962 vocabulary. The integration of different engineering semantics makes it necessary to find other means for model representation that could be useful for understanding different systems integration. This has been shown either by RDF language implementation or SysML component representation. Then, due to the different semantic representations of equivalent network models, Chapter 5 describes implementation results as to how to produce the mapping between different modeling languages for the automatic creation of dynamic simulation models.

Finally, Chapter 6 describes a scalable and modular workflow design for an architecture prototype that divides the whole power system analysis process into specialized computational engines that can meet the requirements outlined in Chapter 3. The design offers a complete functionality for power systems analysis: automatic network modeling, running simulations, validating simulation outputs from real measurements, and identification and calibration of parameters that adjust the model to match the real behavior of the equivalent “real” system. This design offers a proof-of-concept of how software applications could be developed to put together the modeling and simulation standards presented in the thesis.

In addition to the design results and implementation results, this thesis tries to provide a theoretical groundwork of using elements from the software engineering discipline for systems integration considering emerging needs of the new cyber-physical electrical systems. As the cyber-physical systems-based solutions become more complex – in terms of information representation and mathematical details – there is a need to provide understandable theories and methodology focused on the development of actions, methods, and tools. General Purpose Modeling Languages (GPML), such as the UML and the SysML, are of great utility for the efficient description of new developments, providing graphical support for the natural language used when new methods of research have to be explained.

The development process discussed in this thesis is continuously under maintenance and revision, with the aim of incorporating new features to the whole architecture presented herein. The proof-of-concept design provides a new approach in the power systems domain on to the use of software-modeling languages, such as the UML, to describe software-related features for new tools. Future work on this proof of concept should address further design and development for the SSE engine and enhancements of the PM engine.

# APPENDIX A

# Mapping Implementation Details

This chapter tries to be a developer's guide on how to implement additional mapping rules' code for the CIM 2 MODelica Transformation Tool. This is a developer's guide on ho to implement additional mapping rules' code for the CIM 2 MODelica Transformation tool. This guide tries give details on the developing environment and the technical understanding for the maintenance and enhancement of the transformation tool. Within in this guide, the algorithm for model transformation will be described, as well as a description of the necessary steps for the development of mapping files and for the development of new classes and methods.

## I. Environment set-up

The current version of the development environment works with Eclipse Neon 4.6.3 and JAVA 1.8. (we suppose that you have available a Git Client, Eclipse and JAVA installed into the system).

1. Download the source code from the repository ( https://github.com/ALSETLab/cim2modelica ), with a Git Tool, i.e., Github Desktop
    a. Make sure you have downloaded the .project and .classpath file
2. Open Eclipse and Import the project into the current workspace: *File> Import…> Existing project into Workspace.*
3. In the Import dialog, browse on the Select Root Directory field and select the project from the list of Projects

The project includes the reference of the original build path. Each developer might update their own build path according to per their JAVA configuration. It is recommended not to update any changes on the .project nor .classpath files.

## II. Quick view of the project structure

The project contains:

1. A ***dist*** folder to store the code releases (also can be used as test directory)
2. A ***lib*** folder to store the external libraries to be used within the project
3. A ***model*** folder, where the tool stores the output models
4. A ***res*** folder, to store all the resources used by the tool: mapping files and input models
5. A ***src*** folder, with the package structure of the tool. The main packages are 1) cim2model; 2) cim2model.cim; and 3) cim2model.modelica

The project includes also a couple of external libraries that have been originally included in the project build path (Apache JENA and JAVA JAXB). In the Project> Preferences> Java Build Path option the project should contains the references to User Libraries JENA and JAXB. If they do not appear in the Java Build Path, you need to create two User Libraries with .jar files from JENA and JAXB.

User Library is created for each folder:

1. Go to Windows> Preferences> Java> Build Path> User Libraries and click on New
2. Give a name to the library
3. Select the recent created library and click on Add JARs
4. Expand the option cim2modelica> libs> apache-jena and select all the .jar files under that folder
5. The .jar files are stored under the new User Library.
6. Do the same for the JAXB library.

Now we add the recent library to the project:

1. Go to Project>Properties> Java Build Path
2. Click on Add Library and select the libraries you have created.
3. The library appears within the project

## III. Run Configuration set-up

To run the transformation tool from the Eclipse, you need to create a Run Configuration.

1. Run> Run Configurations> Java Application> New Launch Configuration
2. Select the corresponding project in Main> Project,
    a. Select the main class, i.e., cim2model.CIM2MOD
3. Specify the arguments of the tool. Arguments> Program Arguments:
    a. <relative_path_CIMFiles_Folder>
    b. <name_of_the_network>

## IV. Software development in detail

This section describes how to implement new mapping files and how to extend the tool class structure with new JAXB Classes.

### I. Create Mapping Files

The folder res.map.openipsl contains the .xml, .xsd and .dtd files that conform the mapping rules for each OpenIPSL component available. To comply with the OpenIPSL packages' organization, the folder res.map.openipsl is organized in the same way as the library (e.g. the folder res.map.openipsl.controls.es contains mapping rules for Excitation System components).

To add a new mapping component rule, follow the next steps:

1. Copy one of the existing *.dtd, .xml, and .xsd* with the new component name. Follow the name syntax of the files. Place the new files into the corresponding folder.
2. Modify the *.dtd* changing the name the component **<!ELEMENT** and **<!ATTLIST** tags with a new name, i.e.: *<!ELEMENT iEEET1Map* replaced by *<!ELEMENT sEXSMap*)
3. Modify the *.xsd* changing the name of the **<xsd:element** main tag, i.e.: *<xs:element name="iEEET1Map"* replaced by *<xs:element name="sEXSMap">*
4. Modify the .xml changing the name of the .dtd file in the DOCTYPE tag:

```
1.  < !DOCTYPE model SYSTEM "cim_openipsl_sexs.dtd" >
```

5. Modify the *.xml* changing the name of the main tag, and the values of the main tag attributes with the corresponding OpenIPSL names (you can leave the tags *rdf_id* and *rdf_resources* empty)

```
<sEXSMap cim_name="ExcSEXS" rdf_id="" rdf_resource=""
package="OpenIPSL.Electrical.Controls.PSSE.ES" stereotype="class">
```

