SysML Executable Model of an Energy-Efficient House and Trade-Off Analysis

Kersasp A. Cawasji
Institute for Systems Research
University of Maryland
College Park, USA
kersasp@terpmail.umd.edu

John S. Baras
Institute for Systems Research
University of Maryland
College Park, USA
baras@ isr.umd.edu

***Abstract*— With the growing complexity of energy efficient buildings, the methods of modeling and simulating such structures must account for monitoring several thousand design parameters across multiple diverse domains. As a result, modeling tools are now very specific to their respective domains and are growing more and more incongruous with each other. This calls for a way to integrate multiple modeling tools in the effort to create a single, large model capable of encapsulating data from multiple, different models. Thus, in this paper, different methods to perform an integration with Systems Modeling Language (SysML) and a simulation tool were identified, described and evaluated. Then, a new method was developed and discussed. Finally, the new method was demonstrated by developing a SysML executable model of a simple two-room house. Using the Functional Mock-up Interface (FMI) standard, the SysML model is integrated with a Modelica model, and a simulation is run in Simulink. Finally, a tradeoff analysis was performed for the purpose of design space exploration.**

Keywords—SysML, executable, FMI, integration, MBSE, Simulink

I. INTRODUCTION

When it comes to modeling such a large system, for the purpose of construction, all the tools to do that are not very congruent with each other. In this work, we will be approaching this problem from a Systems Engineering point of view with the intention of being able to run a simulation using a Systems Modeling Language (SysML) tool known as No Magic Cameo Systems Modeler. The definition and execution of engineering models, some of which may simply be represented as black boxes through the use of SysML constraint blocks is of great interest in terms of practicing Model Based Systems Engineering [1]. In addition, this work demonstrates the implementation of an Agile Model Based Systems Engineering approach through the usage of a SysML/ Model/ Simulation integration. This means that SysML was used to drive the models and simulations used in performance analyses and tradeoff analyses that are performed during system development [2].

To do this, using the Functional Mock-up Interface (FMI) standard, the SysML model is integrated with a Modelica model, through an intermediate Simulink Model, in which the simulation runs. Finally, a tradeoff analysis is run through SysML, in Matlab, for the purpose of design space exploration to demonstrate that meaningful decisions can be carried out using this approach. In this case, the tradeoff is between the cost of the thermal insulation used in the construction of the house versus the heat-load needed by the heat pump to maintain a constant indoor air temperature.

II. NEED FOR MBSE IN TODAY'S WORLD

Model Based Systems Engineering (MBSE) is an engineering paradigm gaining traction towards inculcating a model-centric approach to engineering instead of the age-old document-centric approach. As defined in the International Council on Systems Engineering (INCOSE) Systems Engineering Vision 2020, Model Based Systems Engineering is referred to as 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" [3].

The MBSE approach encourages Systems Engineers to improve the precision and efficiency of their communication with other Systems Engineers as well as stakeholders through the usage of a common visual modeling language [4], [5]. The most popular choice for this modeling language is the Object Management Group's System Modeling Language, commonly known as OMG SysML.

The need for MBSE is felt when clear communication is required between the system designers and various stakeholders across the Systems Development Life Cycle. MBSE is also able to capture and manage corporate intellectual property related to systems architectures, designs, and processes [4], [7]. Along with being able to provide a scalable structure, for problem solving, as well as being able to explore multiple architectures with minimum risk, an MBSE approach also helps in catching errors early in the Development Life Cycle. Through all these functions, the MBSE approach is able to enhance system performance.

In this work, in the spirit of MBSE, an attempt was successfully made to integrate Cameo Systems Modeler, an OMG SysML environment with an FMU, an output of Dymola,

978-1-5386-4446-1/18/$31.00 ©2018 IEEE

a Modelica based modeling and simulation engine. More about Cameo and Dymola and their usage in this paper will be discussed in Section III. The main reasoning for following such an approach was based on the reasoning that integrated models reduce inconsistencies, enable automation and support early and continual verification by analysis.

III. Modeling Tools

In Section II, the need for the Model-Based Systems Engineering approach was discussed. As models obviously play a key role in MBSE, it is important for all the different modeling tools to work together in harmony in order to approach the modeling of this energy efficient house from a Systems Engineering standpoint. In this regard, the three tools that were used to create the executable model were Cameo Systems Modeler, Dymola and Simulink.

