ABSTRACT

Title of thesis: INTEGRATION OF SYSML WITH TRADE-OFF ANALYSIS TOOLS

Dimitrios Spyropoulos, Master of Science, 2012

Thesis directed by: Professor John S. Baras
Department of Electrical and Computer Engineering and ISR

Changes in technology, economy and society create challenges that force us to rethink the way we develop systems. Model-Based Systems Engineering is an approach that can prove catalytic in this new era of systems development. In this work we introduce the concept of the modeling "hub" in order to realize the vision of Model-Based Systems Engineering and especially we focus on the trade-off analysis and design space exploration part of this "hub". For that purpose the capabilities of SysML are extended by integrating it with the trade-off analysis tool Consol-Optcad. The integration framework, the implementation details as well as the tools that were used for this work are described throughout this thesis. The implemented integration is then applied to analyze a very interesting multi-criteria optimization problem concerning power allocation and scheduling of a microgrid.

INTEGRATION OF SYSML WITH TRADE-OFF ANALYSIS TOOLS

by

Dimitrios Spyropoulos

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2012

Advisory Committee:
Professor John S. Baras, Chair/Advisor
Professor Mark Austin,
Professor David Lovell

Dedication

To my parents, Costas and Maria, and my sister Eirini.

Acknowledgments

I would like to sincerely thank and express my gratitude to my advisor Prof. John S. Baras for giving me the opportunity to work with him on very challenging and interesting research problems. With his technical expertise, his insightful comments and advice helped me to successfully complete this thesis. Dr. Baras continuous energy and passion for research will always follow me on whatever I do in the future.

I am also grateful to Prof. Mark Austin for the valuable knowledge I gained by attending his lectures and for the interesting discussions we had concerning the Systems Engineering field. Also, I would like to thank Prof. David Lovell for agreeing to serve on my committee and for his constructive feedback during the thesis examination.

A special thank to Anthony Anjorin from TU Darmstadt, for his support and cooperation.

I could not have completed this thesis without the continuous support and love from my parents, Costas and Maria, and my sister Eirini, and therefore I dedicate it to them. Moreover, I deeply thank my godparents, Kyrgiakos and Olga, for their support and encouragement.

These two years I met and became friend with exceptional people, whom I would like to take the chance to thank for their cooperation, the interesting discussions we had and the free time we spent together.

Finally, I would like to thank Kim Edwards for her efficiency in handling all

the administrative issues, and the ISR staff for always trying to do their best helping students with official matters.

This work was supported by the National Science Foundation (NSF) under the award number 10092651-52 to the University of Maryland and by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171.

Table of Contents

List of Figures

Chapter 1

The Era of Systems

1.1 Challenges

Last decade we entered a new era where systems complexity has increased dramatically. Complexity is increased both by the number of components that are included in each system and their heterogeneity as well as by the dependencies between those components. Developments in the field of network science and technology allowed for wireless and wireline interconnections between components, a fact that increased exponentially component dependencies. The so called networked systems are often also distributed and asynchronous, adding one more layer of complexity.

Moreover, today, systems tend to be more software dependent and that is another challenge that engineers and people involved in the development of such systems, face. The challenge is even greater when a safety critical system is considered, like the software controlling an airplane or a passenger car. There is a need for development of software that is by proof error-free. Moreover, when software dependent systems interact also with the physical environment then we have the class of cyber-physical systems (CPS). The key challenge in CPS is to incorporate the inputs and requirements from the physical environment in the logic of the embedded software.

Nowadays, more frequently we observe systems that cooperate to achieve a common goal, even though they were not built for that reason. These are called systems of systems. For example, the Global Positioning System (GPS) is a system by itself. However, it needs to cooperate with other systems when the air traffic control system of systems is under consideration. The analysis and development of such systems should be done extremely carefully mainly because of the emergent behavior that systems exhibit when they are coupled with other systems.

However, apart from the increasing complexity and the other technical challenges, there is a need to decrease time-to-market for new systems as well as the associated costs. This trend is expected to continue.

As it can be understood, due to these challenges and market pressure the whole process of how systems are designed and developed is changing dramatically. There is a need for rigorous ways to understand, analyze and develop systems. Percentages of systems that fail or have cost and schedule overruns confirm this need.

1.2 Systems Engineering of Tomorrow

1.2.1 Systems Engineering

1.2.1.1 Basic Characteristics of Systems Engineering

Systems Engineering is an interdisciplinary holistic approach in dealing with complex systems throughout their lifecycle [4]. It has its roots in the aerospace industry, which was the first that faced the challenge to develop extremely complex

systems. It focuses on establishing the right processes that will handle risk and will allow the development of a system on time, on budget and according to the stakeholders needs. Those processes start from the conceptual phase of the system and continue until its disposal. Systems Engineering takes into account all the factors that can affect the system during its lifecycle stages. Such factors are societal, economic, technical and organizational.

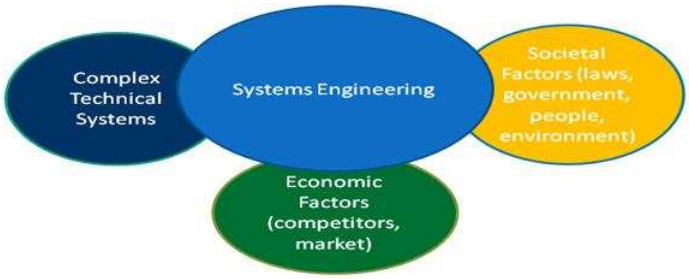

Figure 1.1: Systems Engineering Environment

The development of every system follows a framework that is know as lifecycle model. This framework emphasizes the stages of development and their sequence. There are many lifecycle models, although the one that is most favorable today is the V-model (Fig. 1.2). The V-model incorporates abstraction and decomposition that are two key concepts in Systems Engineering.

At the beginning of the development the user requirements are defined. Those requirements describe the concept of operations for the system, which mainly includes the desired behavior of the system and under what conditions the system shall operate. Moreover, at this initial stage a plan is developed to specify the way that the user requirements will be validated. The next stage includes the definition

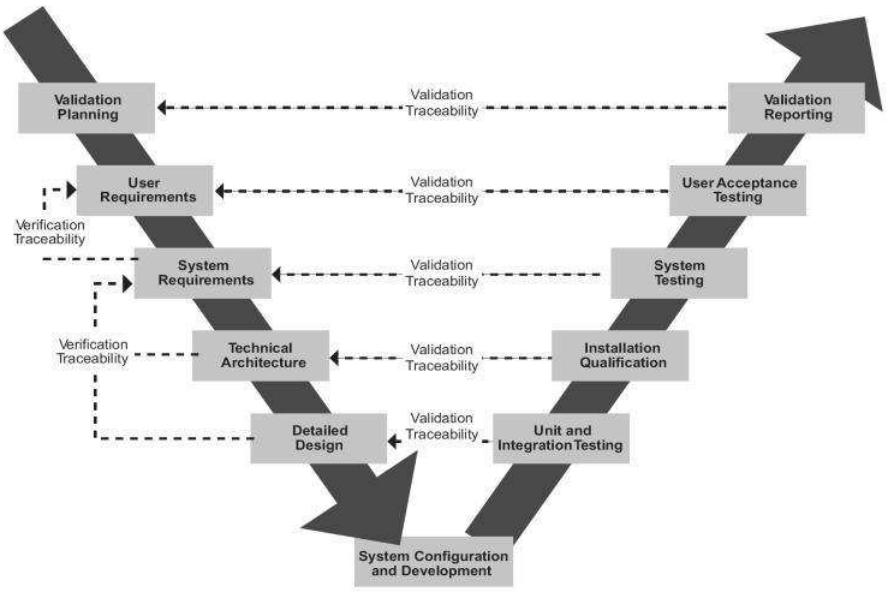

Figure 1.2: The V Lifecycle Model

of system requirements, which are technical requirements that describe in detail the system under development. System requirements should be developed in such a way so as to verify the user requirements.

The description of the system architecture follows, including both structural and behavioral analysis of the system. A system architecture can be analyzed further into several subsystems until we reach the step that we define in detail the system components. Components compose the last level in the hierarchical decomposition of a system. Usually, the levels of decomposition that are used during the development of a system is a designer's choice and is affected by many factors like system complexity, COTS already available for a bottom up approach and integration facilitation. However, the components should be neither oversimplified nor difficult to analyze and develop.

The next step as we go down to the structure of the V-model is the development of the system components. After this step the right hand side of the V-model starts. In all those stages mainly the products of each level are integrated together, while simultaneously the result is checked against the specified requirements. The end result is an operational system that meets all the stakeholder needs.

1.2.1.2 Document-centric Systems Engineering

Till some years ago Systems Engineering was document-centric. In this approach, documents are the basic product of each process and information about requirements, system design, system analysis is captured in documents in either textual or electronic form [1]. Models are developed for different purposes like reliability analysis, design optimization but there were neither connected in a coherent way nor supported all lifecycle phases. This document-centric method has generic inefficiencies. Traceability among requirement and system design documents in very difficult. As a consequence changes in the documentation of the system can cause errors due to inconsistencies. In addition, maintenance and reusability of system requirements or design information is also difficult and error prone. Communication among teams taking part into system development that is based on documents is slow and requires a lot of effort to avoid misunderstandings and achieve parallel development. To deal with those inefficiencies and address better the challenges of today there is a turn towards what is called Model-Based Systems Engineering, which is analyzed in the next section.

1.2.2 Model-Based Systems Engineering (MBSE)

"Model-Based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning with the conceptual design phase and continuing throughout development and later life cycle phases"[5]. In contrast with the document-centric approach, here the models are the main product of each process and are used for the communication between the different teams that take part in the development. In the MBSE environment the information contained in the models should be consistent through all the phases. Moreover, models need to be developed carefully in order to meet their purpose. They need to be accurate enough but simultaneously avoid containing unnecessary details that add to the complexity.

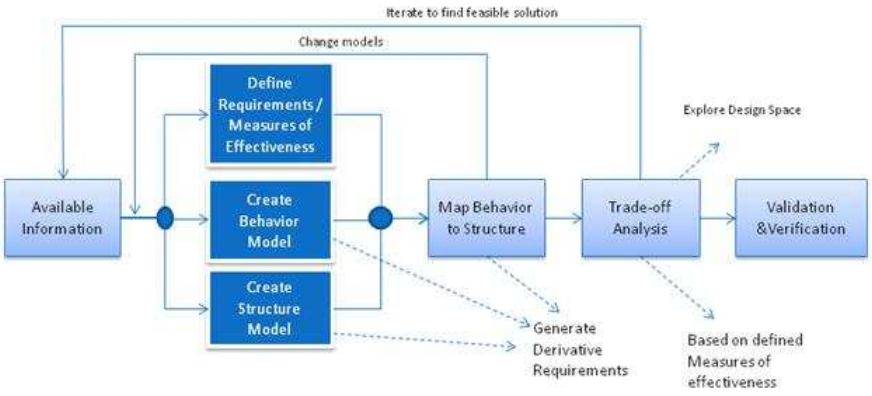

Figure 1.3: The MBSE Process

In Figure 1.3 the core steps of the MBSE process are illustrated. For each system the starting point is the available information. Afterwards, the initial system requirements are defined together with the desired measures of effectiveness (MoE).

The MoE are used afterwards at the trade-off stage as a criterion for the selection of the best system configuration. After the requirements phase, the models of behavior and structure are developed. Then the process continues by mapping the specified system behavior to the structure. During the MBSE process derivative requirements are generated and thus, if needed, changes to the system specification are performed. As it was explained also before, the trade off analysis phase is used to choose the best feasible solution based on the specified measures of effectiveness. After exploring the design space and selecting the best alternative the system shall be validated and verified. This phase is crucial because it makes sure that all the requirements are satisfied and that the system meets its goals.

1.2.3 How MBSE Address Today's Challenges?

As it was mentioned in section 1.1 several factors have increased greatly the complexity of systems. Systems development is an orders of magnitude harder process today than one decade ago, takes more time and is more prone to errors. Therefore, there is a need to rethink the way we develop systems. Model-Based Systems Engineering is a promising approach to this crucial for the economy and the society problem. While keeping all the advantages of Systems Engineering that help in reducing complexity and better manage a system development project, MBSE offers even more capabilities.

The greatest advantage of MBSE is that separates design from production. That is a major change to the existing status in systems development. Each com-

pany could have an “artificial” product made by using only models and then let other companies compete for the actual manufacturing/production. That will consequently drive costs down and possible reduce time to market. The same method was used in VLSI with phenomenal success.

The use of models allows for faster and more rigorous communication between engaged teams and stakeholders. Time overhead to manage system documentation will be extinct. Fewer errors will come up due to misunderstandings or oversights, mostly because of the more formal semantics that models offer.

Another characteristic of MBSE is the re-usability of models. A subsystem or a component model can be used in many systems without the need to be developed from scratch every time. Time to market as well as the probability of errors will be decreased, since many parts of the system model will be already completed and validated. Finally, all the above can lead in achieving a better overall quality for the system.

Chapter 2

The Modeling "Hub"

In section 1.2.2 of the first chapter the Model-Based Systems Engineering approach to system development was analyzed. Models play a key role in this approach and are used at every stage of development. A challenge in MBSE is to have models that are consistent with each other. However, besides having consistent data there is a need for the models to work together in order to offer a holistic Systems Engineering approach to the designer. In this work the concept of a modeling "hub" is introduced to deal with this challenge.

2.1 Hub Architecture

As is depicted in Figure 2.1 SysML is in the core of the modeling "hub", and is used for the high level design of the system; e.g. by providing annotated block diagrams of the structure and behavior of complex systems. The main aim is to integrate this core module with external multi-domain and multidisciplinary tools, each one used in a different phase of the Systems Engineering development process [4]. To achieve the integration a three layer approach needs to be followed. Initially, for the tool we need to integrate, a domain specific profile is implemented in SysML. Then a model transformation is defined, followed by the implementation of tool adapters that are used as a middleware for exchanging information between

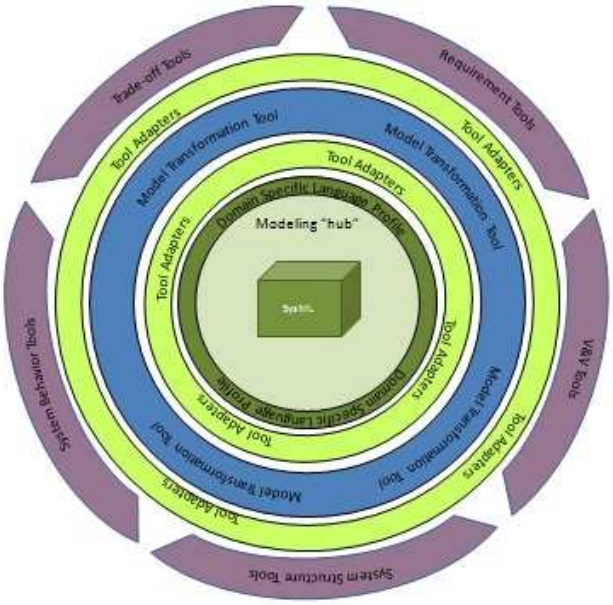

Figure 2.1: The Modeling Hub

the model transformation layer and the other components of the hub. The figure above presents these layers as well as the areas for which we need to integrate tools with the core module in order to realize the MBSE vision of offering a holistic system design experience.

2.2 SysML

SysML is a general purpose modeling language that was developed based on UML and aims to support the MBSE process by providing ways for the representation and analysis of complex systems. The structure in Figure 2.2 shows the different diagrams that are part of the SysML specification. Some of them are exactly the same as in UML, some other have differences like the activity and the internal block diagram, and other are completely new like the parametric and the requirement

diagram.