6. Modify the .xml changing the number of <attributeMap> tags, and their parameter values, per number of parameters of the OpenIPSL component.

```
<attributeMap cim_name="ExcSEXS.tatb" name="T_AT_B" datatype="Real"
variability="parameter" visibility="public"> 0 </attributeMap>
```

## II. Create Mapping Files

Using the API provided by the JAXB Library, we can create for each XML mapping rule its corresponding JAVAX class, which will be integrated within the transformation tool. There are two ways of doing this:

1. First option is to create a new *Run Configuratio*n, within the Eclipse environment, with the class *MappingStructureGenerator.java* as main class. This class is prepared for placing the resulting JAVAX classes into the corresponding tool packages. Create the Run Configuration specifying two program arguments:
    a. Name the package to store the JAVAX class, e.g.: `cim2modelica.cim.map.openipsl.controls.es` (in case of the mapping of a new excitation system component)
    b. Relative path with the name of the mapping schema file, e.g., `./res/map/openipsl/controls/es/cim_openipsl_sexs.xsd`)

2. Second option is to use the JAXB tool, XJC, in the command line. The XJC executable will generate an additional external package, in the folder you have executed the XJC command, with the generated classes. They need to be included into the tool class structure manually.

## III. Modify the code from the generated JAVAX classes

After the execution of the XJC tool three JAVAX classes are created: SEXSMap.java (following the example of the mapping of the excitation system), AttributeMap.java and ObjectFactory.java. Because most of the mapping rules share the same parameters, the package cim2modelica.cim.map contains and abstract class ComponentMap.java that contains the general attributes and the getters/setters methods. To adapt the generated classes to the Mapping Meta-Model structure of the project, follow these steps:

1. In the SEXSMap.java add the import statement:

```
import cim2modelica.cim.map.ComponentMap;
```

2. Update the class declaration with:

```
public class SEXSMap extend ComponentMap
```

3. In case of this component, you can delete all the JAVAX elements and the *getters/setters* methods, leaving the class empty. It inherits every attribute and

method from the ComponentMap class. Just leave those attributes and methods that do not appear within the ComponentMap class, i.e.:

```
package cim2modelica.cim.map.openipsl.controls.es;
import cim2modelica.cim.map.ComponentMap;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement(name="sEXSMap")
public class SEXSMap extends ComponentMap
{…}
```

4. The package cim2modelica.cim.map already contains the class AttributeMap.java. Thus, you can delete the newest one.
5. The generated *ObjectFactory.java* class can be discarded because we use the JAXB API to create a specific factory class for the new JAVAX class. This factory class, contains a factory method that unmarshalls the values from the corresponding .xml mapping file into memory.
6. In this example, copy/paste an existing factory method within the same ExcSysMapFactory.java class and adapt its code to the new component name: (Each package of the *Mapping Meta-Model* structure contains a factory class, to group the factory methods per components).

```
public SEXSMap sexsXMLToObject(String _xmlmap) {
    JAXBContext context;
    Unmarshaller un;
    try {
        context = JAXBContext.newInstance(SEXSMap.class);
        un = context.createUnmarshaller();
        SEXSMap map = (SEXSMap) un.unmarshal(new File(_xmlmap));
        return map;
    } catch (JAXBException e) {
        e.printStackTrace();
        return null;
    }
}
```

## IV. Updated controller classes to use the new component map

1. Updated the ModelDesigner.java class, adding a new method to populate the values of the new component map. Just copy one of the existing create_ methods and adapt it to the new mapping object:

```
public SEXSMap create_SEXSModelicaMap(
       Resource _key, String _source, String _subjectName)
{…}
```

2. Updated the ModelBuilder.java class, adding a new method to create the OpenIPSL component instance with the values of the new component map. Just copy one of the existing create_ methods and adapt it to the new mapping object:

```
public MOClass create_SpecificComponent(SpecificComponentMap _map)
{…}
```

3. See that the component created by the ModelBuilder controller class return objects of type MOClass. In case of a new ExcitationSystem component you can copy the next declaration:

```
public OpenIPSLExcitationSystem create_SEXSComponent(
```

```
        ComponentMap _mapExcSys)
{…}
```

4. Last step is to update the identification process of the CIM classes, within the main CIM2MOD.java class. The algorithm first starts with the identification of the CIM Terminals. Then, it identifies the ConductingEquipement and TopologicalNode classes associated to the Terminal.

```
cimClassResource= cartografo.get_EquipmentClassName(key);
if (cimClassResource[1].equals("Terminal"))
{…
equipmentResource=
   cartografo.get_EquipmentClassName(cartografo.get_CurrentConnectionMap().
   get_ConductingEquipment());
topologyResource=
   cartografo.get_TopoNodeClassName(cartografo.get_CurrentConnectionMap().
   get_TopologicalNode());
…}
```

5. Then, for each equipmentResource and topologyResource their corresponding mapping rule is loaded with the appropriate method from the ModelDesigner class. With the case shown in this guide, the ExcitationSystem component is identified within the static method factory_plant, used when the equipmentResource is a CIM SynchronousMachine class:

```
if (equipmentResource[1].equals("SynchronousMachine"))
{…
    factory_Plant(momachine, machineType, mopin);
}…
/**
* Creates plant object given MachineMap, adds esmap, tgmap and stabmap
* MachinMap can contain ES[0..1], TG[0..1], PSS[0..1]
* @param _momachine
* @param _machineType
* @param _mopin
*/
public static void factory_Plant(
       OpenIPSLMachine _momachine, String _machineType,MOConnector _mopin)
{…
     switch (excSysData.getKey())
     {
          case "SEXS": mapExcSys= cartografo.create_SEXSModelicaMap(
          excSysData.getValue(),"./res/map/openipsl/controls/es/cim_openip
sl_sexs.xml",
          excSysData.getKey());
      …}
      moexcsys = constructor.create_ExcSysComponent(mapExcSys);
…}
```

## V. Algorithm for Model-2-Model

The algorithm builds a network model starting from the identification of the Terminals within the CIM Model. The mapping rules for the OpenIPSL *PwPin* component are loaded into the internal model of the tool, and are used to gather the corresponding values associated to each *Terminal*, with the *rdf:id* attribute of the CIM *Terminal* class:

- Using the mapping rules, the *ComponentMap* XML file is loaded into the corresponding *ComponentMap* object.
- Terminal data from the EQ profile, which contains general information of the Terminal and the *rdf:resource* of the associated *ConductingEquipment* and *TopologicalNode* objects.
- State Variable data from the SV profile, which contains P and Q values.
- Topology data from the TP Profile, which contains V and angle values.