A. Cameo Systems Modeler – SysML Environment

No Magic Cameo Systems Modeler is a commercial cross-platform collaborative Model-Based Systems Engineering (MBSE) environment, which provides smart, robust, and intuitive tools to define, track, and visualize all aspects of systems in the most standard-compliant SysML models and diagrams [7]. For the purpose of this paper, version 18.5 sp3 of Cameo Systems Modeler was used in a Windows 10 environment.

The reason Cameo Systems Modeler was the chosen SysML environment is because it is one of the two most widely used SysML Environments in the industry. The other most commonly used tool is IBM Rhapsody. In addition to that, the Systems Engineering courses at UMD were taught using Cameo Systems Modeler and it was also readily available for research purposes.

Systems Modeling Language or SysML is the modeling language most widely used by Systems Engineers. The main idea is that SysML is a standardized medium for communication; the rules defined in it give the model's elements and relationships unambiguous meaning. The capability to construct and read well-formed models is at the heart of the MBSE approach [8]. It enables the visualization of the system's design in the form of the four pillars of Systems Engineering, namely, the system structure, behavior, parametric relationships, and requirements. SysML is based off UML, however, it has multiple additions to it like Internal Block Diagrams, Parametric Diagrams, and Requirement Diagrams [8]. The SysML Diagram Taxonomy is shown in Fig. 1.

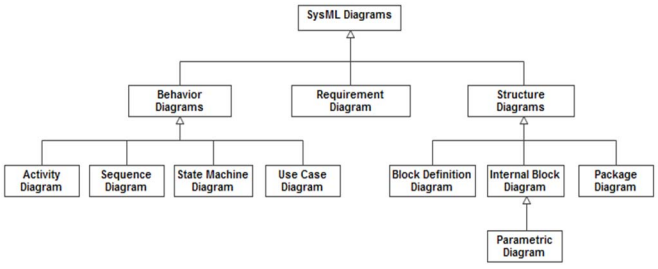

Fig. 1. The SysML Diagram Taxonomy [9]

B. Dymola – Modelica Environment

Dymola is a commercial modeling and simulation tool that is based on the Modelica modeling language. It is capable of modeling integrated and complex systems from various domains like mechanical, electrical, control, and thermodynamics. It is known for its multi-engineering capabilities with compatible model libraries for many different engineering fields. This allows for models of complete systems to be built that better represent the real world [10]. For the purpose of this paper, Dymola was used as the modeling tool used to create the Functional Mock-up Unit (FMU) of the two-room house. FMUs will be discussed in the following section.

Modelica, the modeling language used in Dymola, is a non-proprietary, object-oriented, equation-based language to conveniently model complex physical systems [11]. The Modelica Standard library consists of over 1600 model components and 1350 functions over many domains. The Modelica version used in this paper was 3.2.2. Since it is an open-source language, there are a large number of third party libraries that are built using Modelica.

In this paper, the *BuildingSystems* Library v2.0.0 beta was extensively utilized for creating the model of the energy efficient house. This model will be extensively discussed in Section 5.3. The library can be found on the GitHub page at https://github.com/UdK-VPT/BuildingSystems. It was developed by a team in the Universität der Künste Berlin under the guidance of Dr. Christoph Nytsch-Geusen.

The Modelica open-source *BuildingSystems* library is developed for dynamic simulation of the energetic behavior of single rooms, buildings and whole districts [12]. Using this library, modeling a living space and its HVAC system became possible. The library also allowed for the use of renewable energy systems to be included in the model.

C. Functional Mockup Interface

A Functional Mock-up Unit (FMU) is a product of the Functional Mock-up Interface (FMI) standard. This standard is tool-independent that helps support model-exchange and co-simulation of dynamic models using .xml files and compiled C code [13]. The first version, FMI 1.0, was published in 2010, followed by FMI 2.0 in July 2014 [13]. For this paper, the FMI 2.0 standard was utilized for the model exchange purpose. As mentioned in the standard documentation, the goal behind FMI for model exchange is that a modeling environment can generate C code of a dynamic system model that can be utilized by other modeling and simulation environments[14]. The model of interest is distributed in one zip file called FMU that contains several files like An XML file containing the definition of all exposed variables in the FMU and other static information; all needed model equations are provided with a small set of easy to use C functions; extraneous data is included in the FMU zip file, especially a model icon (bitmap file), documentation files, maps and tables needed by the FMU, and/or all object libraries or dynamic link libraries that are utilized [14].