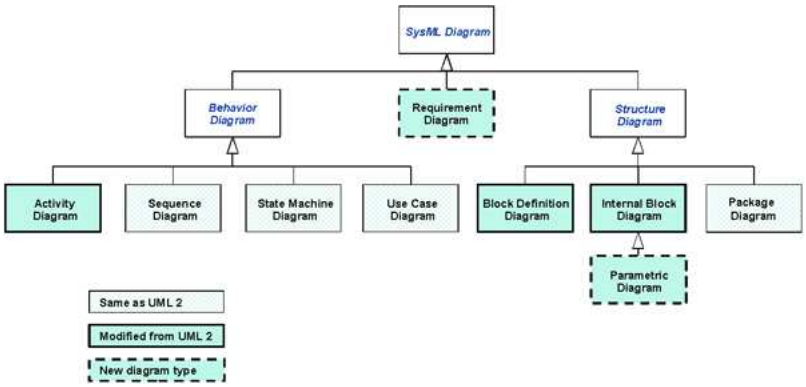

Figure 2.2: SysML Diagrams Taxonomy

SysML inserted the concept of requirement diagram in order to better represent and handle requirements. Apart from representing the requirements, the designer can specify their decomposition and also can allocate low level requirements to parts of structure. Consequently requirement diagram offers a better traceability among requirements and system components. Parametric diagrams are used to specify equations that characterize the system and link them to system component properties. The four main parts of a SysML model, which are called the pillars of SysML, are shown in Figure 2.3. According to these when a system is designed one should specify the requirements, the system behavior and structure as well as the parametrics of that system.

2.3 Trade-off Analysis and Design Space Exploration

In this work the focus is on integrating SysML with Consol-Optcad, a trade off analysis tool that was developed at the University of Maryland [6, 21].

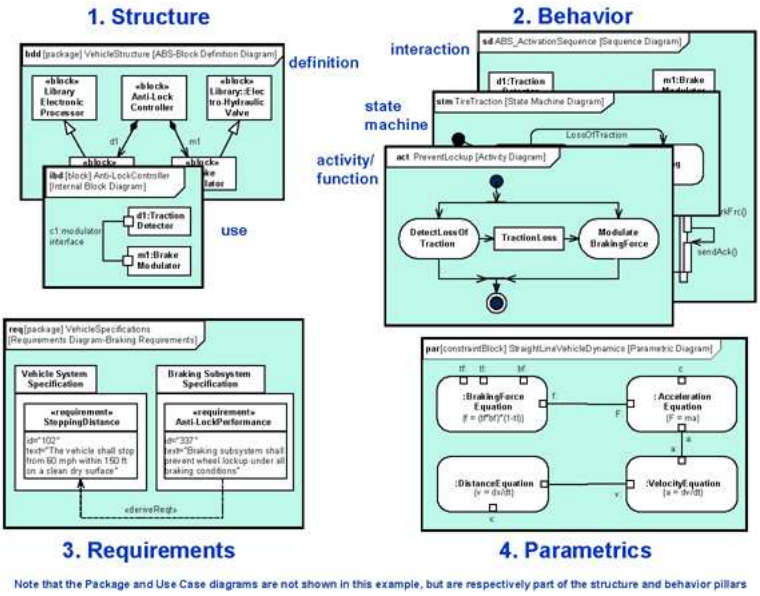

Figure 2.3: The Four Pillars of SysML

2.3.1 Why is it Important?

Trade-off is an essential part of the system design process, as it is a principal methodology for design space exploration. An integration of SysML with a trade off tool will allow the designer to make decisions faster and with more confidence. The probability of human error during transfer of data between tools will not exist and consequently the overall quality and performance of the system will be increased. This integration becomes even more crucial when we think of today's systems that have multiple competing objectives and requirements to satisfy and a lot of design parameters.

2.3.2 Consol-Optcad

Consol-Optcad is a multi-objective optimization tool that allows interaction between the model and the user. It can handle non-linear objective functions and constraints with continuous values. In systems development and after the system structure is defined there is a need to calculate the design parameters that best meet the objectives and constraints. Usually when we deal with complex systems and optimality is under consideration, this is not a trivial task. The support of an interactive tool, like Consol-Optcad, to help the designer resolve the emerging trade-offs is crucial. A major advantage of Consol-Optcad is that it allows the user to interact with the tool, while the optimization is under way. The designer might not know or might not be in position at the beginning to specify what he means by optimal design. Therefore such interaction with the tool could be of great benefit [6, 21]. Another key feature of Consol-Optcad is the use of Feasible Sequential Quadratic Programming (FSQP) algorithm for the solver. FSQP's advantage is that as soon as we get an iteration solution that is inside the feasible region, feasibility is guaranteed for the following iterations as well. Moreover, very interesting is the fact that besides traditional objectives and constraints Consol-Optcad allows the definition of functional constraints and objectives, that depend on a free parameter. Consol Optcad has been applied to the design of flight control systems [13], rotorcraft systems [14], circuits and others.

The structure of Consol-Optcad Software is depicted in Figure 2.4. The problem description file is used to specify the problem by using the specific commands

of Consol-Optcad. Convert function compiles that file and then Solve is used to perform the optimization. As it can be seen Solve can communicate with the user as well as with external simulators and gives as an output the optimal solution.

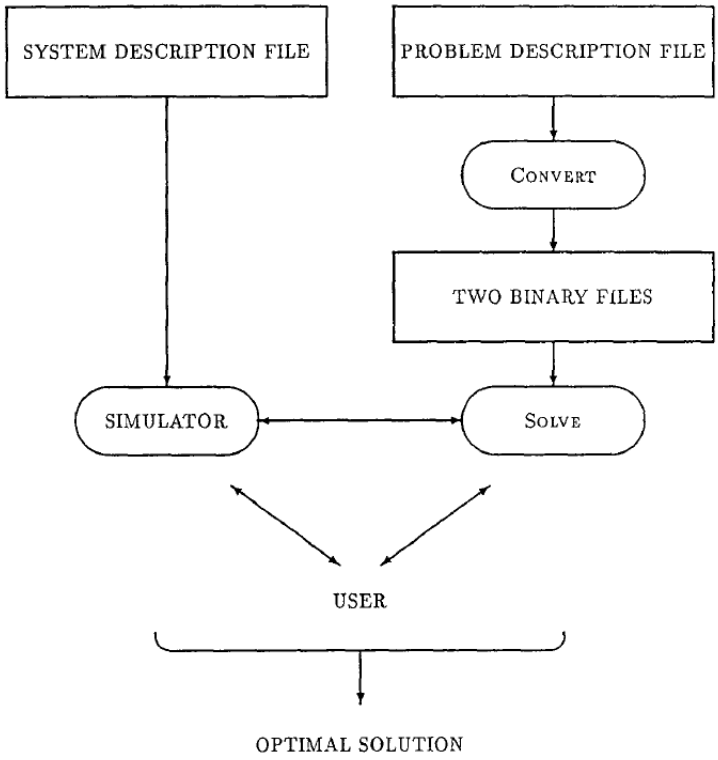

Figure 2.4: Consol-Optcad Structure

2.3.2.1 Consol-Optcad Constructs

At this section the different constructs of Consol-Optcad that represent commands that the user can specify in the tool environment, will be presented as they are defined in the user's guide [6].

Design Parameter

Represents a parameter that can be adjusted by the optimization algorithm. The form of the design parameter command is presented below:

design_parameter ⟨ identifier ⟩

init = ⟨ expression ⟩

variation = ⟨ expression ⟩

min = ⟨ soft or hard expression ⟩

max = ⟨ soft or hard expression ⟩

Init attribute is the initial value of the parameter, while min and max, if specified, they provide respectively low and upper bounds for the parameter. Variation is a quantity indicating the degree of confidence in the initial guess that the designer provided through init. According to the Consol-Optcad user's guide variation should be the difference of the initial value and the next value the designer would choose if he had to proceed by hand. The variation is used to scale the initial parameter space according to the formula:

$$scaled\_value = \frac{raw\_value}{variation} \tag{2.1}$$

If the choice of variation is improper the initial progress of the optimization is slower. The default value is one.

Objective

The objective declares an objective function and has the following form:

objective ⟨ quoted string ⟩

⟨ optimize ⟩ ⟨ pseudo-C code ⟩

good_value = ⟨ expression ⟩

bad_value = ⟨ expression ⟩

The objective can be either to minimize or to maximize the function. This is specified at the ⟨ optimize ⟩ part of the command. The ⟨ pseudo-C code ⟩ is C-code the returned value of which is the value of the objective function. Good_value represents our desired value for the objective function. Respectively a bad_value is used to describe an unfavorable design solution for the objective function. Another point to illustrate is that when the objective is to be minimized (maximized) the good_value must be smaller (larger) than the bad_value.

Constraint

A constraint in Consol-Optcad has the following form:

constraint ⟨ quoted string ⟩ ⟨ soft or hard ⟩

⟨ pseudo-C code ⟩ ⟨ inequality ⟩

good_value = ⟨ expression ⟩

bad_value = ⟨ expression ⟩

The structure of constraint is similar to the objective function and only two differences exist. First, a constraint can be characterized as soft or as hard in the ⟨ soft or hard ⟩ part of the command. Consol-Optcad treats soft constraints similar to objective functions. It tries to minimize/maximize the value of the constraint in order to reach as close as possible to the good value. If the constraint is hard the value of that constraint shall meet the good value for

the problem to have a solution. The second difference is that for the constraint the ⟨ inequality ⟩ should also be defined. Again if ⟨ inequality ⟩ is equal to <= (>=) the good_value must be smaller (larger) than the bad_value.

Functional Objective

A functional objective has the following form:

functional_objective ⟨ quoted string ⟩

for ⟨ identifier ⟩ from ⟨ expr ⟩ to ⟨ expr ⟩ ⟨ mesh type ⟩ ⟨ expr ⟩

⟨ optimize ⟩ ⟨ pseudo-C code ⟩

good_curve = ⟨ pseudo-C code ⟩

bad_curve = ⟨ pseudo-C code ⟩

A functional objective is similar to an objective. The difference is that the functional objective depends on a free parameter, like time, frequency and therefore the objective should be minimized or maximized for all the values of that parameter. For that reason, there is a need to define good and bad curves and not just values. The second row of the command is responsible for specifying the limits of the free parameter and how the range of the values is going to be divided. The syntax is similar to a for-command in C.

Functional Constraint

A functional constraint has the following form:

functional_constraint ⟨ quoted string ⟩ ⟨ soft or hard ⟩

for ⟨ identifier ⟩ from ⟨ expr ⟩ to ⟨ expr ⟩ ⟨ mesh type ⟩ ⟨ expr ⟩

⟨ pseudo-C code ⟩ ⟨ inequality ⟩

good_curve = ⟨ pseudo-C code ⟩

bad_curve = ⟨ pseudo-C code ⟩

The differences between an objective and a functional objective described above, are valid as well for the case of a functional constraint and a constraint.

2.3.2.2 Phases of the Optimization

In the previous section the main constructs, that the designer can use to specify an optimization problem in Consol-Optcad, were discussed. One of the key concepts that Consol-Optcad introduces is that of specifying good and bad values for objectives and soft constraints. These values are used by the optimizer in order to treat all objectives and soft constraints equally. Having any of the various objectives or soft constraints achieve their corresponding good value should provide the same level of satisfaction to the designer. Respectively achieving a bad value should give the designer the same level of dissatisfaction for any objective or soft constraint. The formula below is used by Consol-Optcad to normalize the value of any objective or soft constraint according to their good and bad value:

$$scaled\_value = \frac{raw\_value - good\_value}{bad\_value - good\_value} \tag{2.2}$$

As it can be determined from the formula above a good value corresponds to a scaled value of 0 whereas a bad value to a scaled value of 1. Apart from objectives and soft constraints also hard constraints are assigned good and bad values, but those values are used only in the first phase of the optimization that is going to be described below.

According to M.Fan et al in [6] a problem (P) in Consol-Optcad takes the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & obj_k(x) \ \ \forall k \\ \text{subject to} \quad & soft_i(x) \leq 0 \ \ \forall i \\ & hard_j(x) \leq 0 \ \ \forall j \\ & hard\_bound_l(x) \leq 0 \ \ \forall l \end{aligned}$$

,where obj_k, $soft_i$, $hard_j$ are scaled values of objectives, soft and hard constraints respectively and $hard\_bound_l$ represent hard bound constraints on design parameters.

The optimization process has three phases according to feasibility or infeasibility of x with respect to soft and hard constraints. The three phases are listed below:

- Phase 1: In this phase the design parameters are adjusted to ensure that hard constraints are satisfied. Problem (P) takes the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \ \max_j \quad & hard_j(x) \\ \text{subject to} \quad & hard\_bound_l(x) \leq 0 \ \ \forall l \end{aligned}$$

  Once all hard constraints are satisfied the optimization process moves to Phase two.

- Phase 2: All the hard constraints are satisfied when the optimization reaches this stage. In this phase the main emphasis is on satisfying the soft constraints

and problem (P) takes the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \ \max_{k,i} \quad & \{obj_k(x), soft_i(x)\} \\ \text{subject to} \quad & hard_j(x) \leq 0 \ \ \forall j \\ & hard\_bound_l(x) \leq 0 \ \ \forall l \end{aligned}$$

When all soft constraints satisfy their good values we move to phase 3. However, we can have also a feasible design when all soft constraints meet at least their bad value threshold.

- Phase 3: Since all hard and soft constraints are satisfied when we reach phase 3, all iterations in this phase give feasible design solutions. The main focus is on minimizing at each step the objective that has the worst value. The form of the problem (P) is presented below:

$$\begin{aligned} \underset{x}{\text{minimize}} \ \max_{k} \quad & obj_k(x) \ \ \forall k \\ \text{subject to} \quad & soft_i(x) \leq 0 \ \ \forall i \\ & hard_j(x) \leq 0 \ \ \forall j \\ & hard\_bound_l(x) \leq 0 \ \ \forall l \end{aligned}$$

As it has been mentioned before a key concept of Consol-Optcad is the interaction with the designer. This interaction will take place mostly on phase 2 and 3 of the optimization process. A last thing to underline is that in the phases described above functional constraints and objectives were omitted, to make the optimization steps easier to explain. The only thing that changes however is the way that scaled values are computed for functional constraints and objectives. The following

formula is used for the normalization of functional objectives and constraints:

$$scaled\_value(\omega) = \frac{raw\_value(\omega) - good\_curve(\omega)}{bad\_curve(\omega) - good\_curve(\omega)} \tag{2.3}$$

Chapter 3

SysML Integration with Consol-Optcad

As it was mentioned in section 2.3 in this work we give emphasis on the integration between SysML and Consol-Optcad. This section describes the integration framework and the separate steps that were followed to achieve this integration. Here is a good point to mention that MagicDraw was the SysML tool that was used for this integration. SysML is not a tool specific language, but MagicDraw was used because it is more open than other tools and it can be modified more easily. To simplify things from now on when integration with SysML is mentioned, we refer to integration with MagicDraw SysML. This chapter will analyze in detail the integration framework and the separate steps that were followed to achieve the integration.

3.1 Integration Framework

Figure 3.1 presents the architecture of the integration together with numbered steps that need to be followed to complete the integration process. According to the three layer approach presented in section 2.1, the integration process is divided in three main parts. The first part, concerns the mapping of the objects between the two languages (SysML, Consol-Optcad). It also includes the development of specific semantics that are going to be used for that purpose and in this case a profile of

Consol-Optcad in SysML is created. The second part, is the metamodeling layer where the transformation between the two models takes place. The last part consists of implementing the appropriate tool adapters.