This information is used to create the *PwPin* component, which is an instance of the *MOConnector* class from Figure 48. Additionally, A *ConnectionMap* object is created with the *rdf:id* data of the Terminal, the associated *ConductingEquipment* and *TopologicalNode* objects. A set of *ConnectionMap Objects* are used in the last step, to create the model connections of the components.

After processing of *Terminal* class into a *PwPin* component, the *Populate Mapping Objects* and *Instantiate Network Objects* actions of the main workflow are executed to create the corresponding OpenIPSL components related to *ConductingEquipment* and *TopologicalNode*. Figure 69 shows the sequence of messages, and the objects used to create a *Machine* component and the existing regulator components associated to the machine:

- The specific XML file of a machine component is loaded into the corresponding *SynchMachineMap* object, e.g. mapping rules for a GENROU component.
- General data from the EQ profile and Dynamic data from the DY Profile are loaded.
- An *OpenIPSLMachine* object is created – as an instance of the *MOClass* – with the corresponding data.
- The previous *PwPin*, which is related with the current Machine component, is added to this new OpenIPSLMachine component.

Then, the mapping rules for regulator component is loaded, e.g. for an Excitation System component:

- The specific XML file, with the mapping rules for an *Excitation System* component, is loaded into the *ExcSysMap* object.
- Dynamic data from the DY Profile is loaded.
- An *OpenIPSLExcitationSystem* object – which is also an instance of the *MOClass* – is created with the corresponding data.

Once the machine component and its regulators are created, they are encapsulated into a *MOPlant* object, which is specifically design to create sub-models of power plant configuration. With the method ***assemble_Plant*** (in Figure 70) the internal connections of inputs and outputs of the machine component and regulators are created. Thus, an object of type *MOPlant* created and added to the main network object. For the rest of the main components e.g. *Lines, Loads and Buses*, the procedure follows the sequence on Figure 71 – it shows the sequence to create a Bus component:

- The specific XML file of *ComponentMap* is loaded into the corresponding *ComponentMap* object.
- General data from the EQ profile is loaded.
- An *MOClass* object is created, with the corresponding data.

The automated creation of *PowerTransformer* component follows a different approach (see Figure 72), because of the relation of the CIM *Terminal* class and *PowerTransformerEnd* class and its characteristics:

- Its relation is 0..1 to N, it is easier to start the mapping algorithm by getting the *PowerTransformerEnd* class and then its associated Terminal.
- The relevant parameters of the *OpenIPSL PowerTransformer* component are in the CIM *PowerTransformerEnd* class.

Thus, a single *MOClass* representing a *PowerTransformer* component is updated with every *PowerTransformerEnd* map – due to the characteristics of the OpenIPSL *PowerTransformer* model.

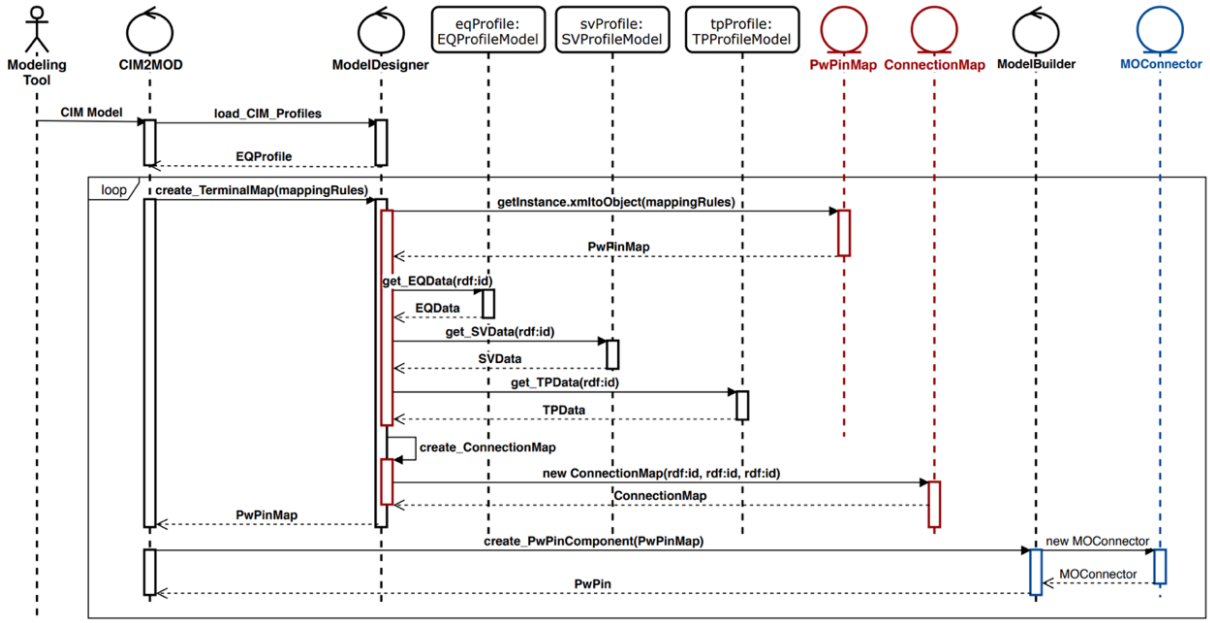


*Figure 69 Simplified sequence diagram with the main messages and objects involved in the creation of PwPin component*