D. Simulink/ Matlab

Simulink, like Dymola, is a commercial graphical modeling and simulation environment developed by Mathworks Inc., in

conjunction with Matlab. It offers a close integration with Matlab environment and can either drive Matlab or be scripted from it. Simulink is widely used in automatic control and digital signal processing for multidomain simulation and Model-Based Design [15]. Although Simulink is capable of modeling multi-domain systems too, for large systems with several hundred components, it is very cumbersome to do so.

For the purpose of this paper, Simulink wasn't used for the purpose of simulating the model of the energy efficient house. Rather, it was merely used as a shell or an interface that imported an FMU from the Dymola Model. This Simulink shell model was then imported in Cameo Systems Modeler to create the executable model making use of the preexisting Cameo Systems Modeler-Matlab Integration. The details of how exactly this was performed can be found in Section V, Subsection D.

IV. Current Approaches to Creating an Executable Model

A. OpenModelica – SysML Integration Using OMG Specification

The first approach that was tried for creating the executable model was a Java based approach using an existing specification called the SysML – Modelica Transformation Specification Version 1.0 using No Magic Cameo Systems Modeler and OpenModelica, another Modelica based environment [16]. This method was introduced by the Object Management Group (OMG) in 2012 where the vision was to provide a bi-directional mapping between SysML and Modelica to leverage the benefits from both languages. By integrating SysML and Modelica, SysML's strength in descriptive modeling can be combined with Modelica's Differential Algebraic Equation (DAE) solving capability to support analyses and trade studies [16]. Using this approach, Cameo Systems Modeler users could use a plugin created by the Model Based Systems Engineering Center at Georgia Tech University to import and export Modelica models to SysML [17]. These plugins were named SysML4Modelica and Modelica4SysML.

This seemed to be the ideal method to convert a Modelica Model into a SysML representation of it. This would enable an Internal Block Diagram (IBD) of the system to be created. Next, having developed the structural, behavioral and parametric diagrams, of the system, they could be linked to the parameters and linkages and other data items from the imported Modelica model, the next step would be to execute this SysML model and have it run a simulation within SysML itself. Using this functionality, a tradeoff analysis could then be performed.

However, on trying to implement this method, there was no indication of any other information from the Modelica model or any of its constituent blocks after having being imported in the Cameo Systems Modeler environment. There was also no mention of any ports, paths, values or properties listed in the specifications. In fact, there was no information at all and the specification was completely blank. In addition, there was no information about any of the equations, constraints or variables related to the Spring Mass Damper System. There was also an error pop-up that showed up when the Modelica Model was imported. The unhandled errors were Java related as is visible from the figure. However, due to a lack of experience with Java, not much of an effort could be made in debugging the program and solving the problem.

B. FMU Import into Cameo Systems Modeler Itself

The latest No Magic Cameo Systems Modeler beta version has crude support for importing FMUs using the older FMI 1.0 standard for Co-Simulation [18]. The Cameo Simulation Toolkit in Cameo Systems Modeler is capable of reading FMU files which are imported into the model in the form of FMU Blocks with the stereotype of ≪FMU≫. However, this functionality doesn't allow for FMUs to be imported into the system for the Purpose of Model-Exchange.

Also, for the purpose of the co-simulations, it was found that importing the FMU into Cameo didn't allow for the change in the input parameters that resided in the FMU block, making it practically useless for the purpose of tradeoff analyses. The restriction of the design space exploration ruled out this method for creating an executable SysML model. However, in the future, it is very likely that No Magic will update the Cameo Systems Modeler to be able to read FMU using the later FMI 2.0 standard for the purpose of model exchange as well as co-simulation. If this paper work should be recreated two more years from now, following this method could be a much better and more robust solution.

V. Executable Model Using SysML-Simulink-FMI Integration

A. Overview and High-Level Description of the Integration Procedure

In this section, the integration procedure will be defined. An IBD of the integration framework is shown in Fig. 2 to provide a high-level description of the framework. Following this, an in-depth description of the framework is provided.