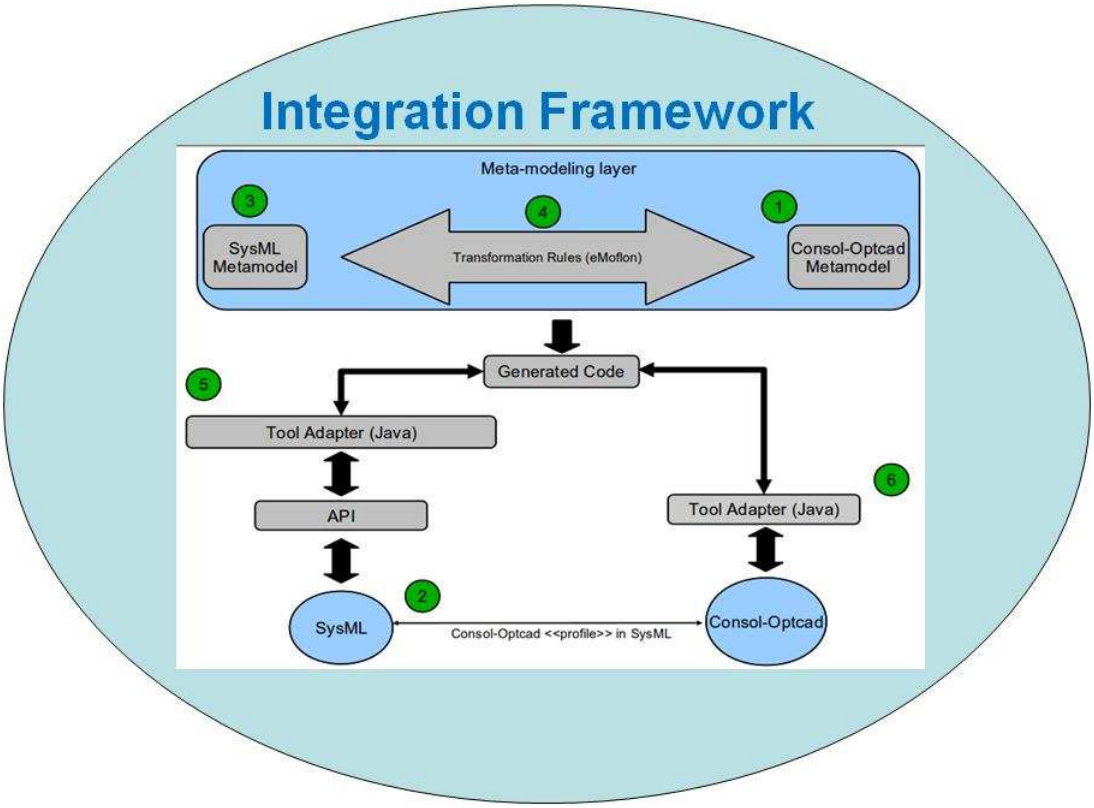

Figure 3.1: Integration Framework

3.2 Consol-Optcad Profile in SysML

The creation of a Consol-Optcad profile is the first step of the integration process. Profiling is the mechanism that SysML has in order to allow the designer to use additional constructs inside the development environment. MagicDraw tool support that function. After a profile is built and since it gives the user the ability

to use constructs of a specific tool directly in SysML, it decreases significantly the design effort.

3.2.1 Mapping of Constructs between SysML and Consol-Optcad

Before implementing the actual profile in MagicDraw there is a need to match conceptually the constructs of the two languages. The difficulty of the mapping process is case dependent. For this work there were not many alternatives and thus each construct of Consol-Optcad is mapped to a SysML Block. Each SysML Block has an applied stereotype so that the different constructs of Consol-Optcad can be distinguished. In the table below the aforementioned mappings are presented. Moreover, the specific constructs of Consol Optcad that are mentioned here were analyzed in section 2.3.2.1.

Mapping Table	
SysML Constructs	**Consol Optcad Constructs**
SysML BLock with applied stereotype	Design Parameters
SysML BLock with applied stereotype	Objective
SysML BLock with applied stereotype	Constraint
SysML BLock with applied stereotype	Problem description file
SysML BLock with applied stereotype	Functional objective
SysML BLock with applied stereotype	Functional constraint

3.2.2 Building the Profile

A SysML Profile is composed by a set of stereotypes and their relationships. For each stereotype the designer should specify tag values that correspond to attributes of the construct that the stereotype describes. Those tag values should have a specific value type. The value type can be either a predefined one, like integer, string, real or a user defined value type. A detailed description of profile building process can be found in MagicDraw user's guide [7]. In Figure 3.2 the implemented Consol-Optcad profile is depicted. Each Consol-Optcad construct of the mapping table is represented in this diagram by a stereotype. The tag values for each construct were defined according to the Consol-Optcad specification document [6]. Moreover, four new value types were used in this specification.

Until now the process of building the profile is tool independent. That means that a profile can be build by the same way in any other tool that supports SysML. However in MagicDraw two more steps are needed so that the profile can be usable. First, a so called customization diagram must be developed. A customization diagram (Figure 3.3) has one customization class for each stereotype of the profile. Inside a customization class a user can define rules about this construct and also specify the value of a set of tags. The only two tags that need to be always set to a value are the customizationTarget tag and the hideMetatype tag. A detailed information about all the different tags and their functionality can again be found on MagicDraw user's guide [7].

The last step in order to get a working profile in SysML is to modify the

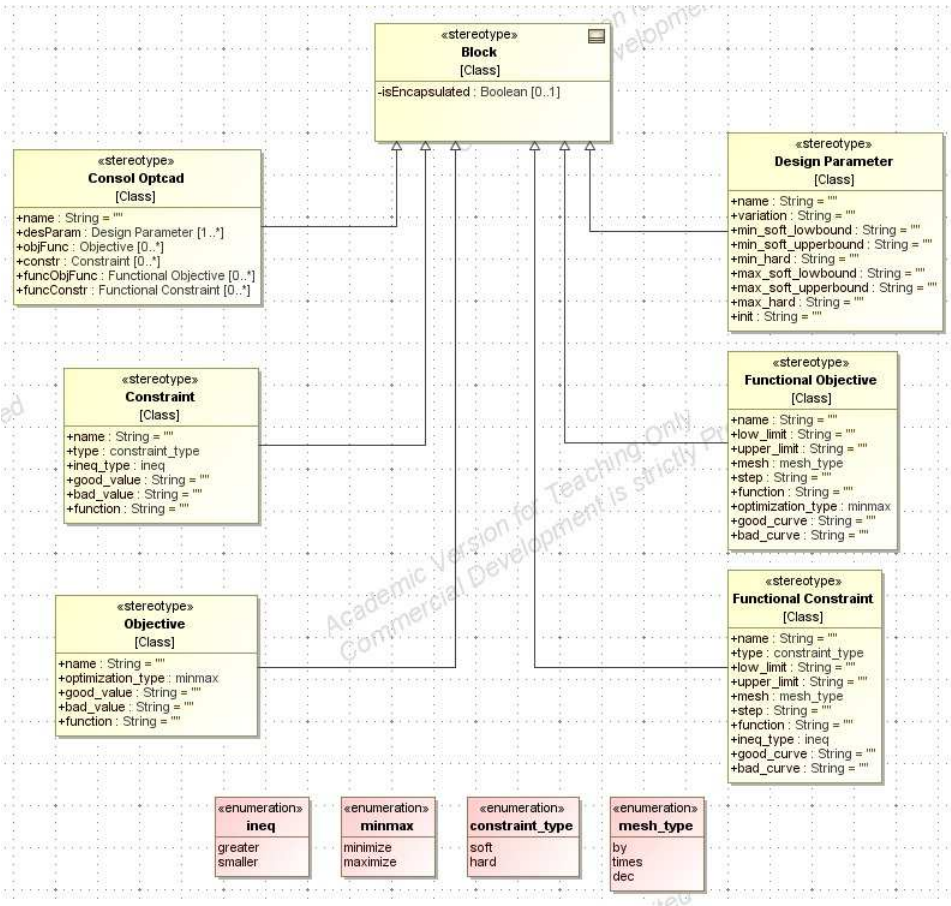

Figure 3.2: Consol-Optcad Profile in SysML

block definition diagram so as to have the new constructs on a special section of the panel. Then a designer can load the new profile inside a project and start using it by simply drag and drop Consol-Optcad constructs in the block definition diagram area. Figure 3.4 presents a snapshot of the new block definition diagram panel.

3.3 Metamodeling Layer

The metamodeling layer is the second major part of the integration process. A metamodeling layer stands one abstraction layer above the actual design imple-

Figure 3.3: SysML Customization Diagram

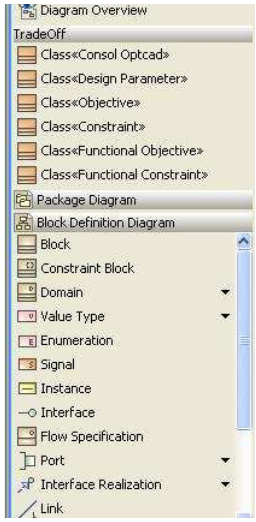

Figure 3.4: Modified Block Definition Diagram Panel

mentation in a modeling language. A metamodel consists of the constructs of a modeling language together with the rules that specify the allowable relationships between these constructs. It can be resembled as the grammar of the modeling

language.

3.3.1 Model Transformation

At the metamodeling layer model transformations take place. In general a model transformation is the conversion of a metamodel of one language to the metamodel of another. There are many alternatives in terms of model transformation tools, like ATL, GME, eMoflon, QVT. In this research the eMoflon model transformation tool was used which was developed at TU Darmstadt [9] [10]. In the sections that follow the architecture of eMoflon tool as well as the actual model transformation between SysML and Consol-Optcad are described in more detail.

3.3.1.1 eMoflon Toolsuit

The eMoflon tool was chosen instead of other metamodeling tools due to the following reasons. First of all, graph transformations is the underlying theory for the model transformations; a fact that makes the semantics strong and can lead to satisfaction of formal properties such as correctness, completeness and termination. In addition, the way eMoflon toolsuit is developed allows for graphical representation of metamodels and transformation rules; a fact that makes the model transformation process less cumbersome and less error prone. Another advantage of eMoflon is that it can generate automatically Java code for the model transformations. Since most of the tools today are implemented in Java, eMoflon code for transformations can be easily used and integrated in other tools. Finally, the development environment

is user friendly, the tool is well documented and it has a strong support/developing team.

In Figure 3.5 the architecture of the eMoflon tool is presented at a high abstraction layer.

Figure 3.5: eMoflon Architecture

A more detailed system architecture of eMoflon is presented in Figure 3.6, which comes from a recent paper of the eMoflon developing team [10].

As it can be seen the front end of the eMoflon tool is realized in Enterprise Architect (EA) through a special EA Add-in. Enterprise Architect is an industry strong Computer Aided Software Engineering (CASE) tool that is used at many companies. The reasons for choosing EA for the front end part of the system, instead of some other CASE tool, are analyzed by eMoflon team in [10]. The front end component of the system architecture, should provide the user the capabilities for specifying metamodels by using class diagrams, unidirectional model transformations via Story Diagrams SDMs (see section 3.3.1.4) and bidirectional transformations via triple graph grammars (TGGs). Moreover, the front end component should have an interface that will allow the stored data to be exported to the Integrated Development Environment (IDE) component in the correct form. The developed EA Add-in meets those requirements apart from the capability of specifying bidirec-

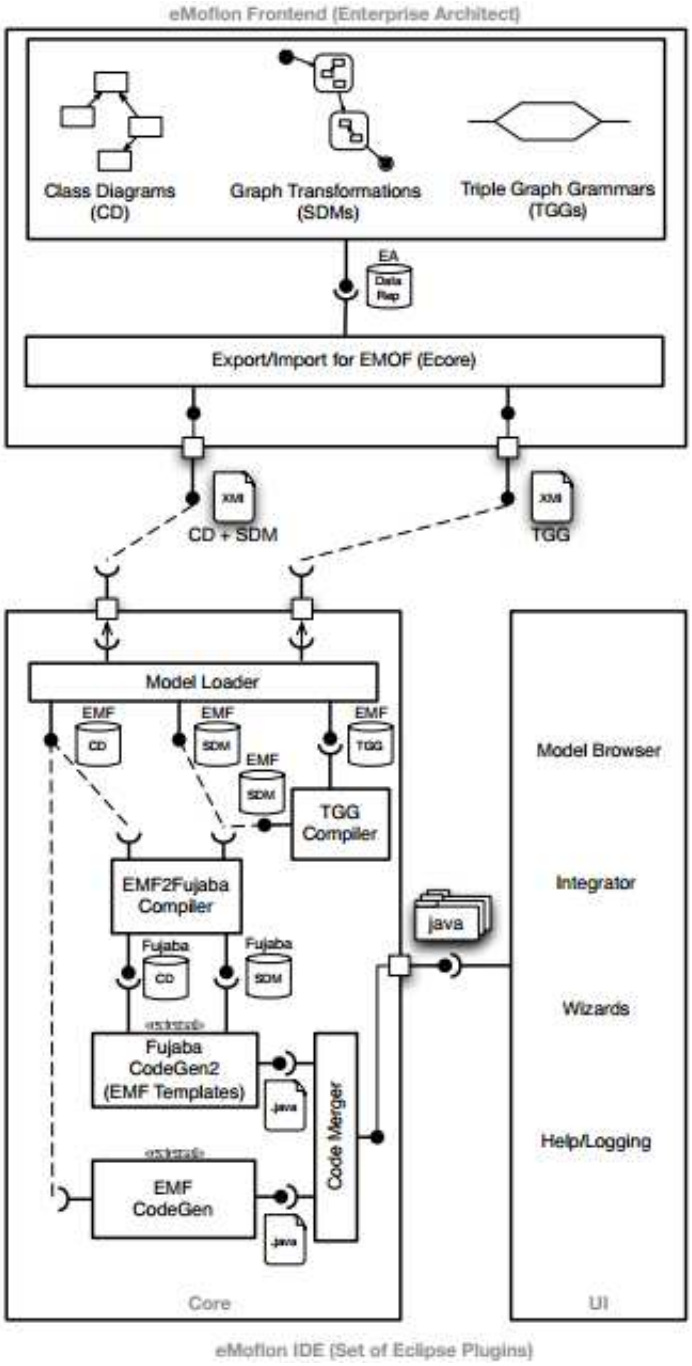

Figure 3.6: eMoflon Detailed System Architecture

tional transformations using TGGs. According to the eMoflon team this capability will be available soon. Here is a good point to emphasize that eMoflon does not

depend on EA or any other editor. That independence allows eMoflon to be used as a command line operated code generator and compiler fact that is useful when trying to automate a process. In addition the eMoflon team is developing a textual front end for the system.

Another thing to mention about the front end component is that the eMoflon tool requires all metamodels and other diagrams to be developed according to the Ecore format. With the increasing popularity of Eclipse and Eclipse Modeling Framework (EMF), Ecore established as a standard for metamodeling. Ecore is also isomorphic to a subset of MOF, named EMOF, that is an OMG standard. Ecore/EMF for the expression of the metamodel allows objects, associations between them, operations inside each project, attributes and also some simple constraints on objects, like multiplicity and relationship. The notation that is used to build the metamodels is similar to a UML class diagram. Figure 3.7 presents all the constructs that can be used to develop a metamodel that is compatible with the Ecore format.

The second major component of the eMoflon system architecture is the IDE, that is realized by a set of eclipse plug-ins. The CORE set of plug-ins is responsible for code generation. Through the Model Loader the information from the front end is stored in EMF format. A TGG compiler converts the TGG model to a set of unidirectional SDMs. Then all models are compiled through the EMF2FUJABA compiler to a format understandable by FUJABA. FUJABA (Universität Paderborn) is a model transformation engine that generates java code for unidirectional transformations expressed with SDMs. The EMF compliant java code from the

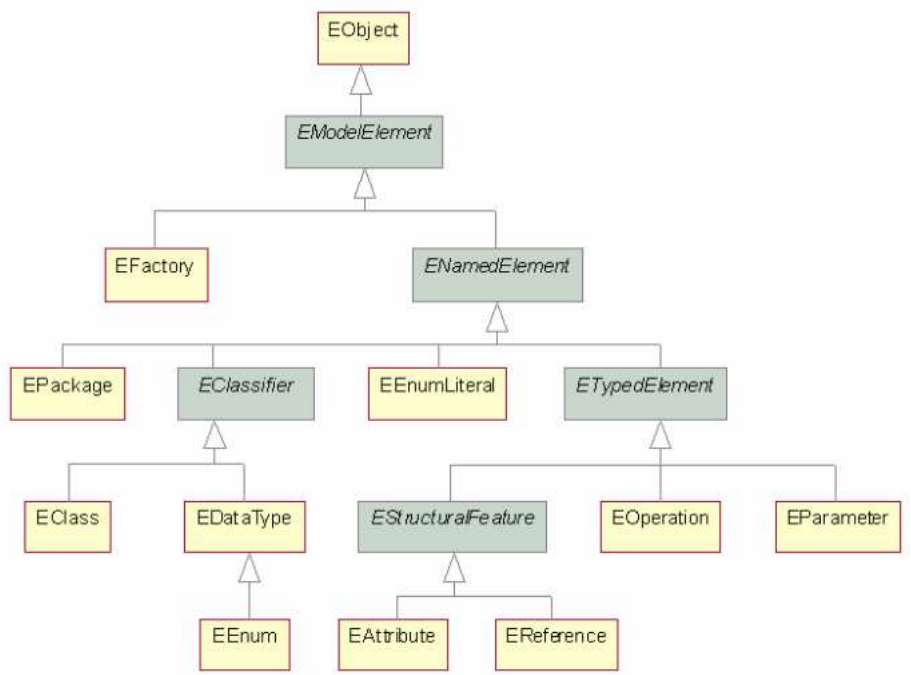

Figure 3.7: Ecore Metamodel

SDMs and the metamodels is then merged and forward to the UI set of plug-ins. The UI is responsible for allowing the user to visualize, navigate through and manipulate the generated code.