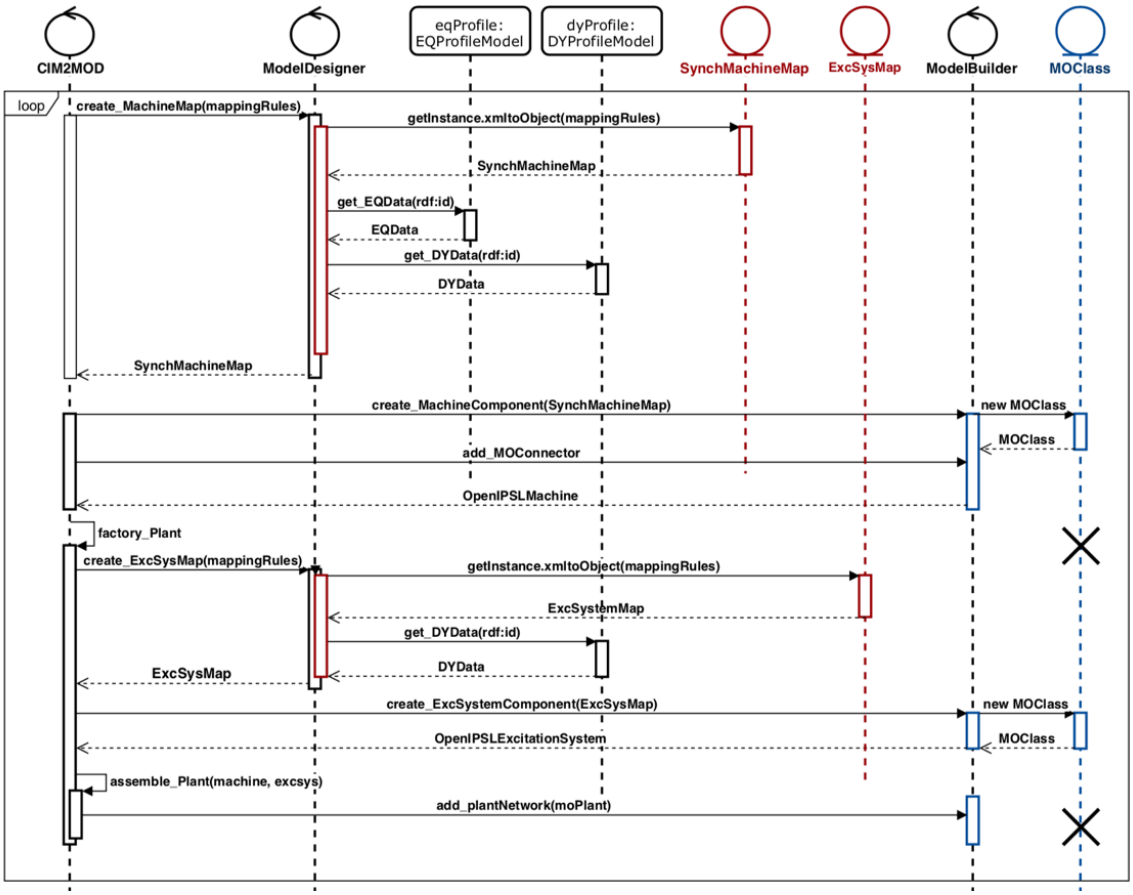


*Figure 70 Simplified sequence diagram with the main messages and objects involved in the creation of Machine and Excitation System component. The same life line of the MOClass object is used to illustrate the creation of different components.*

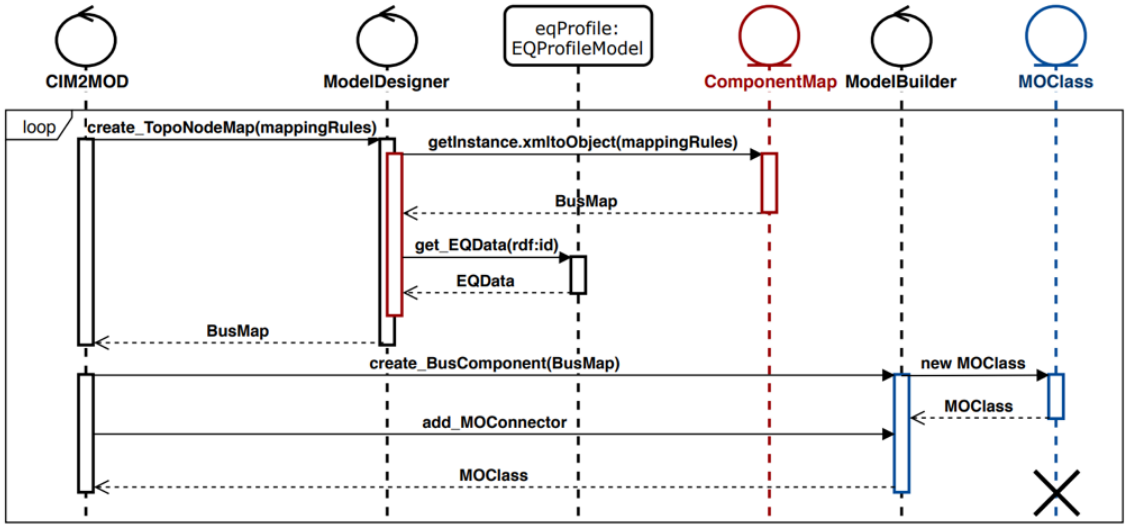


*Figure 71 Sequence diagram with the main messages and objects involved in the creation of a Bus component. Similar sequence structure applies for Line and Loads.*

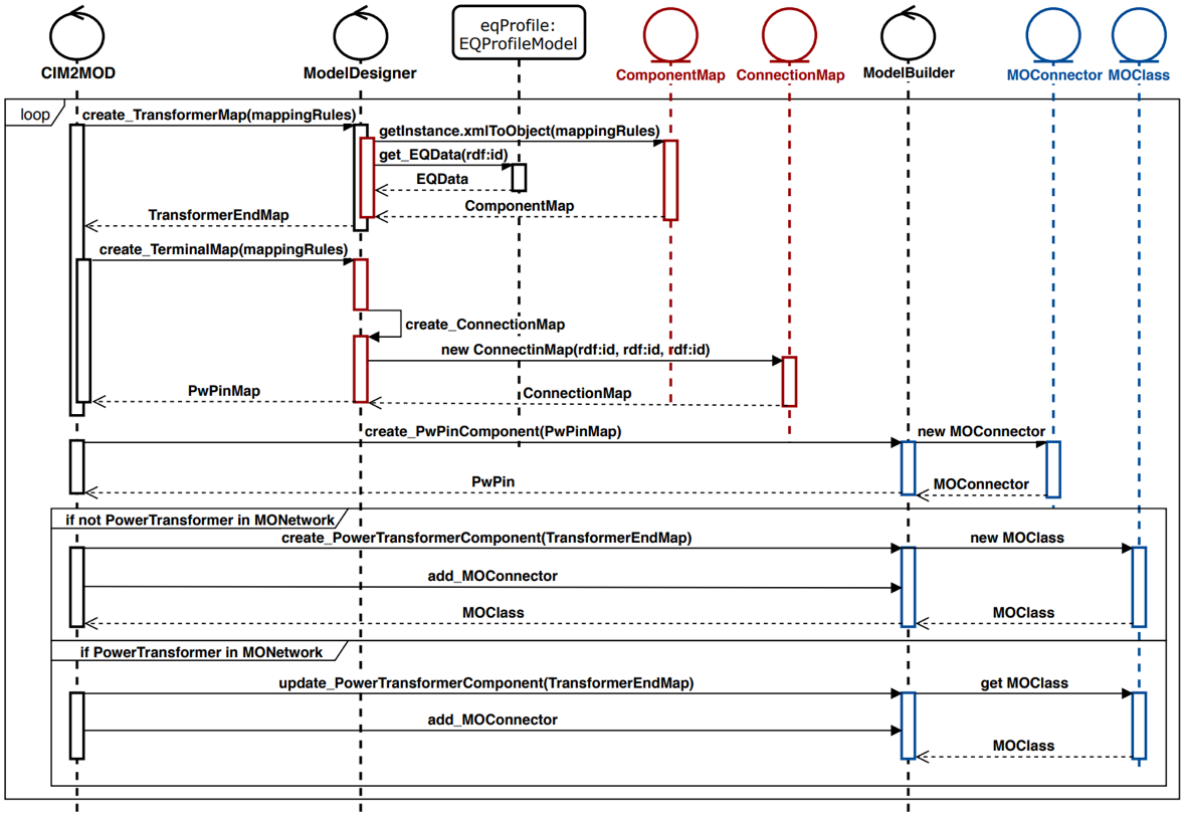


*Figure 72 Simplified sequence diagram with the main messages and objects involved in the creation of PowerTransformer. The sequence considers both 2/3 winding transformers.*

# Bibliography


[1]. Europe's National Regulators Of Electricity And Gas At Eu And International Level [On-Line]: Http://Www.Ceer.Eu/Portal/Page/Portal/Eer_Home/Eer_Publications/Ceer_Papers/Electricity/2007/E06-Bag-01-06_Blackout-Finalreport_2007-02-06.Pdf, Accessed On 2015.

[2]. Ch. Ivanov, T. Saxton, J. Waight, M. Monti and G. Robinson, "Prescription for Interoperability: Power System Challenges and Requirements for Interoperable Solutions" in IEEE Power and Energy Magazine, vol. 14, no. 1, pp. 30-39, Jan.-Feb. 2016, doi: 10.1109/MPE.2015.2485798

[3]. M. Uslar, M. Specht, S. Rohjans, J. Trefke, and J. M Gonzalez. "The Common Information Model CIM: IEC 61970, 61968 and 62325". Springer Heidelberg, 2012.

[4]. L. O. Osterlund, K. Hunter, K. Demaree, M. Goodrich, A. McMorran, B. Iverson, T. Kostic, "Under the Hood: An Overview of the Common Information Model Data Exchanges" in IEEE Power and Energy Magazine, vol. 14, no. 1, pp. 68-82, Jan.-Feb. 2016, doi: 10.1109/MPE.2015.2485859

[5]. CEN-CENELEC-ETSI Smart Grid Coordination Group. "Smart Grid Reference Architecture". November, 2012. [On-line] Available: http://ec.europa.eu/energy/gas_electricity/smartgrids/doc/xpert_group1_reference_architecture.pdf

[6]. ENTSO-E Common Grid Model Exchange Standard (CGMES) [On- line], https://www.entsoe.eu/digital/common-information-model/#developing-cim-standards, Accessed on 2017.

[7]. ENTSO-E CIM Inter-Operability Tests, 2015. [On-line] Available: https://www.entsoe.eu/major-projects/common-information-model-cim/interoperability-tests/Pages/default.aspx

[8]. ENTSO-E, "Making non-mandatory requirements at European level mandatory at national level ENTSO-E Guidance document for national", [On-line], https://electricity.network-codes.eu/network_codes/cnc/cnc-igds/, 2016.

[9]. EnergyNet.dk, "Technical Regulations 3.2.2. for PV Power Plants above 11KW", July 2016, Revision 4 [On-line], Available: https://en.energinet.dk/Electricity/Rules-and-Regulations/Regulations-for-grid-connection

[10]. Arora, C., Sabetzadeh, M., Briand, L.C. and Zimmer, F., “Extracting domain models from natural-language requirements: approach and industrial evaluation”, MODELS 2016, pp. 250–260, ACM, doi: 10.1145/2976767.2976769

[11]. H. Kaindl, S. Brinkkemper, J. A. B. Jr., B. Farbey, S. J. Greenspan, C. L. Heitmeyer, J. C. S. do Prado Leite, N. R. Mead, J. Mylopoulos, and J. I. A. Siddiqi, “Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda.” Requirements Engineer- ing, vol. 7, no. 3, pp. 113–123, 2002.

[12]. E. Raymond, “The cathedral and the bazaar”, Knowledge, Technology & Policy, 1999, 12(3), 23–49. https://doi.org/10.1007/s12130-999-1026-0

[13]. “Open Cyber-Physical System Model-Driven Certified Development”, [On-Line], Available at: https://opencps.eu

[14]. Buffoni, L. and Fritzson, P., “Expressing Requirements in Modelica”, Proceedings of the 55th International Conference on Simulation and Modeling, 2014, October 21-22, Aalborg, Denmark, doi: 10.11128/sne.25.tn.10314

[15]. Carnegie Mellon University, “Managing Variable Energy Resources to Increase Renewable Electricity’s Contribution to the Grid,” 2013. (intro multi-domain)

[16]. S. K. Yee, J. V Milanovic, and F. M. Hughes, “Overview and comparative analysis of gas turbine models for system stability studies,” IEEE Trans. Power Syst., vol. 23, no. 1, pp. 108–118, 2008.

[17]. T. Holm, L. Christiansen, M. Göring, T. Jäger and A. Fay, "ISO 15926 vs. IEC 62424 — Comparison of plant structure modeling concepts," Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012), Krakow, 2012, pp. 1-8. doi: 10.1109/ETFA.2012.6489662

[18]. P. Fritzson, Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica. Wiley-IEEE Press, 2011. ISBN: 978-1-118-01068-6

[19]. L. Vanfretti, T. Rabuzin, M. Baudette, M. Murad, “iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations”, SoftwareX, Available online 18 May 2016, ISSN 2352-7110, http://dx.doi.org/10.1016/j.softx.2016.05.001.