3.3.1.2 SysML/UML Metamodel

In Figure 3.8 the constructs from SysML and UML that can be used for model transformations are presented. Those constructs are a subset of the two metamodels. Actually the way eMoflon is implemented you need to develop a metamodel with only the constructs that are used in the transformation rules. The files that contain the whole metamodels in Ecore format need to be imported in eclipse for the transformations to take place. Until now it is not possible in eMoflon to input

large external metamodel files. Because of that, as it was mentioned above, we need to rebuild part of the metamodel, even though we might have the whole metamodel structure in a separate file.

3.3.1.3 Consol-Optcad Metamodel

Like with the SysML/UML metamodel, the Consol-Optcad metamodel was developed in Ecore format (Figure 3.9).

3.3.1.4 Transformation Rules and Code Generation

As it was mentioned above the transformation rules are expressed using Story Diagrams (SDMs). Story diagrams is a graph grammar language and they provide a mechanism for defining unidirectional graph transformations. SDMs adopt concepts from UML class, activity and collaboration diagrams [12].

As described in [9], a rule r:(L,R) defined by an SDM follows the three steps below when is applied to a graph:

- Find a match m for the precondition L in G
- Delete all the elements that are present in the precondition but not in the postcondition (Destroy := (L \ R)), to form (G \ Destroy)
- Create new elements that are present in the postcondition but not in the precondition (Create := (R \ L)), to form a new graph H = (G \ Destroy) ∪ Create

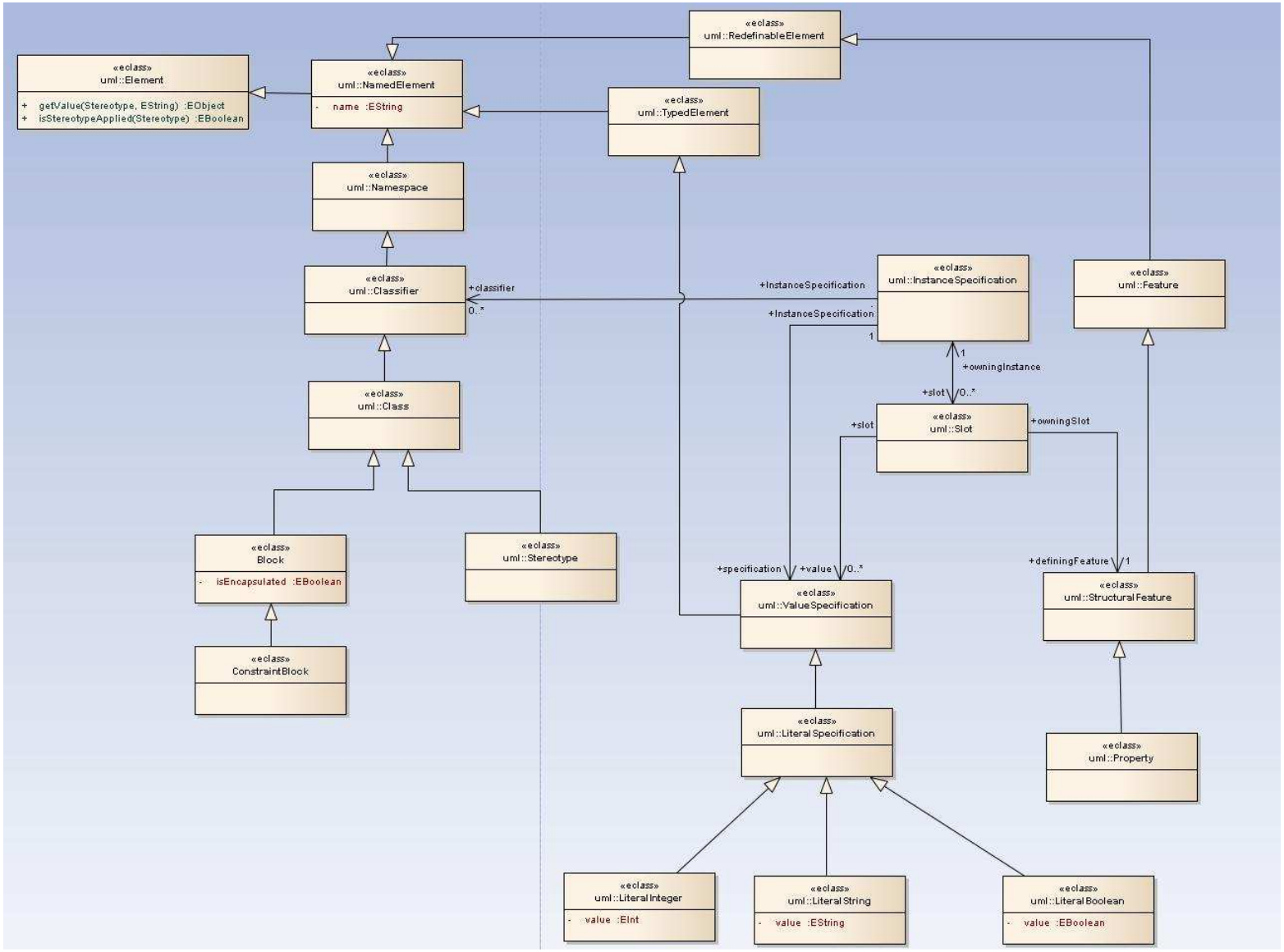

Figure 3.8: SysML/UML Metamodel

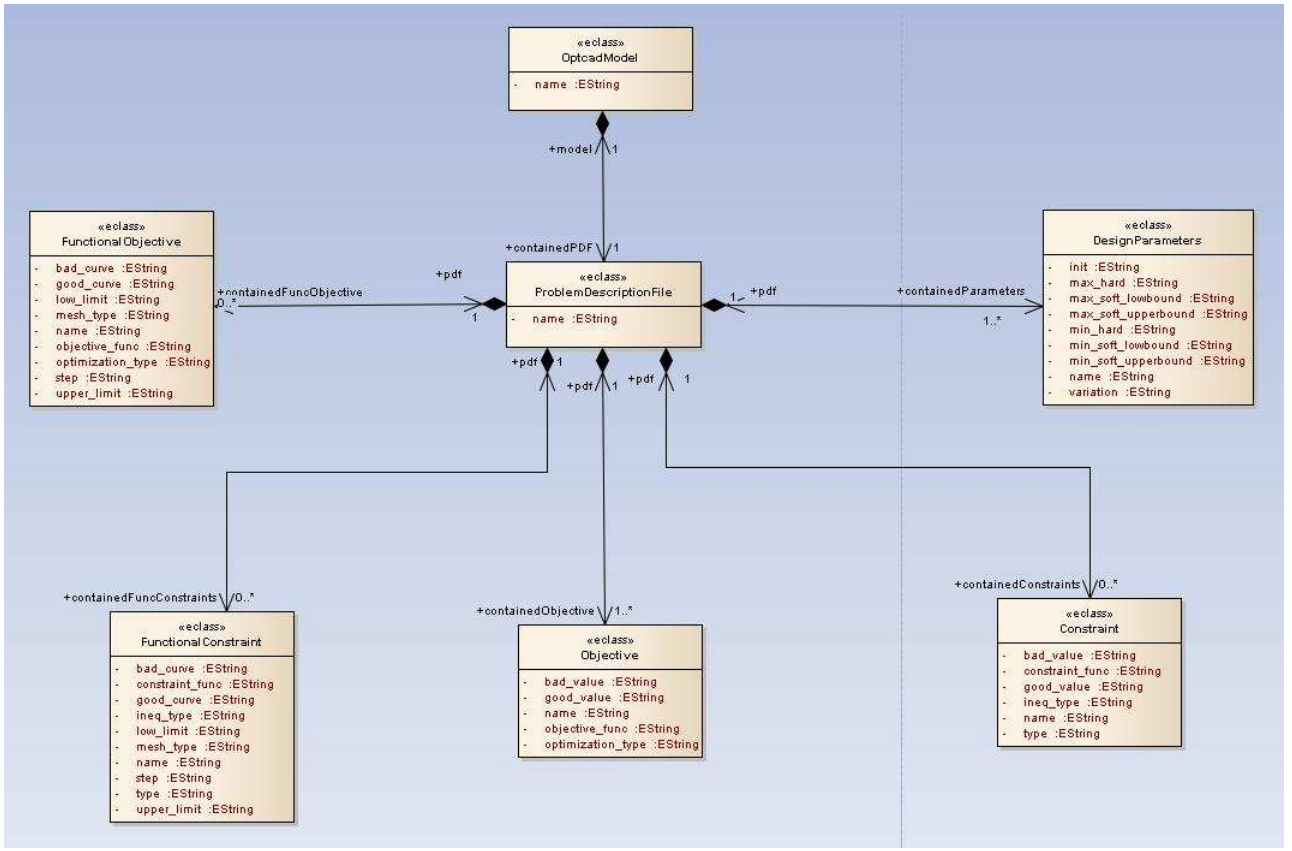

Figure 3.9: Consol-Optcad Metamodel

Figure 3.10 presents two graph transformations rules and Figures 3.11 and 3.12 depict how those rules can be applied on graphs. The nodes of these graphs can be thought as model blocks and the edges as associations between those blocks. Both examples represent rules that evolve the structure of only one graph. However, a rule can be defined so as the precondition (left part of the rule) and postcondition (right part of the rule) are applied to different graphs.

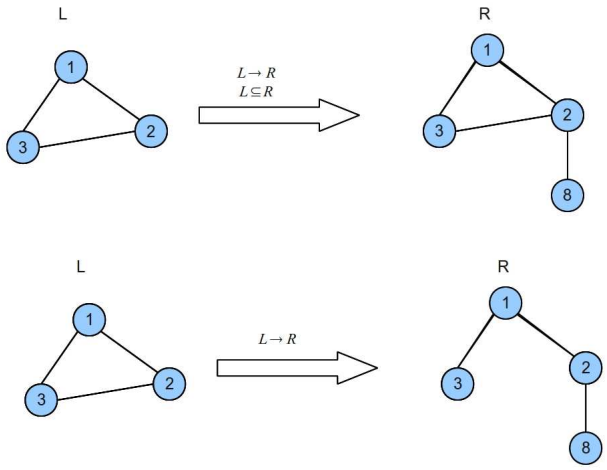

Figure 3.10: Graph Transformation Rules

At this point we focus on the actual eMoflon project for the integration of SysML with Consol-Optcad. The starting point is the definition of the metamodels in Enterprise Architect. Both metamodels were presented in sections 3.3.1.2 and 3.3.1.3. For the transformation to take place we need to define a new package that will contain an object responsible for the transformation. The rules for the model transformation are defined as operations inside this object (see Figure 3.13).

Each operation has an attached story diagram that specifies the rule. Below

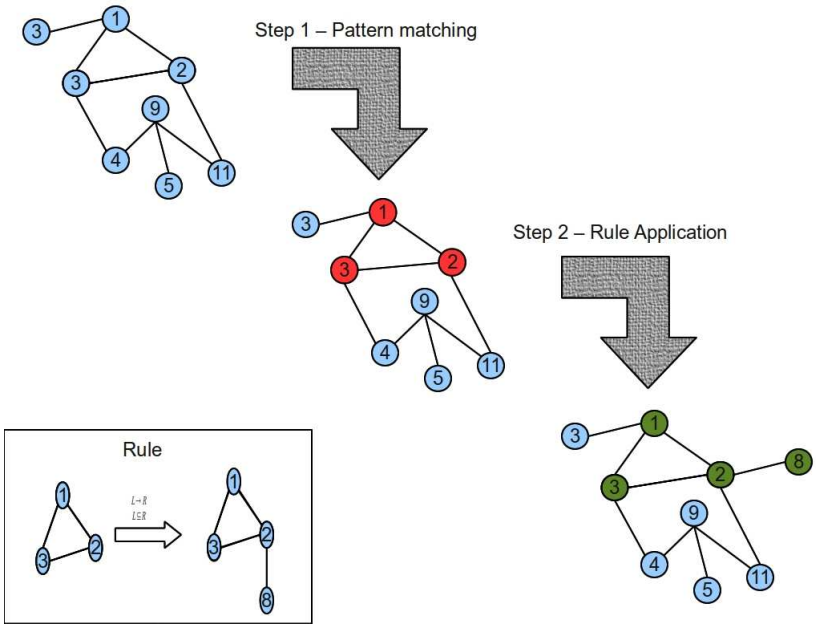

Figure 3.11: Graph Transformation Rule Application (1)

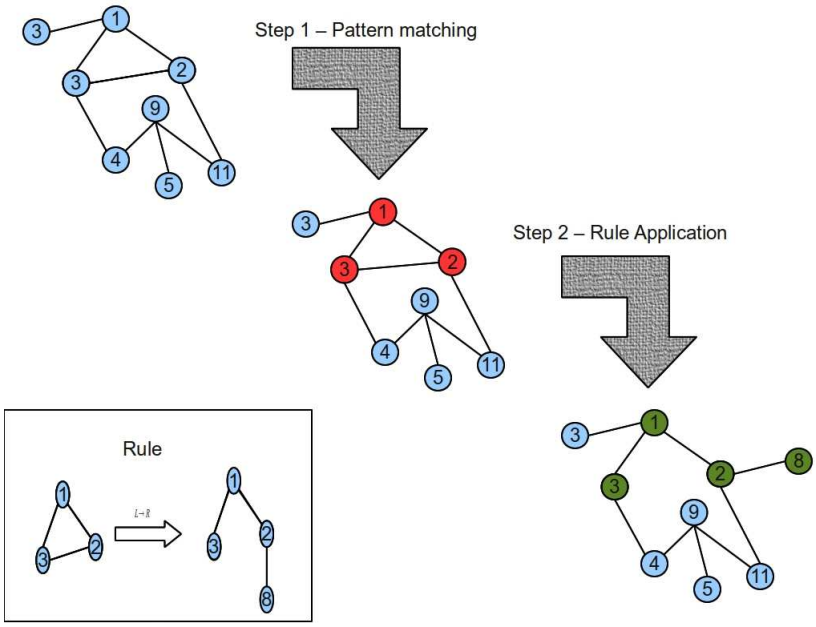

Figure 3.12: Graph Transformation Rule Application (2)

Figure 3.13: Object Responsible for the Transformations

Figure 3.14 corresponds to the rule that creates a functional constraint if a model already exists. As is going to be described in section 3.4.1 when a functional constraint block is identified by the developed plug-in the Java function that corresponds to that rule is called and a Consol-Optcad functional constraint construct is created. The end result, after applying all the rules for all the objects, is a Consol-Optcad model that corresponds to the initial SysML model.

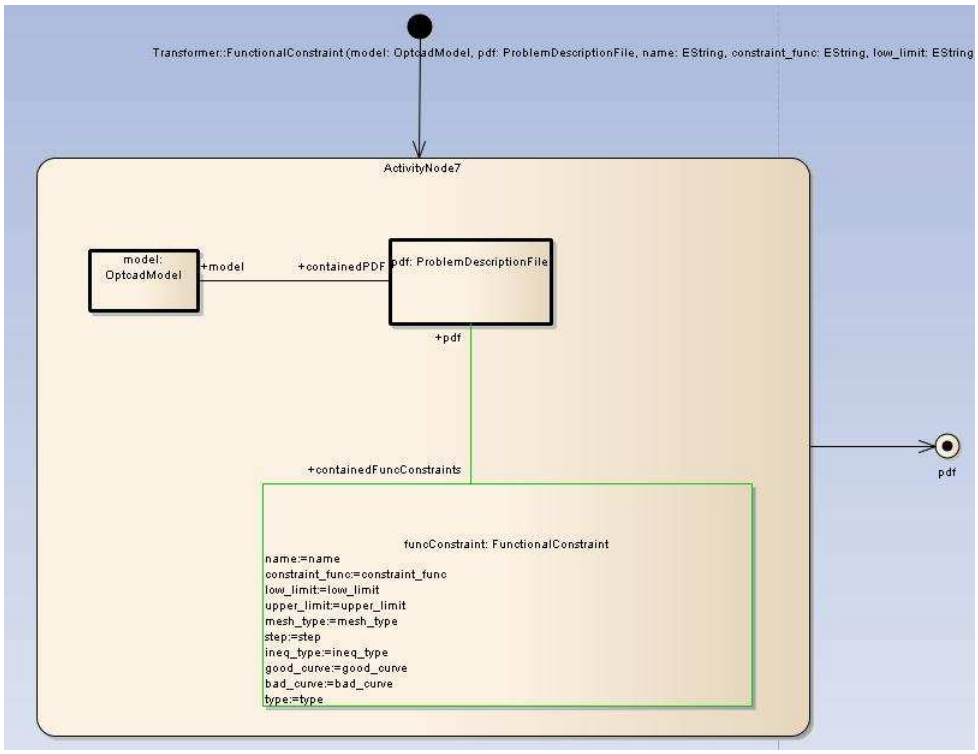

Figure 3.14: Rule for Functional Constraint

After defining all the SDMs, the EA models should be exported to eclipse. Figure 3.15 shows the project structure that we get in eclipse after the export function of eMoflon in EA is invoked.