[20]. F.J. Gómez, L. Vanfretti, Svein H. Olsen, “A Modelica-Based Execution and Simulation Engine for Automated Power Systems Model Validation”, Innovative Smart Grid Technologies (ISGT) Europe, Istanbul, Oct. 12-15, 2014, doi: 10.1109/ISGTEurope.2014.7028828

[21]. W. Schamai, P. Fritzson, C. Paredis, and A. Pop, “Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Excecutable Behavior Using Graphical Notations,” Proceedings of the 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009, pp. 612-622.

[22]. J. Nutaro, “Building software for simulation: Theory and algorithms, with applications in c++,” Wiley, Ed. Wiley, 2011.

[23]. “Innovative Tools for Electrical Systems Security within Large Areas”, [on-line], Available at: http://www.itesla-project.eu/

[24]. “European Commission Third Energy Legislative Package.” [Online]. Available: https://ec.europa.eu/energy/en/topics/markets-andconsumers/market-legislation.

[25]. International Electrotechnical Commission: IEC 61970-302: Energy management system application program interface (EMS-API) – Part 302: Common Information Model (CIM) Dynamics. International Electrotechnical Commission, 2015

[26]. Resource Description Framework for Semantic Web, 2014, [On-line] Available https://www.w3.org/RDF/

[27]. D. Leal, "ISO 15926 "Life cycle data for process plant": An overview", Oil & Gas Science and Technology - Rev. IFP, Vol. 60, No. 4, pp. 629-637, 2005.

[28]. ISO 15926-1, “Industrial automation systems and integration – Integration of life-cycle data for process plants including oil and gas production facilities – Part 1: Overview and fundamental principles”, 2004.

[29]. ISO 15926-2, “Industrial automation systems and integration - Integration of life-cycle data for process plants including oil and gas production facilities - Part 2 Data model,” 2003.

[30]. ISO 15926-4, “Industrial automation systems and integration -Integration of life-cycle data for process plants including oil and gas production facilities - Part 4 Initial reference data,” 2007.

[31]. ISO 15926-8, “Industrial automation systems and integration - Integration of life-cycle data for process plants including oil and gas production facilities - Part 8 Implementation methods for the integration of distributed systems: Web Ontology Language (OWL),” 2011.

[32]. A. L. Opdahl, "A Platform for Interoperable Domain-Specific Enterprise Modelling Based on ISO 15926", 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, Vitoria, 2010, pp. 301-310. doi: 10.1109/EDOCW.2010.29

[33]. F. Milano, “Power Systems Modeling and Scripting”, Springer, 2010, doi: https://doi.org/10.1007/978-3-642-13669-6b.

[34]. M. Baudette, M. Castro, T. Rabuzin, J. Lavenius, T. Bogodorova and L. Vanfretti, “OpenIPSL: Open-Instance Power System Library — Update 1.5 to iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations”, SoftwareX, Volume 7, January–June 2018, Pages 34-36, ISSN 2352-7110, doi: https://doi.org/10.1016/j.softx.2018.01.002.

[35]. D. Zimmer, “A new framework for the simulation of equation-based models with variable structure”, Journal SIMULATION, Vol. 89, num. 8, pp. 935-963, 2013, doi: https://doi.org/10.1177/0037549713484077.

[36]. P. Fritzson, “Principles of Object-Oriented Modeling and Simulation with Modelica.”, Wiley-IEEE Press, 2003, doi: https://doi.org/10.1002/9780470545669

[37]. M. Tiller, “Introduction to Physical Modeling with Modelica”, Springer, 2001, doi: https://doi.org/10.1007/978-1-4615-1561-6

[38]. P. Fritzson, P. Aronsson, A. Pop, H. Lundvall, K. Nyström, L. Saldamli, D. Broman, A. Sandholm, “OpenModelica - A Free Open-Source Environment for System Modeling, Simulation, and Teaching”, IEEE International Symposium on Computer-Aided Control Systems Design, October 4-6, 2006, Munich, Germany.

[39]. R. Viruez, S. Machado, L.M. Zamarreño, G. León, F. Beaude, S. Petitrenaud & J.B. Heyberger, “A Modelica-based Tool for Power System Dynamic Simulations”, 2017, 235–239. https://doi.org/10.3384/ecp17132235

[40]. J. Akesson, K.E. Arzen, M. Gafvert, T. Bergdahl and H. Tummescheit, “Modeling and optimization with Optimica and JModelica.org— languages and tools for solving large-scale dynamic optimization problems,” Computers and Chemical Engineering, vol. 34, no. 11, pp. 1737– 1749, Nov. 2010.

[41]. J. Mahseredjian, V. Dinavahi, and J. Martinez, “Simulation Tools for Electromagnetic Transients in Power Systems: Overview and Challenges,” IEEE Transactions on Power Delivery, vol.24, no.3, pp.1657–1669, 2009.

[42]. L. Vanfretti, W. Li, T. Bogodorova, and P. Panciatici, “Unambiguous power system dynamic modeling and simulation using Modelica tools," IEEE Power and Energy Society General Meeting (PES), July 2013.

[43]. M.A. Adib Murad, F.J. Gomez, and L. Vanfretti, “Equation-Based Modeling and Simulation of Three-Winding, and Regulating Transformers using Modelica,” IEEE PowerTech 2015.

[44]. M.A. Adib Murad, F.J. Gomez, and L. Vanfretti, “Equation-Based Modeling of FACTS using Modelica,” IEEE PowerTech 2015.

[45]. M. Sabate, G. Leon, M. Halat, J.B. Heyberger, F.J. Gómez and L. Vanfretti, “Aspects of Power System Modeling, Initialization and Simulation using the Modelica Language,” IEEE PowerTech 2015.

[46]. M. Zhang, M. Baudette, J. Lavenius, S. Løvlund, and L. Vanfretti, “Modelica Implementation and Software-to-Software Validation of Power System Component Models Commonly used by Nordic TSOs for Dynamic Simulations,” In Proceedings of the 56th Conference on Simulation and Modelling (SIMS 56), 7-9 October 2015, Linköking, Sweden

[47]. L. Vanfretti, M.A. Adib Murad, F. J. Gómez, G. León, S. Machado, J. B. Heyberger, S. Petriteneud, “Towards Automated Power System Model Transformation for Multi-TSO Phasor Time Domain Simulations using Modelica” IEEE PES Innovative Smart Grid Technologies Europe, Ljubljana, 2016

[48]. B. W. Boehm, "A spiral model of software development and enhancement," in Computer, vol. 21, no. 5, pp. 61-72, May 1988. doi: 10.1109/2.59

[49]. J. O. Clark, "System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective," 2009 3rd Annual IEEE Systems Conference, Vancouver, BC, 2009, pp. 381-387. doi: 10.1109/SYSTEMS.2009.4815831

[50]. R. M. Stallman, “Free Software, Free Society: Selected Essays of Richard M. Stallman”. Boston, Massachusetts: GNU Press, 2002.

[51]. Grid Open Source Software Alliance, 2013, [online] Available: http://gridossa.org/home/default.htm.

[52]. L. Vanfretti and F. Milano, “Facilitating constructive alignment in power systems engineering education using free and open-source software,” Education, IEEE Transactions on, vol. 55, no. 3, pp. 309–318, Aug 2012.

[53]. Unified Modeling Language Specification, OMG. (2015). [On-line] http://www.omg.org/spec/UML/2.5/

[54]. Systems Modeling Language Specification OMG (2015) [On-line] http://www.omg.org/spec/SysML/1.4/

[55]. Resource Description Framework for Semantic Web [On-line], https://www.w3.org/RDF/

[56]. D. Allemang and J. Hendler, Semantic Web for the Working Ontologist: Effective modeling in RDFS and OWL (Second Edition), Morgan-Kaufman, 2011.

[57]. OWL: Web Ontology Language [Online]. Available at: https://www.w3.org/standards/techs/owl#w3c_all

[58]. M. Brambilla, J. Cabot, and M. Wimmer, "Model-driven Software Engineering (MDSE) in Practice", Morgan & Claypool, USA, 2012, pp. 9-11, ISBN: 978-1-608-45882-0.

[59]. Functional Mock-Up unit Interface, [On-line], Available: http://fmi-standard.org

[60]. eXtensible Markup Language (XML), [On-Line], Available: https://www.w3.org/XML/

[61]. PTI Load Flow Data Format 2009 [Online]. Available: http://www.ee.washington.edu/research/pstca/formats/pti.txt

[62]. Matlab Mat-File format, [On-line], Available: https://www.mathworks.com/help/pdf_doc/matlab/matfile_format.pdf

[63]. M. Poinot, "Five good reasons to use the hierarchical data format," Computing in Science Engineering, vol. 12, no. 5, pp. 84–90, Sept 2010.

[64]. F. Milano, "Power Systems Modeling and Scripting", Springer, 2010, doi: https://doi.org/10.1007/978-3-642-13669-6

[65]. F. Milano, R. Zárate-Miñano, On the Stochastic Nature of Deterministic Power System Models for Dynamic Analysis, IEEE PES General Meeting, Boston, MA, 17-21 July 2016

[66]. M. dos Santos Soares, J. Vrancken, "Model-Driven User Requirements Specification using SysML", Journal of Software vol. 3, no. 6, pp. 57-68, 2008, doi: 10.4304/jsw.3.6.57-68

[67]. Papyrus Moka plug-in [On-line], Available: https://wiki.eclipse.org/Papyrus/UserGuide/ModelExecution, Accessed: 2018.

[68]. M. Gottschalk, M. Uslar, C. Delfs, "The Use Case and Smart Grid Architecture Approach", Springer, 2017, doi: https://doi.org/10.1007/978-3-319-49229-2.

[69]. F. Casella, "ThermoPower Gas Library," 2009. [Online]. Available: http://home.deib.polimi.it/casella/thermopower/help/ThermoPower_Gas.html#ThermoPower.Gas.

[70]. J. Köhler, H-M. Heinkel, P. Mai, J. Krasser, M. Deppe and M. Nagasawa, "Modelica Association Project: System Structure and Parametrization – Early Insight", Proceedings of the 1st Japanese Modelica Conference, May 23-24, 2016, Tokyo, Japan, doi: 10.3384/ecp1612435

[71]. M. Sjölund, F. Casella, A. Pop and A. Asghar, “Integrated Debugging of Equation-Based Models”, in Proceedings of the 10th International Modelica Conference, 2014, pp. 195–204, doi: 10.3384/ecp14096195

[72]. L. Vanfretti, M. Baudette, A. Amazouz, T. Bogodorova, T. Rabuzin, J. Lavenius, F.J. Gómez, “RaPId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models”, SoftwareX, Volume 5, 2016, Pages 144-149, doi: https://doi.org/10.1016/j.softx.2016.07.004.

[73]. F.J. Gómez, L. Vanfretti and S. H. Olsen, "CIM-Compliant Power System Dynamic Model-to-Model Transformation and Modelica Simulation," in IEEE Transactions on Industrial Informatics, 2017, doi: 10.1109/TII.2017.2785439

[74]. J. Cao, R. Wimmer, M. Thorade, T. Maile, J. O’Donnell, J. Rädler, J. Frisch & C. van, Treeck, “A Flexible Model Transformation To Link BIM With Different Modelica Libraries For Building Energy Performance Simulation”, 2015, Institute of Energy Efficient Building E3D, RWTH Aachen , Germany Berlin Universit. Building Simulation Conference, (2014), 434–441.

[75]. International Electrotechnical Commission: IEC 61970-501 Energy management system application program interface (EMS-API) - Part 501: Common Information Model Resource Description Framework (CIM RDF) schema.

[76]. L. Pereira, J. Undrill, D. Kosterev, D. Davies, and S. Patterson, New Thermal Turbine Governor Modeling for the WECC, WECC Modeling & Validation Work Group, Oct. 11, 2002.

[77]. F. P. De Mello and D. J. Ahner, “Dynamic models for combined cycle plants in power system studies,” IEEE Trans. Power Syst., vol. 9, no. 3, 1994.

[78]. W. I. Rowen, "Simplified mathematical representations of single shaft gas turbines in mechanical drive service", Int. Gas Turbine and Aeroengine Congr. and Expo., 1992.

[79]. Energinet, “Technical Regulation 3.2.3 for thermal plants above 11MW”, [Online]. Available: https://en.energinet.dk/Electricity/Rules-and-Regulations/Regulations-for-grid-connection

[80]. P. Pourbeik. (2013). “Dynamic models for turbine- governors in power system studies”. IEEE Task Force on Turbine-Governor Modeling.