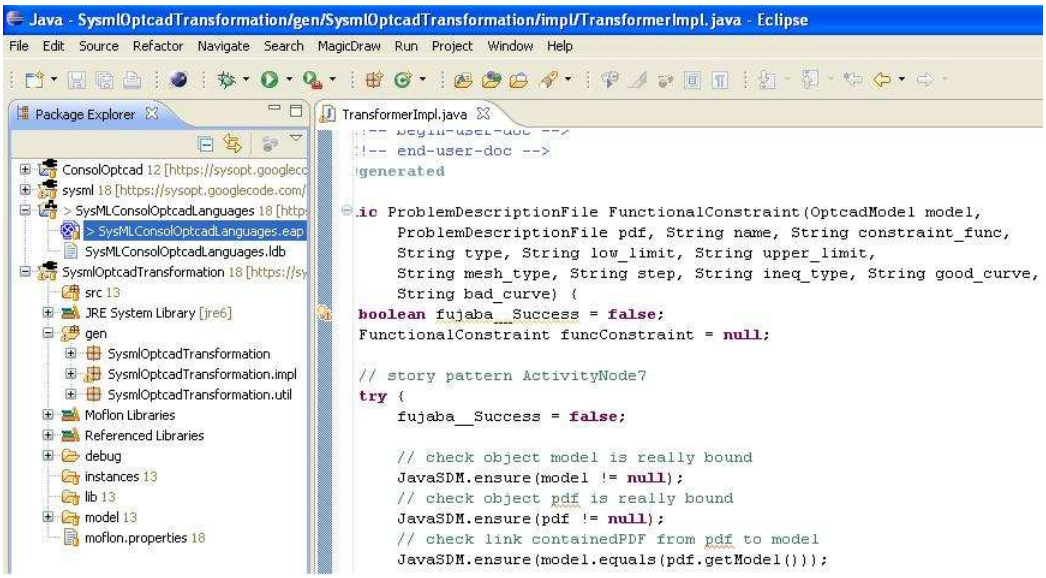

Figure 3.15: eMoflon Project Structure in Eclipse

3.4 Tool Adapters

Tool adapters work as the "glue" between the different pieces of software. Their role is to access/change information inside a model and also call the appropriate Java functions generated by the eMoflon tool to perform the model transformations. All adapters for this project were developed inside a single MagicDraw plug-in, which is the way to extend the functionality of MagicDraw. In the next section the developed MagicDraw plug-in will be analyzed in detail.

3.4.1 MagicDraw Plug-in

The plug-in is the core of the software part of the integration process. It was developed in the Eclipse platform using Java. Eclipse offers a way to develop a plug-in inside its environment and then start MagicDraw from Eclipse (see Figures

3.16 and 3.17). That makes the development process much easier since debugging information is immediately accessible through the Eclipse environment [8].

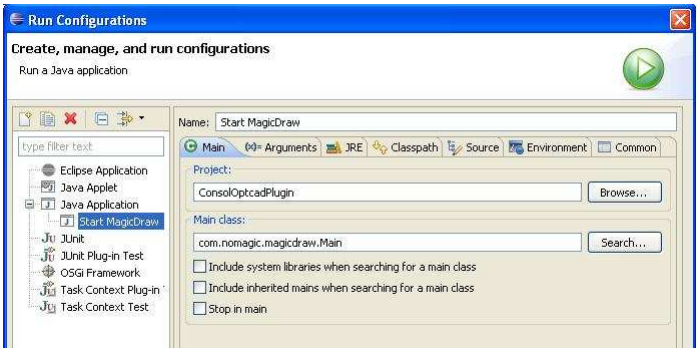

Figure 3.16: Starting MagicDraw from Eclipse Environment

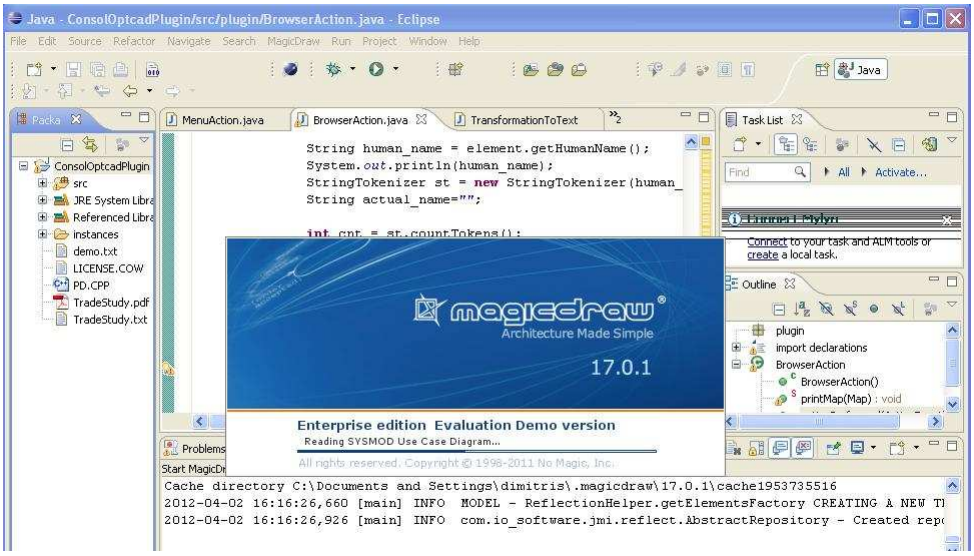

Figure 3.17: Starting MagicDraw from Eclipse Environment cont'd

Figure 3.18 shows the software architecture of the plug-in, which was obtained using ObjectAid UML tool.

To create a plug-in for Magicdraw we need to implement three classes. Myplugin, which inherits from the Plugin class, is the class that is called when MagicDraw

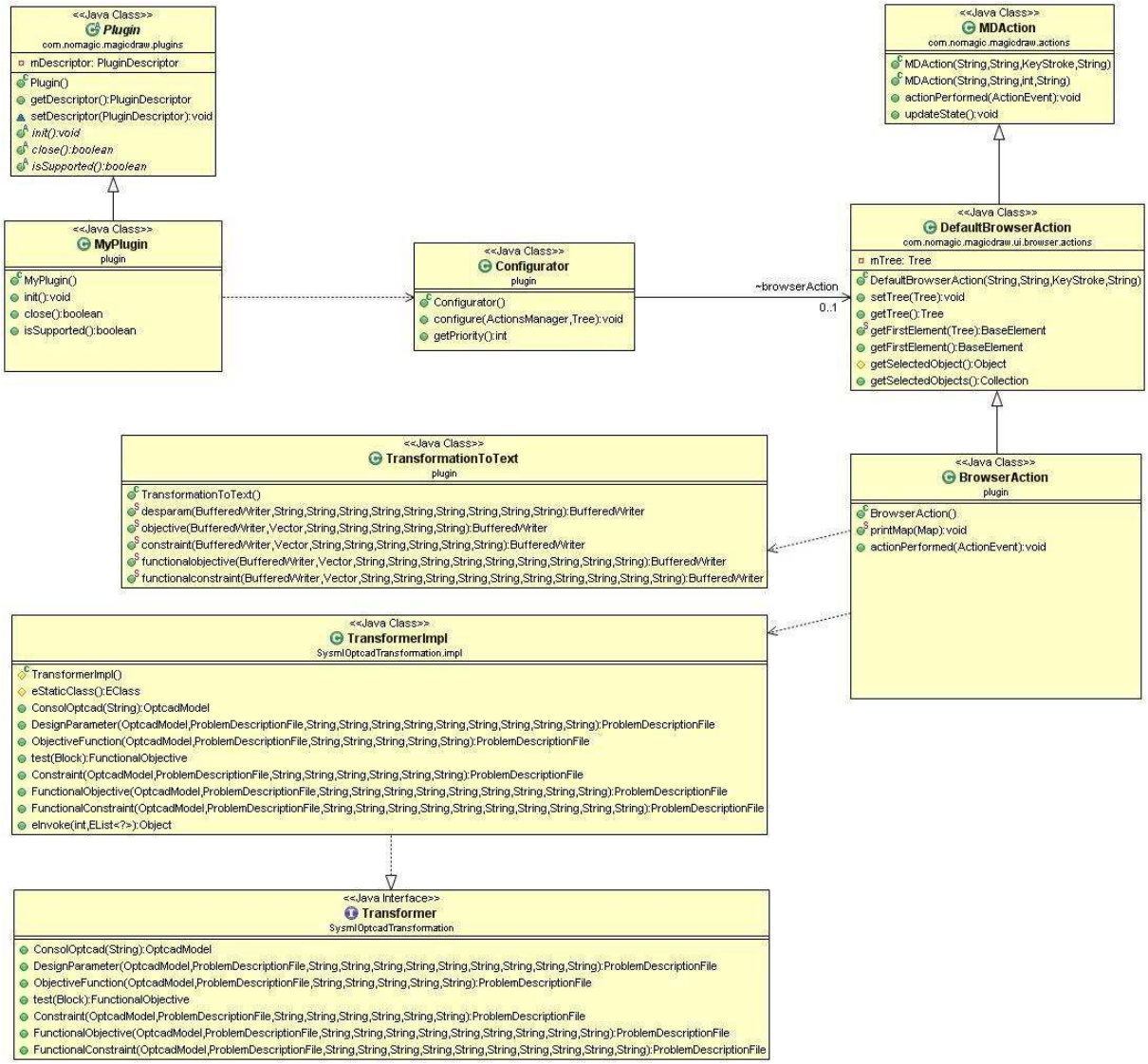

Figure 3.18: MagicDraw Plug-in Software Structure

starts and loads all the available plug-ins. This class is responsible to create a Configurator Class and attach it to the current plug-in. The Configurator class adjusts the User Interface (UI) of MagicDraw, to show the Consol-Optcad plug-in as a choice in the MagicDraw environment. The exact panel that this option will be available is up to the developer of the the plug-in. Moreover, the Configurator class assigns

the specific action that will be executed when the user starts the Consol-Optcad plug-in from MagicDraw's drop-down menu. In this case the BrowserAction class, which inherits from the DefaultBrowserAction class, is used to specify the action that will be performed. Inside the BrowserAction class the MagicDraw API [8] is used to access both block and parametric diagrams of SysML, and get the needed information from the model. After the needed data are gathered the functions of the TransformerImpl class are invoked to perform the transformation. The TransformationImpl class is part of the project that holds the generated code from the model transformation. When the transformation is complete, the Consol-Optcad model, in terms of Java classes, will be available. Here it should be mentioned that all the code generated by eMoflon for the transformations was made available to the Eclipse project by importing it as a .jar file, which works like a library. The next thing that the BrowserAction does is to call the functions available by the TransformationToText class. This class is mainly the adapter of Consol-Optcad. Each function of this class takes as an input an object of Consol-Optcad in Java and returns the Problem Description File, which is a text file that contains the problem formulation description for the Consol-Optcad environment. When all the objects are translated the plug-in starts Consol-Optcad and automatically opens the Problem Description File.

Figure 3.19 is the same plug-in architecture but in a more abstract representation.

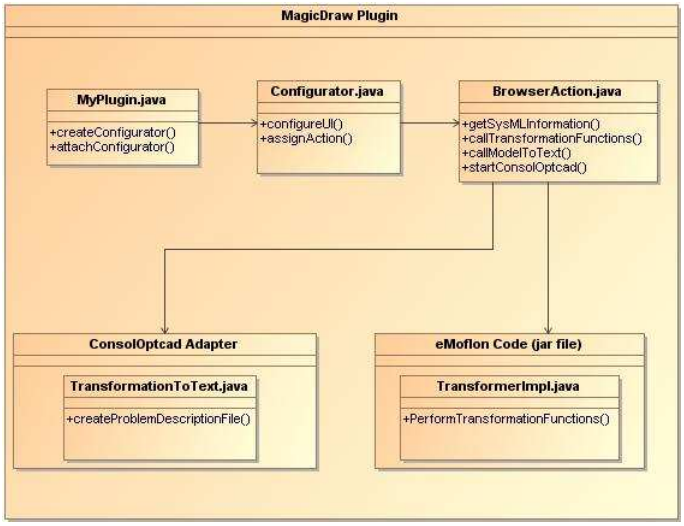

Figure 3.19: MagicDraw Plug-in Abstract Software Structure

Section 4.3 will illustrate better with the use of an example, the outcome of the integration process described in this chapter.

Chapter 4

Trade-off Analysis of an Electrical Microgrid

In Chapter 3 we analyzed the integration methodology and implementation. In this chapter the aim is to illustrate the way the integration works through an example. A trade-off analysis of an electrical microgrid was chosen as an interesting example for that purpose.

4.1 A Microgrid and its Components

The typical way of producing and distributing power is through a centralized power system. This way of generating power has served well the humanity during the last century. However it has some important inefficiencies. To generate power, those systems use mainly fossil fuels, which are a heavy environmental pollutant and also are available in reduced quantities year by year. Moreover 50% to 70% of the fuel used to produce power is lost as heat waste and around 8% of the generated power is lost in transmission lines. The infrastructure has huge maintenance costs and its complexity makes the whole system vulnerable and prone to black-outs. Besides that new investments in this market are difficult due to regulations and large capital investments that are needed. Finally, the price of electricity for consumers is high [22].

To address all these shortcomings the notion of Distributed Generation (DG)

has been developed. In DG the generating systems are of small scale, their use is local and they are geographically distributed. Distributed energy resources (DERs) offer very good power quality with less frequency variations, voltage transients or other disruptions, they can be used as back-up system for the electrical grid and also when the demand and charges for electricity are high. Furthermore, DERs offer low-cost energy, due to the local production, and many of them can be used as sources of both heat and power [22]. However, DERs can cause problems to the network, like reverse power flow, excessive voltage rise, increased fault levels, harmonic distortion and stability problems, because of their independent operation. To overcome problems caused by independent operation, DERs are grouped together and together with loads to form what is called a microgrid. A microgrid is operating as a single controllable entity. In Figure 4.1 a typical structure of a microgrid is depicted.

As it can be seen the point of common coupling (PCC) is the only connection point between the microgrid and the utility. A central role in a microgrid's smooth operation is played by the Energy Management System, which makes the decisions about generation and distribution of electrical energy. Those decisions are based on many factors, like power demand, weather, price of electricity and heat, fuel cost, emissions cost and government policies, to name a few. The DERs that take part on a microgrid can be electrical, thermal or a combination. Solar panels, small wind and hydro generators, micro turbines, diesel engines, fuel cells, gas turbines are some examples of DERs. At this point, more detailed information will be provided about the DERs that are used at the example developed to demonstrate the tool

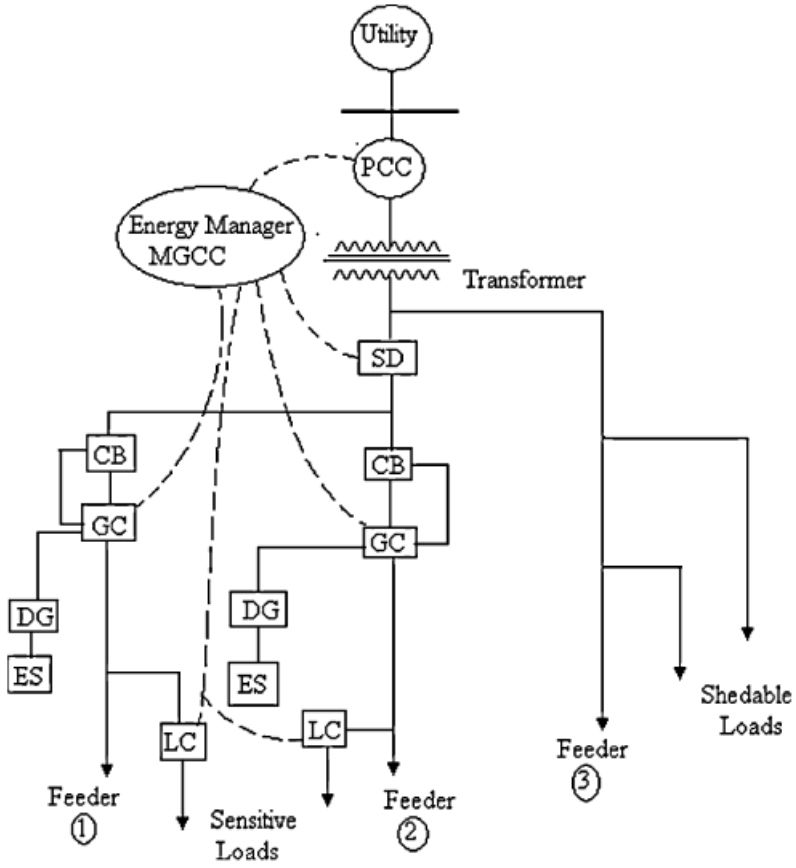

Figure 4.1: Microgrid Structure [22]

integration (see section 4.2).

Micro turbine

Micro turbines are small combustion turbines that were developed from technologies found in large tracks and in the auxiliary power unit (APU) of the aircrafts. They can produce power between 25-500kW, which is suitable for small offices, retail stores, hotels, apartment houses and restaurants. The prevailed type is the recuperated micro turbine, which gives efficiencies in the range of 20-30% and utilizes a variety of fuels, like natural gas, hydrogen,

diesel or propane. In Figure 4.2 a schematic of a recuperated micro turbine is presented [20].

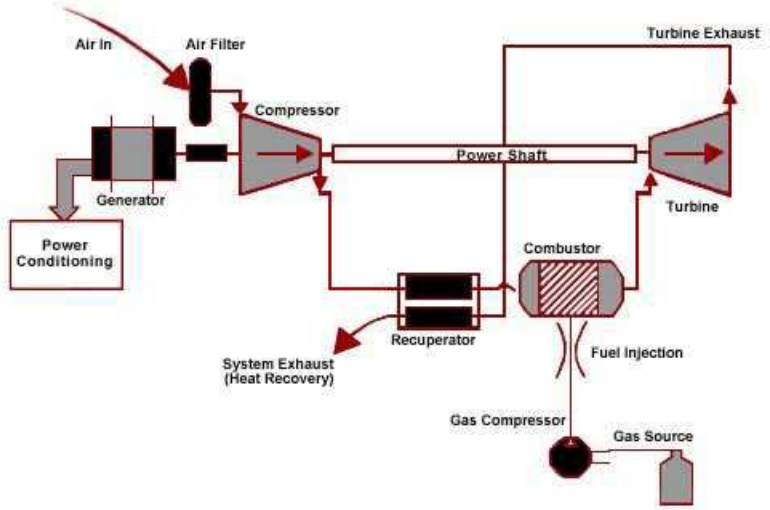

Figure 4.2: Micro Turbine Diagram

The recuperator recovers heat from the exhaust gas and boosts the temperature of the air supplied to the combustor. This increases significantly the efficiency of the micro turbine. If the micro turbine is used also as a thermal resource then the combined efficiency can reach up to 85%. Table 4.1 illustrates the major advantages and disadvantages of a micro turbine [20].

Fuel Cells

Fuel cells is a technology that is around for a long time and was first used by NASA. They work similarly to a battery in the sense that an electro-chemical reaction is used in order to create electrical current. The difference is that for the chemical reaction fuel cells use hydrogen and oxygen as reactants. Since those gas reactants can be fed into the fuel cell continuously, the unit will

	Micro turbine
Advantages	compact size, low-weight, low emissions, good efficiencies when used as source of heat and electricity, long maintenance intervals, can re-use waste fuels
Disadvantages	loss of power output and efficiency in high temperatures, low fuel to electricity efficiency

Table 4.1: Advantages and disadvantages of a micro turbine

never run down. When we are considering fuel cells as DERs we refer to a stack of fuel cells, because individual fuel cells can produce only low voltages [20].

There are many types of fuel cells. In this case the emphasis is on phosphoric acid fuel cells (PAFC), which is the only fuel cell technology that has been commercialized. Table 4.2 illustrates the major advantages and disadvantages of phosphoric acid fuel cells [20].

	Phosphoric Acid Fuel Cells (PAFC)
Advantages	Quiet, Low emissions, High efficiency, Reliability
Disadvantages	High costs, Low power

Table 4.2: Advantages and Disadvantages of Phosphoric Acid Fuel Cells

Diesel Engine

Diesel engines need to be coupled with an electrical generator to produce electricity. However for simplicity we use only the term diesel engine. Diesel engines are used for the production of electricity for many years. Their biggest advantage is the high power output and reliability however they produce more emissions than other DERs. In this work we focus on a Selective Catalytic Reduction (SCR) Diesel Engine, which has a mechanism to reduce NO_X emissions.

4.2 Problem Formulation

We define a Microgrid that consists of three power sources, one microturbine, one collection of phosphoric acid fuel cells and one diesel engine. The characteristics of each type of power source are listed in table 4.3 . The data were gathered from [16], [17] and [18].

The Microgrid is supposed to provide power to a residential building that has 50 apartments. We would like to find an optimal solution in terms of scheduling and power output of each engine for a period of 24 hours. The optimal solution is searched while trying to minimize operational cost, fuel cost, emissions and meet customer demand. We assume that the power source can be turned on and off two times only during a day. That is valid because of the costs that are associated with turning on/off each power source.

For the specific problem we define 5 design variables for each power source:

Power Source	**Capacity** (kW)	**Efficiency** %	**NO_X** (lb/MWh)	**CO_2** (lb/MWh)	**SO_2** (lb/MWh)	**K_{OM}** ($/kWh)
Phosphoric Acid Fuel Cells	25	37	0.03	1,078	0.0006	0.00419
SCR Controlled Diesel Engine	1000	38	4.7	1,432	0.454	0.01258
Micro Turbine	25	25	0.44	1,596	0.008	0.00587

Table 4.3: Characteristics of the power sources

- P_i $i = 1, 2, 3$ represents the output power of each power source
- t_{i_on1} $i = 1, 2, ...N$ represents the first time that power source i starts to operate
- t_{i_off1} $i = 1, 2, ...N$ represents the first time that power source i is turned off
- t_{i_on2} $i = 1, 2, ..N$ represents the second time that power source i is turned on
- t_{i_off2} $i = 1, 2, ...N$ represents the second time that power source i is turned off

The objectives and the demand constraint are described below:

Operational Cost

This objective aims to minimize the total operation and maintenance cost

needed for the microgrid. The formula that was introduced in [15] is used:

$$OM(\$) = \sum_{i=1}^{N} K_{OM_i} P_i t_{i_{operation}},$$

where N is the number of generating power sources, K_{OM_i} is a constant for each power source defined in table 4.3, $t_{i_{operation}}$(h) is the total time period that power source i is ON and P_i(kW) is the power output of each source.

Fuel Cost

Another objective in our problem is to minimize fuel cost. The following formula [15] is used to calculate fuel cost:

$$FC(\$) = \sum_{i=1}^{N} C_i R_i \frac{P_i t_{i_{operation}}}{n_i},$$

where N is the number of generating power sources, $C_i(\$)$ the price of fuel that each source utilizes, R_i(gallon/kWh) the consumption rate of each power source, P_i(kW) is the power output of each source, $t_{i_{operation}}$(h) is the total time period that power source i is ON, and n_i the efficiency of each power source.

Emissions

Emissions should also be minimized and they are calculated with the help of the following formula [16]:

$$EC(\$) = \sum_{i=1}^{N} \sum_{k=1}^{M} \alpha_k (EF_{ik} P_i t_{i_{operation}}/1000),$$

where N is the number of generating power sources, M the number of emission types, $\alpha_k(\$/\text{lb})$ a constant showing the cost of emission k, EF_{ik}(lb/MWh) is

the emission factor of power source i and emission type k, $t_{i_{operation}}$(h) is the total time period that power source i is ON and P_i(kW) is the power output of each source.

Meet Demand

Meet demand is a functional constraint that corresponds to the power demand in the fifty apartment residence of the example each time of day. Time is the free parameter and can take value from 1 to 24. Data from [19] were used to define an approximate function for the power demand:

$$Demand(kW) = 50 \cdot (-0.6 \sin(\frac{\pi t}{12}) + 1.2),$$

where t is time.

There are also three constraints that ensure the correct operation of each generating power unit:

- t_{i_off1} - $t_{i\_on1} \geq x_i$ AND t_{i_off2} - $t_{i\_on2} \geq x_i \;\; i = 1, 2, ...N$

  When a generating power source is turned ON, it shall remain ON for at least x_i(h) time units. If that constraint is not met then the power sources may malfunction.

- t_{i_on2} - $t_{i\_off1} \geq y_i$

  When a generating power source is turned OFF, it shall remain OFF for at least y_i(h) time units. If that constraint is not met then the power sources may malfunction.

The problem has a total of 15 design variables, 10 constraints and 3 objective functions. In the next sections the description of the problem in SysML will be presented together with the actual solution is Consol-Optcad.

4.3 Problem Solution using the SysML Consol-Optcad Integration

In this section a solution to the microgrid problem (see section 4.2) using our integration will be presented. The solution includes both the SysML model of the problem and the trade-off analysis part in Consol-Optcad, after the transformation is being performed.

4.3.1 SysML Model of the Problem

The first step towards a solution is to build in SysML the microgrid system structure as well as specify the trade-off analysis problem by utilizing the constructs offered by the Consol-Optcad profile (see section 3.2).

The block definition diagram that shows the three power sources that are part of the microgrid is shown in Figure 4.3. As described in problem formulation each of the tree power sources will have 5 variables. Those variables are easily recognized in the block diagram. They represent the output power of each source and the times the source goes on and off. There is also one constraint block named Transfer. This constraint block is used in the parametric diagram to pass attribute values of the power sources to the trade off analysis model. Moreover, this block should have one input parameter of type 'Real' and one output parameter of type 'String'.

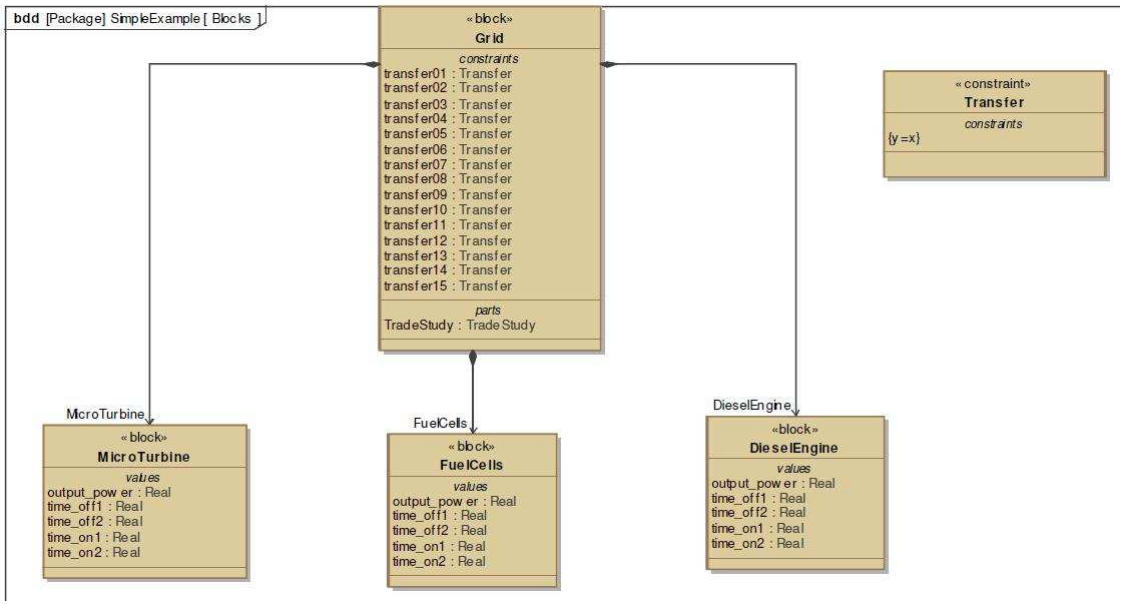

Figure 4.3: Block Definition Diagram of the Microgrid

The process continues by specifying the trade-off analysis model in SysML. For this part both the SysML block and SysML parametric diagrams are used. The block diagram represents the constructs of the trade-off analysis, like design variables, objectives and constraints. Parametric diagram is used to pass the information from the system structure to the trade-off analysis model. It should be mentioned that in order to pass parameters of the system for trade-off analysis, an instance of the system model should be created first (Figure 4.4). That is a trivial process that can be done automatically by MagicDraw.

The following figures illustrate better the SysML model of the problem. First of all the fifteen design parameters of the problem are presented in Figure 4.5. As it can be seen each design parameter has attribute tags that may be filled up by the designer. Those attributes can be used to bound the design variable or set its

Figure 4.4: Instance Diagram of Microgrid System

variation (see section 2.3.2.1). Also each design parameter has an attribute that is used to obtain the initial value of that parameter. That value comes from the instance diagram and is transferred through the parametric diagram. Figure 4.6 shows how the process of passing parameters is modeled.

As indicated also above, besides design variables, the objectives and constraints that take part in the trade-off study were modeled in SysML. Figures 4.7, 4.8 and 4.9 better illustrate how these constructs look like inside the MagicDraw environment.

When the hole modeling process in SysML is completed the designer can make the transformation and start using Consol-Optcad. This can be easily done by selecting the Consol-Optcad choice from MagicDraw's drop down menu, while having the parametric diagram open. This process is depicted in Figures 4.10 and 4.11.