[81]. A. W. McMorran, G. W. Ault, I. M. Elders, C. E. T. Foote, G. M. Burt and J. R. McDonald, “Translating CIM XML power system data to a proprietary format for system simulation” in IEEE Transactions on Power Systems, vol. 19, no. 1, pp. 229-235, Feb. 2004. doi: 10.1109/TPWRS.2003.820691.

[82]. Apache JENA: A free and open source Java framework for building Semantic Web and Linked Data applications [Online], Available at: https://jena.apache.org

[83]. F. J. Gómez, M. Aguilera Chaves, L. Vanfretti and S. H. Olsen, "Multi-Domain Semantic Information and Physical Behavior Modeling of Power Systems and Gas Turbines Expanding the Common Information Model," in IEEE Access, vol. 6, pp. 72663-72674, 2018. doi: 10.1109/ACCESS.2018.2882311

[84]. CIMSpy [On-line], available at: http://www.powerinfo.us/CIMSpy.html

[85]. CIMTool [On-line], available at: http://wiki.cimtool.org/index.html

[86]. Java Architecture for XML Building (JAXB), On-line: http://www.oracle.com/technetwork/articles/javase/index-140168.html

[87]. The Illinois Center for a Smarter Electric Grid (ICSEG), "WSCC 9-Bus System," Power Cases, [Accessed Apr. 11, 2015]. [Online]. Available: http://publish.illinois.edu/smartergrid/wscc-9-bus-system/

[88]. F. Casella, "Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives". In Proceedings 11th International Modelica Conference, Versailles, France, Sep 21-23, 2015, pp. 459-468. Online.

[89]. F. Casella, A. Bartolini, S. Pasquini and L. Bonuglia, "Object-Oriented Modelling and Simulation of Large-Scale Electrical Power Systems using Modelica: a First Feasibility Study". In Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society IECON 2016, Firenze, Italy, Oct. 24-27, 2016, pp. 0-6

[90]. System Integrity Protection Schemes (SIPS) [On-line], http://cimug.ucaiug.org/Groups/ENTSO-E_IOP/2016/Shared%20Documents/Reference%20Documents/20160714_CGMES_IOP_SIPS.pdf

[91]. ENTSO-E Common Grid Model Exchange Standard (CGMES), version 2.5 [On-line], https://www.entsoe.eu/Documents/CIM_documents/IOP/CGMES_2_5_TechnicalSpecification_61970-600_Part%201_Ed2.pdf

[92]. P.G. Larsen, J. Fitzgerald, J. Woodcock, R. Nilsson, C. Gamble, & S. Foster, "Towards semantically integrated models and tools for cyber-physical systems design". Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, 9953 LNCS, 171–186. https://doi.org/10.1007/978-3-319-47169-3_13

[93]. J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, J. Baras, S. Wang, "Toward a science of cyber-physical system integration", Proceedings of the IEEE, 2012, 100(1), 29–44. https://doi.org/10.1109/JPROC.2011.2161529

[94]. P. Palensky, A. A. Van Der Meer, C. D. Lopez, A. Joseph and K. Pan, "Cosimulation of Intelligent Power Systems: Fundamentals, Software Architecture, Numerics, and Coupling," in IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 34-50, March 2017. doi: 10.1109/MIE.2016.2639825.

[95]. J. Sztipanovits. “Cyber Physical Systems — Convergence of Physical and Information Sciences”, it - Information Technology, 2012, 54. 257-265. 10.1524/itit.2012.0688.

[96]. Dassault Systemes, Dynamic Modeling Laboratory (Dymola), [On-line], Available at https://www.3ds.com/products-services/catia/products/dymola, 2014

[97]. HDF Group, Hierarchical Data Format, [On-line], Available at https://www.hdfgroup.org/solutions/hdf5/, Accessed 2014

[98]. S. Shukla and Mark G. Yao., “An efficient method of extracting network information from CIM asset model”, 7th IEEE Innovative Smart Grid Technologies Conference, 2016, doi: 10.1109/ISGT.2016.7781190

[99]. F. Khomh & Y.G. Gueheneuce, “An Empirical Study of Design Patterns and Software Quality”. 2008 12th European Conference on Software Maintenance and Reengineering, 274–278. https://doi.org/10.1109/CSMR.2008.4493325

[100]. L. Vanfretti, M. A. A. Murad and F.J. Gómez, "Calibrating a VSC-HVDC model for dynamic simulations using RaPId and EMTP simulation data," 2017 IEEE Power & Energy Society General Meeting, Chicago, IL, 2017, pp. 1-5. doi: 10.1109/PESGM.2017.8274525

[101]. J.J. Billings, A.R. Bennett, J. Deyton, K. Gammeltoft, J. Graham, D. Gorin, A. Wojtowicz, “The eclipse integrated computational environment”, SoftwareX, 2018, num: 7, pp: 234–244. https://doi.org/10.1016/j.softx.2018.07.004

[102]. P. Palensky, A.A. Van Der Meer, C.D. López, A. Joseph, & K. Pan. “Cosimulation of Intelligent Power Systems: Fundamentals, Software Architecture, Numerics, and Coupling”. IEEE Industrial Electronics Magazine, 2017, num: 11(1), pp: 34–50. https://doi.org/10.1109/MIE.2016.2639825

[103]. A. Pfeiffer, I. Bausch-Gall, and M. Otter, “Proposal for a Standard Time Series File Format in HDF5”. 9th International Modelica Conference, 03.-05. Sep. 2012, Munich, Germany. ISBN 978-91-7519-826-2

[104]. F. Milano, "An Open Source Power System Analysis Toolbox", IEEE Transations on Power Systems, vol. 20, pp. 1199-1206, 2005.

[105]. K. Davies, ModelicaRes: Library to Analyze Modelica Simulations In Python, [On-line] Available at: http://kdavies4.github.io/ModelicaRes/, 2014

[106]. P. Kundur, “Power System Stability and Control”, New York: Mc-GrawHill, 1994.

[107]. V. S. Perić and L. Vanfretti, "Power-System Ambient-Mode Estimation Considering Spectral Load Properties", in IEEE Transactions on Power Systems, vol. 29, no. 3, pp. 1133-1143, May 2014. doi: 10.1109/TPWRS.2013.2292331