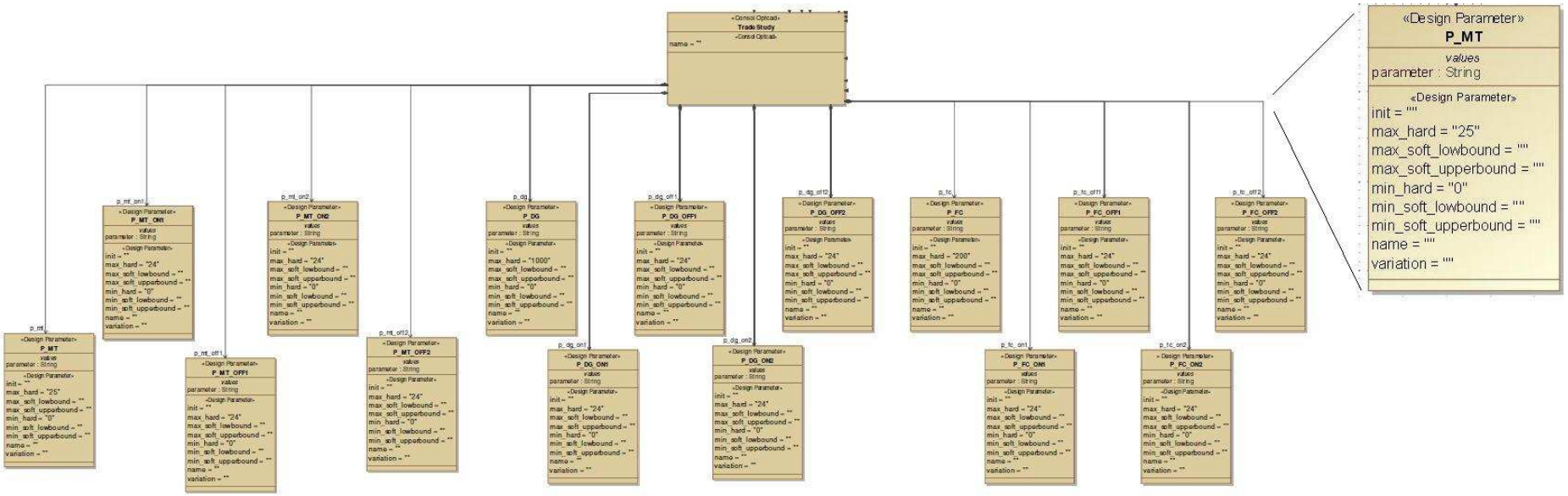

Figure 4.5: Design Parameters of Microgrid Trade-off Model

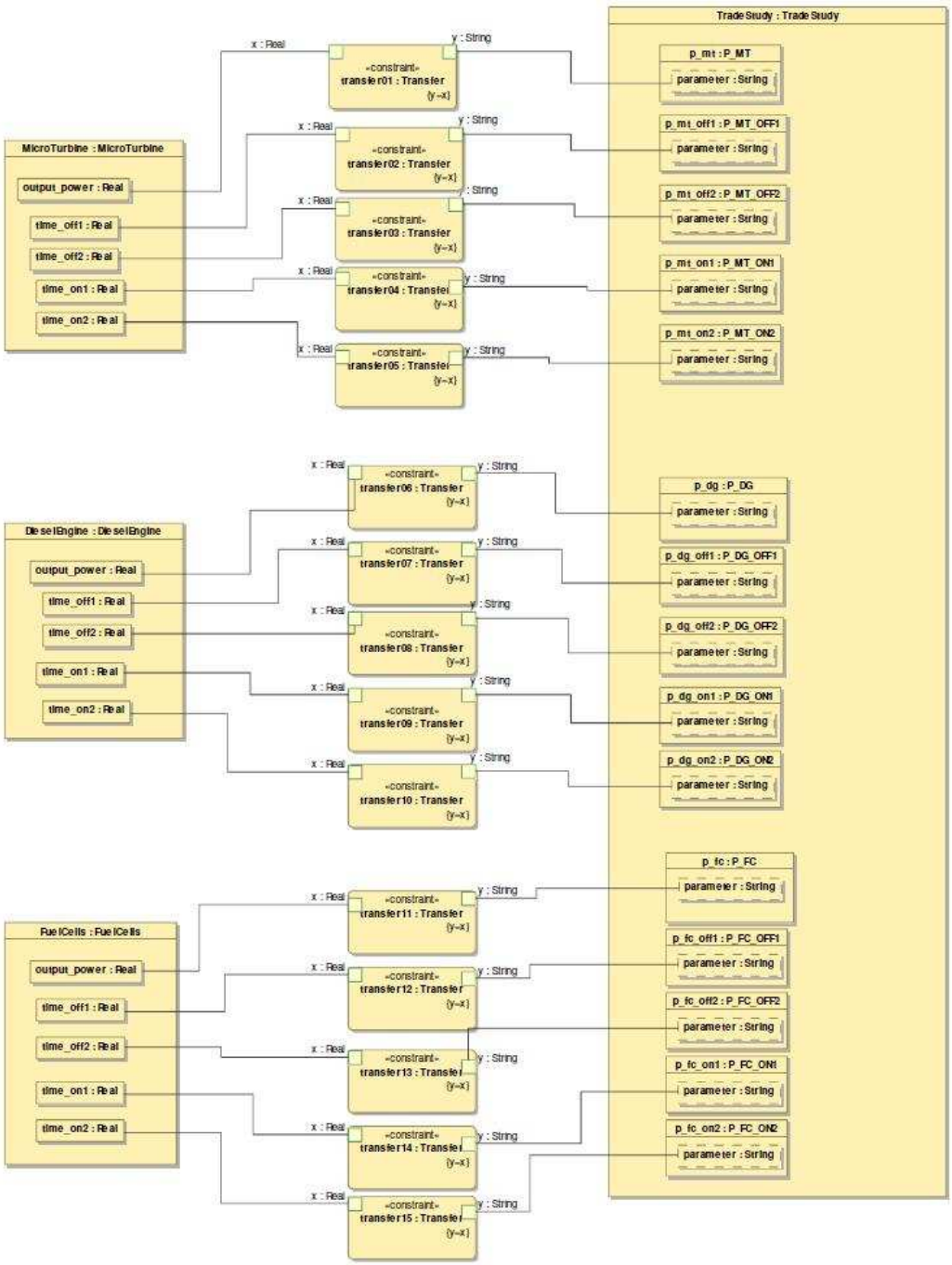

Figure 4.6: Parametric Diagram of Microgrid System

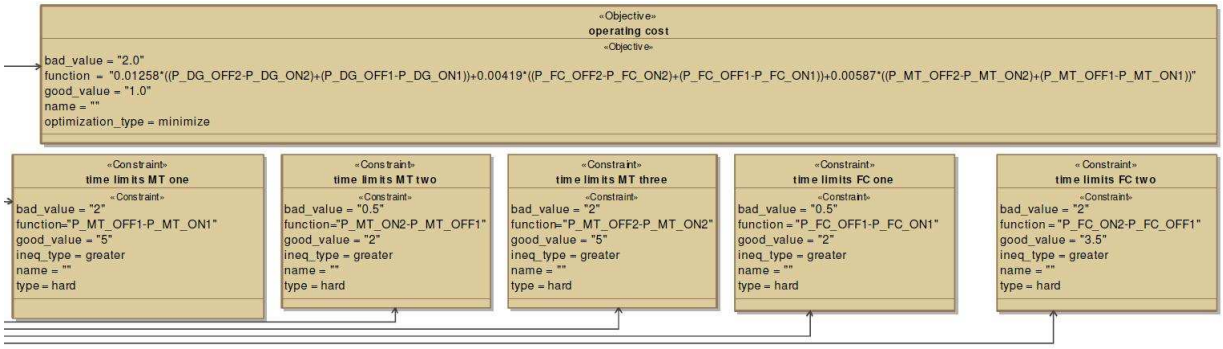

Figure 4.7: Operation/Maintenance Cost Objective and Constraints

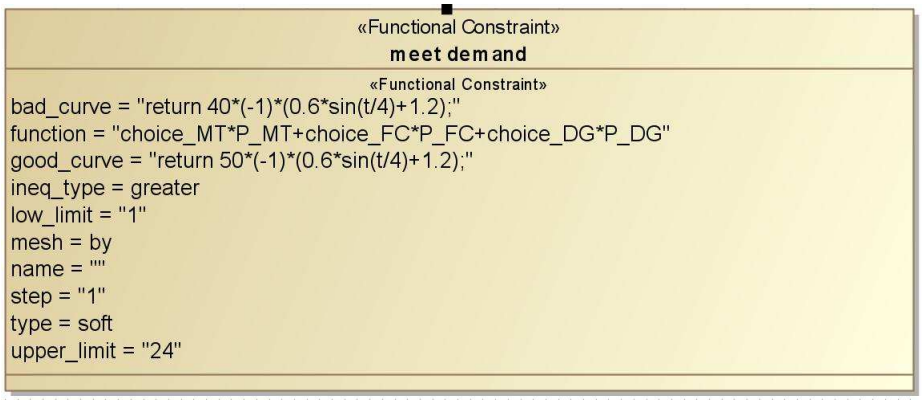

Figure 4.8: Power Demand Functional Constraint

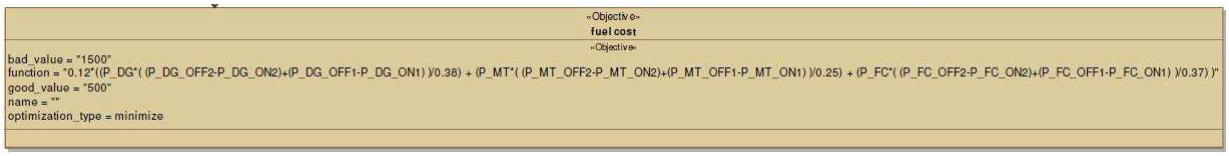

Figure 4.9: Fuel Cost Objective

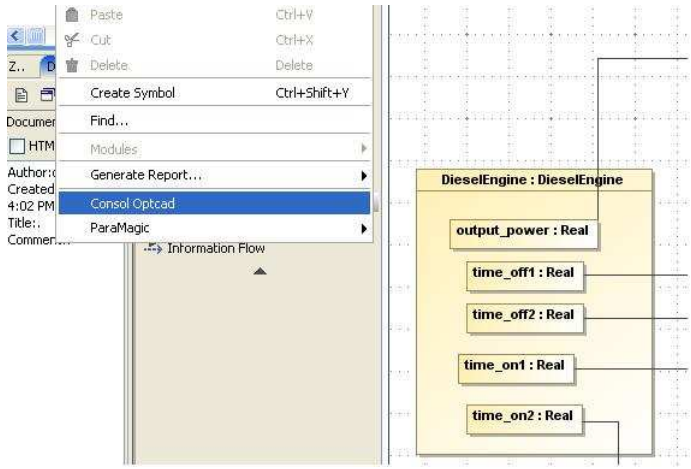

Figure 4.10: Calling Consol-Optcad from MagicDraw Environment

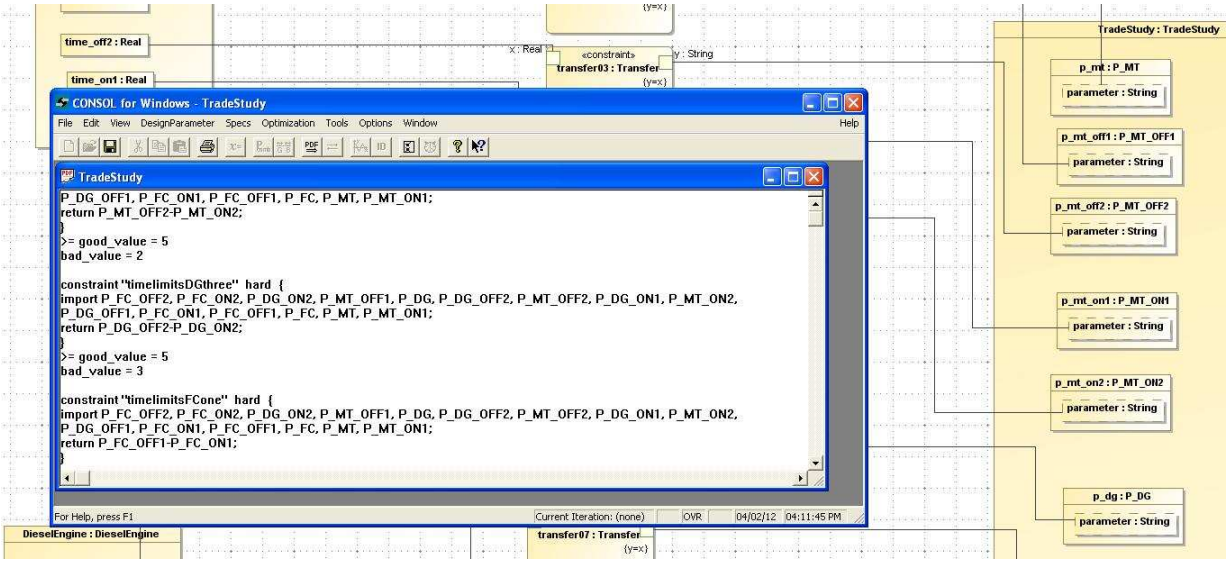

Figure 4.11: Consol-Optcad

4.3.2 Solving Problem in Consol-Optcad

After modeling the system and the trade-off analysis in the MagicDraw SysML environment the integration mechanism can be used to automatically transform the existing model in a problem description file inside Consol-Optcad (see Figures 4.10 and 4.11). For the multi-objective problem that is under consideration, we run the FSQP algorithm of Consol-Optcad several times with different initial conditions. Below the best solution found with regard to the examined initial conditions is presented.

Table 4.4 shows the initial conditions for all the design variables.

DER	**Power Output** (kW)	**time$_{ON1}$** (h)	**time$_{OFF1}$** (h)	**time$_{ON2}$** (h)	**time$_{OFF2}$** (h)
Phosphoric Acid Fuel Cell	15	4	12	17.5	24
SCR Controlled Diesel Engine	20	0	3	12	24
Micro Turbine	20	4	10	16	20

Table 4.4: Value of Initial Variables

The process followed to solve the problem will be illustrated with a series of screenshots from the Consol-Optcad environment together with relative comments

on each one of them.

Initial Phase

Figure 4.12 shows the performance comb (pcomb) at the beginning of the optimization process. Pcomb is the structure that Consol-Optcad uses to present to the user the results of the optimization process at each iteration. Pcomb includes information on the current value of an objective or a constraint and shows if that value satisfies the specified limits. Those limits represent good and bad values that were set by the user and they are marked in pcomb by vertical lines. From the pcomb, in the aforementioned Figure, it can be seen that one hard constraint is not satisfied. The normalized value of that constraint is depicted inside a red circle and the constraint is not met because that value is on the right side of the vertical line that represents the good value. A hard constraint shall strictly have a value above the good value limit while a soft constraint shall be at least above the bad value limit. All other hard constraints and objectives are satisfied at this point.

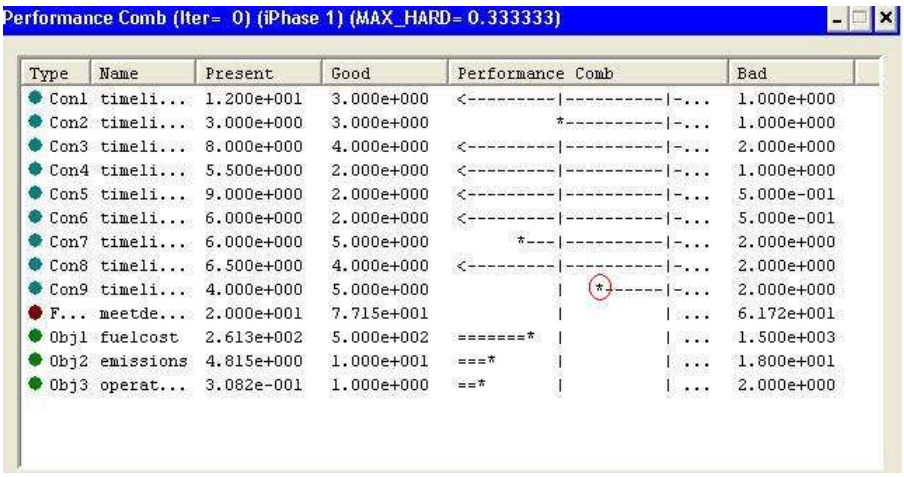

Figure 4.12: Pcomb - Initial Phase

However, the functional soft constraint that represents the need to meet the energy demand, is not satisfied. This is shown on pcomb (red dot) but is more clear in Figure 4.13 where the functional constraint curve (blue) is below the good (green) and the bad (red) curve most of the time.

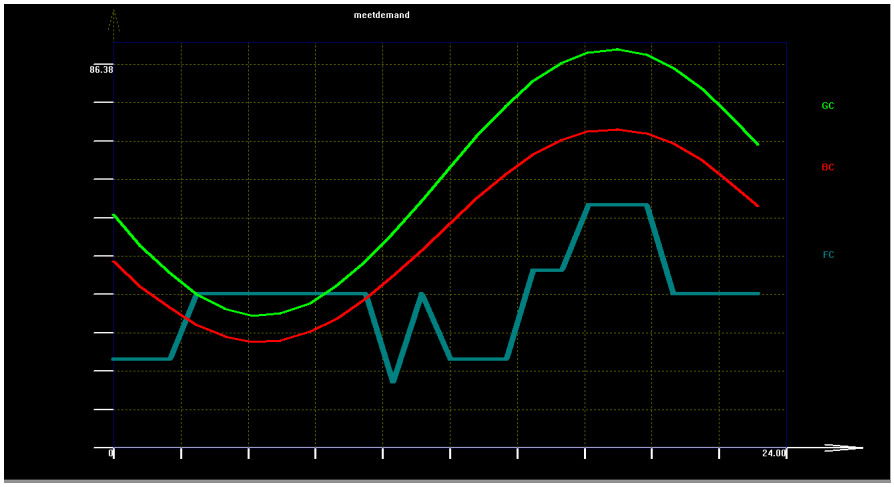

Figure 4.13: Functional Constraint - Initial Phase

Another thing to mention is that normally until all hard constraints are satisfied the user does not interact with the optimization process.

User Interaction (Iteration 18)

At this point all hard constraints are satisfied and all objectives are within limits, as is depicted also in Figure 4.14.

Moreover, Figure 4.15 confirms that the functional soft constraint meets the demand. Even through for a small period of time it goes below the good curve, is considered satisfied because it is specified as a soft constraint. Since all constraints are satisfied and the objectives are within limits we have a valid, feasible design.

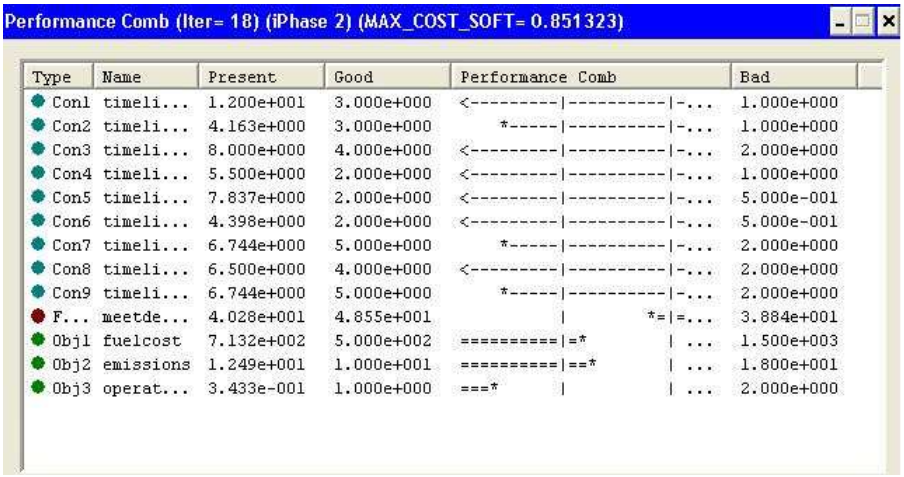

Figure 4.14: Pcomb after the 18^{th} Iteration

If the user is satisfied, the optimization can stop here. Also, if we continue the optimization process without any changes the next iterations will give also feasible designs, due to the FSQP solver used by Consol-Optcad.

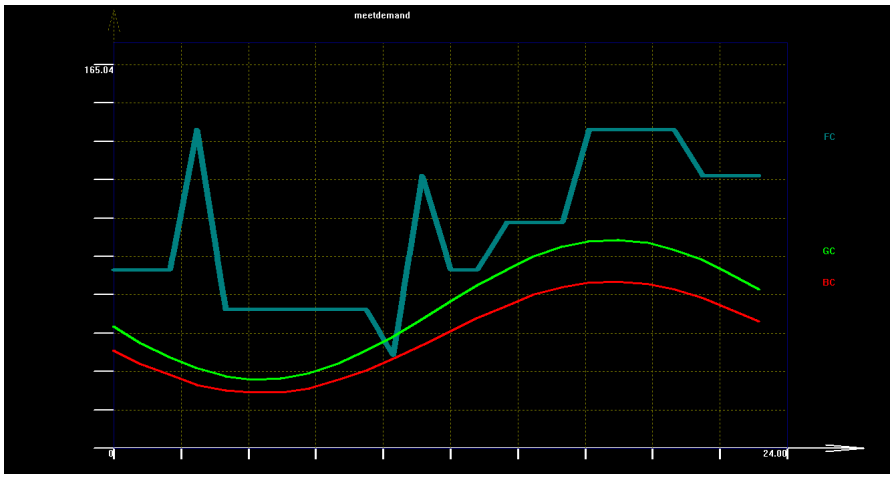

Figure 4.15: Functional Constraint after 18^{th} Iteration

However, one can observe that in order to meet the power demand at this stage, a lot more power is produced than the needed one. Therefore, we decide to

interact with Consol-Optcad and make the limits for fuel cost and emissions tighter and lower the power output. Those changes intend to force the optimizer to find a solution that will be more efficient while keep satisfying all constraints and objectives.

Final Solution (Iteration 52)

At the 52^{nd} iteration, we get a design that satisfies all hard and soft constraints and also meets the new tighter limits for fuel cost and emissions (Figure 4.16).

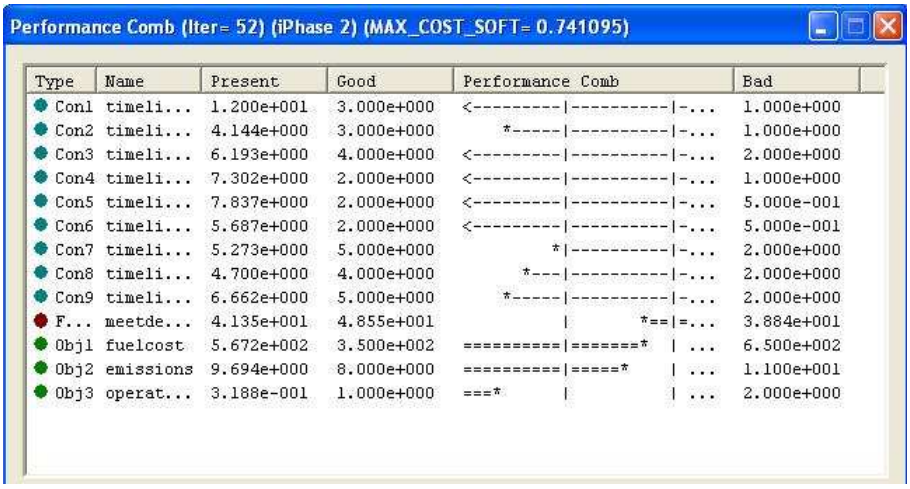

Figure 4.16: Pcomb - Final Solution

Figure 4.17 confirms that the demand is met and is obvious that because of the tighter limits on two of the objectives, less power is needed with the current design to achieve the desired result.

As it was mentioned in section 4.2 for the specific multicriteria optimization problem five design variables were examined. Figure 4.18 presents the results concerning the output power of each DER at the final design solution.

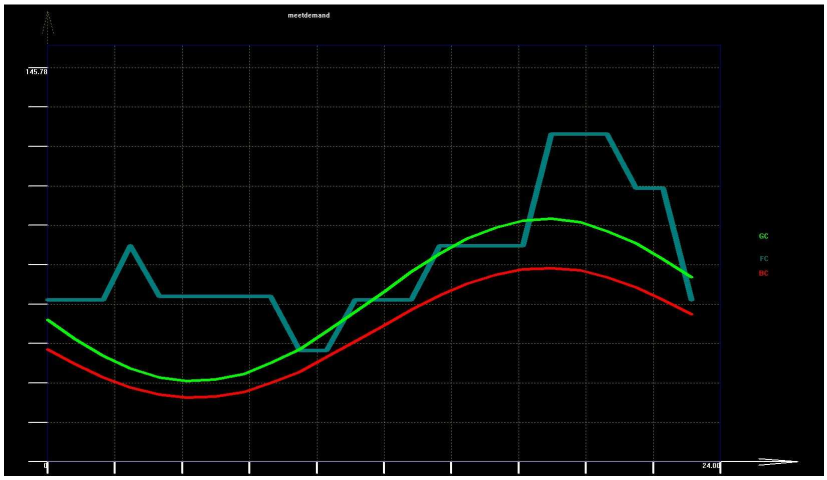

Figure 4.17: Functional Constraint - Final Solution

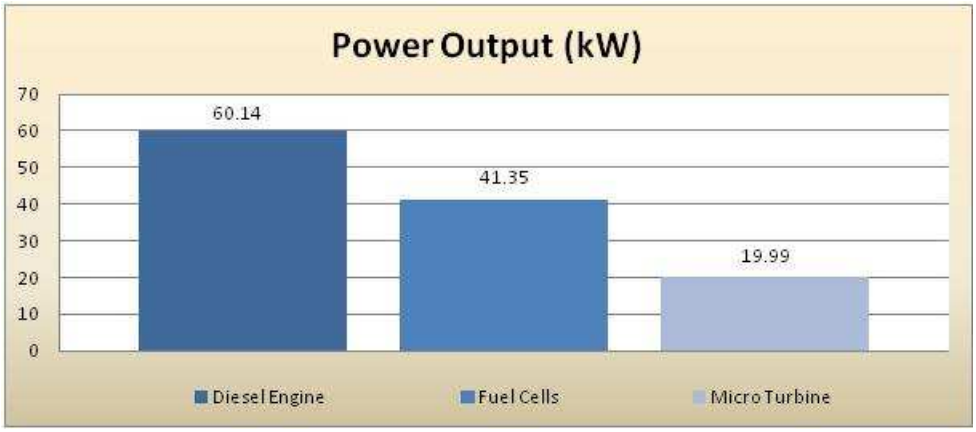

Figure 4.18: Power Output

Moreover, Figure 4.19 illustrates the scheduling of the DERs during a period of 24 hours. Also the diagram in this Figure provides information on the total power at each hour of the day and how much is the contribution of the different DERs.

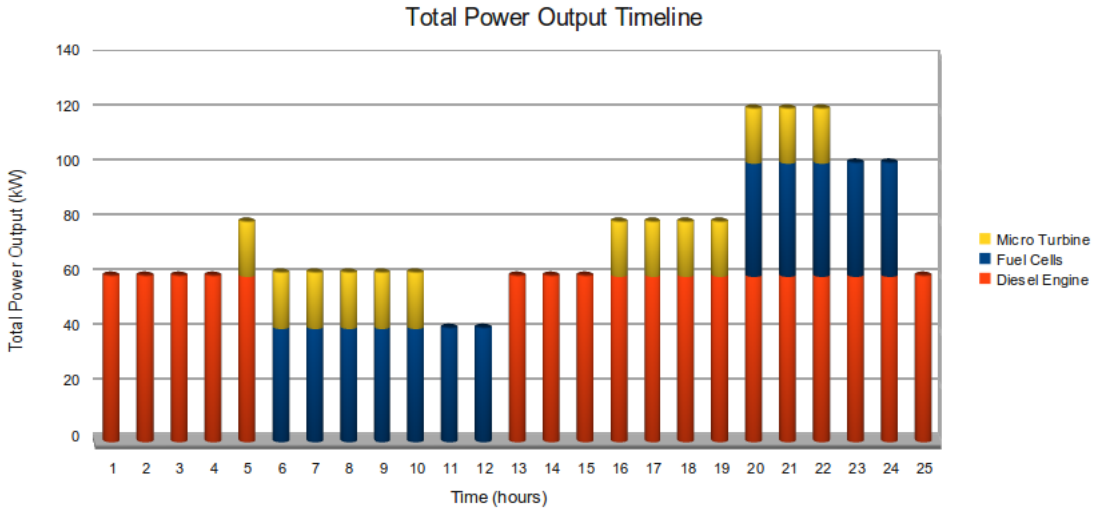

Figure 4.19: Scheduling Timeline

Chapter 5

Conclusions and Future Work

In this work we presented the modeling "hub" as a way to realize the Model-Based Systems Engineering vision and face today's challenges on complex engineered systems. Furthermore, we focused on the trade-off and design space exploration part of that hub and followed the proposed framework in order to integrate SysML with Consol-Optcad. We provided also details on how each step of the integration was implemented and what tools were used throughout this process. The SysML Consol-Optcad integration facilitates the problem formulation for the user and makes a first step towards having the design and optimization processes interacting and working in parallel in order to achieve the best possible design. Moreover, a trade-off analysis of an electrical microgrid was performed to demonstrate the results and utility of this integration.

However, there are still open issues for future research. First of all, regarding the implemented integration of SysML with Consol-Optcad one future goal is to make the integration bidirectional. Currently, the integration allows the transformation of a SysML model to a Problem Description File in Consol-Optcad environment. A bidirectional integration will allow values obtained from the optimization process to be returned back to the SysML model. Moreover, most of the multicriteria problems that occur while designing and developing a complex system, depend on both

continuous and integer design variables. This fact unveils the need to modify the solver of Consol-Optcad in order to handle also integer variables. Another interesting research challenge would be to try to integrate SysML with a trade-off tool that has many capabilities and is currently widely used in industry, like CPLEX.

In addition, the whole implementation of the modeling "hub" should be investigated further. The methodology that was proposed and followed in this work for the trade-off and design exploration part needs to be applied for all other phases of the Systems Engineering process. The appropriate tools need to be identified and then an integration with the "hub" should be implemented. This process will help towards identifying any shortcomings of the current hub architecture and hopefully at the end the modeling "hub" will provide a powerful toolsuit for holistic systems development.

Bibliography

[1] Sanford Friedenthal, Alan Moore, Rick Steiner, *A Practical Guide to SysML* (The MK/OMG Press, 2009).

[2] Mark Austin, *Class notes on Systems Engineering Requirements, Design and Trade-Off* (University of Maryland, College Park, MD, 2011).

[3] Mark Austin, *Class notes on Systems Engineering Design Projects Validation and Verification* (University of Maryland, College Park, MD, 2011).

[4] Cecilia Haskins, Kevin Forsberg, Michael Krueger, David Walden, Douglas Hamelin, *Systems Engineering Handbook* (INCOSE, San Diego, CA, 2011).

[5] International Council on Systems Engineering (INCOSE), *Systems Engineering Vision 2020* (Version 2.03, TP-2004-004-02, September 2007).

[6] Michael K.H.Fan, Andre L. Tits, Jian Zhou, Li-Sheng Wang and Jan Koninckx, *CONSOLE-User's Manual* (Technical Research Report, University of Maryland and Harvard, Version 1.1, June 1990).

[7] No Magic,Inc., *UML Profiling and DSL-User Guide* (Version 17.0, 2011).

[8] No Magic,Inc., *Open API-User Guide* (Version 17.0.1, 2011).

[9] The eMoflon team, *An Introduction to Metamodelling and Graph Transformations with eMoflon* (Version 1.4, TU Darmsadt, September 2011).

[10] A. Anjorin, M. Lauder, S. Patzina, A. Schürr, *eMoflon: Leveraging EMF and Professional CASE Tools* (INFORMATIK 2011, Bonn, Germany, October 2011).

[11] Alexnder Könings, Andy Schürr, *Tool Integration with Triple Graph Grammars - A Survey* (Electronic Notes in Theoretical Computer Science, Volume 148, Issue 1, Pages 113150 , February 2006).

[12] Thorsten Fischer and Jörg Niere and Lars Torunski and Albert Zündorf, *Story Diagrams: A New Graph Grammar Language based on the Unified Modeling Language and Java* (Universität Paderborn, 2000).

[13] M.B. Tischler, J.D. Colbourne, M.R. Morel, D.J. Biezad, *A Multidisciplinary Flight Control Development Environment and its Application to a Helicopter*

(Control Systems, IEEE Journal, Volume 19, Issue 4, Pages 22-33, August 1999).

[14] P.J. Potter, W.S. Levine, *Parametrically Optimal Control for the UH-60A (Black Hawk) Rotorcraft in Forward Flight* (Master's Thesis, University of Maryland, 1995).

[15] H. Vahedi, R. Noroozian, S.H. Hosseini, *Optimal Management of MicroGrid Using Differential Evolution Approach* (7th International Conference on the European Energy Market (EEM), Madrid, Spain, 23-25 June 2010).

[16] F.A. Mohamed, H.N. Koivo *Power Management Strategy for Solving Power Dispatch Problems in MicroGrid for Residential Applications* (IEEE International Energy Conference and Exhibition (EnergyCon), Manama, Bahrain, 18-22 December 2010).

[17] Sauli Jäntti, *Connection of Distributed Energy Generation Units in the Distribution Network and Grid* (CODGUNet Project Final Report, Merinova Technology Center, Vaasa , Sweden, 2003).

[18] W. Morgantown, *Emission Rates for New DG Technologies* (Regulatory Assistance Project, 2001).

[19] NAHB Research Center, Inc., *Review of Residential Electrical Energy Use Data* (Upper Malboro, Maryland, USA, 2001).

[20] The California Energy Commission, *Distributed Energy Resource Guide* (California, USA, 2012).

[21] Michael K.H.Fan, Li-Sheng Wang, Jan Koninckx and Andre L. Tits, *Software Package for Optimization-Based Design with User-Supplied Simulators* (IEEE Control Systems Magazine, Volume 9, Issue 1, Pages 66 - 71, January 1989).

[22] Ashoke Kumar Basua, S.P. Chowdhuryb, S. Chowdhuryb, S. Paul, *Microgrids: Energy management by Strategic Deployment of DERs - A Comprehensive Survey* (Renewable and Sustainable Energy Reviews, Volume 15, Issue 9, Pages 4348-4356, December 2011).