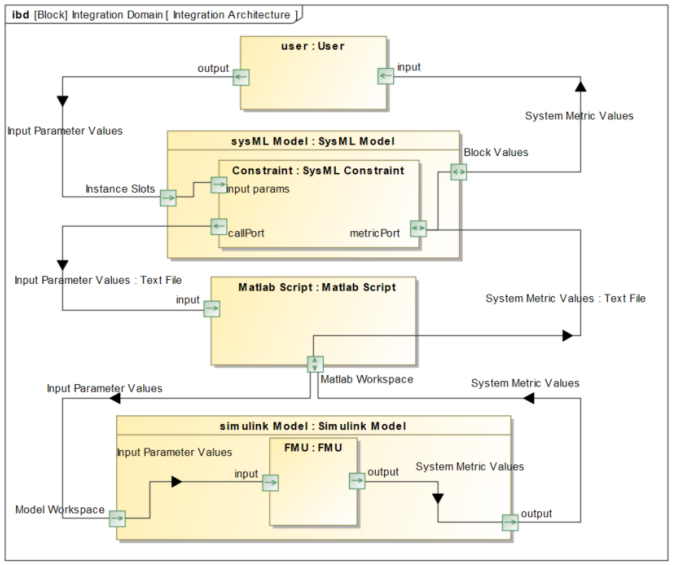

Fig. 2. Internal Block Diagram (IBD) of Integration Framework.

Cameo Systems Modeler is used to create systems architecture of the two-room house in SysML. Dymola is used to create a multi-domain Modelica model of the same two-room energy efficient house. This Modelica model contains all the internal equations, constraints and relations that govern the two-room house. It is also capable of accepting user defined input values to the design parameters of the systems, performing the calculations, and producing the output values for the system metrics. This Modelica model is then exported as FMU as shown in Fig. 3. The FMU is just a "skeletal structure" of the Modelica model and needs to be run from a different modeling tool to be able to accept input values and then output the corresponding metric values.

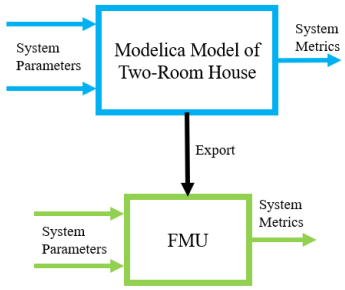

Fig. 3. Export of Modelica Model into the FMU

Next, The FMU is imported into Simulink as shown in Fig. 4. In this case, Simulink doesn't add anything new to the model, but merely acts as a shell or an interface to the FMU. Simulink was chosen as the interface since Cameo Systems Modeler and Matlab/Simulink have an existing integration that could be exploited for this usage.

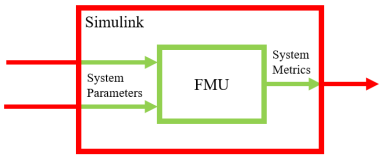

Fig. 4. Simulink as an Interface between the FMU and the SysML Model

Now, through the SysML Model of the two-room house, this Simulink model is called. User-defined design parameter values that were inputted into the SysML model are now sent to Simulink. With these values, Simulink will run the FMU, calculate the values of the output metrics and send them back to SysML to be displayed back to the user. This is shown in Fig. 5.

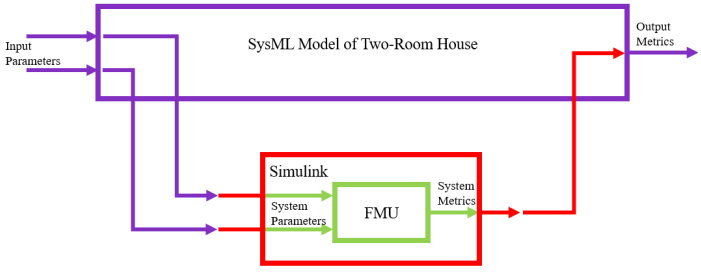

Fig. 5. Interaction between the SysML Model and Simulink Model

B. SysML Model

1) Structure Diagrams

The first structure diagram created was the System Domain Block Definition Diagram (BDD) as is shown in Fig. 6. This BDD articulates the structure of the system's domain, the system itself and its constituent elements. From the diagram it can be seen that the domain of the system comprises of the *System*, the *Users* and the *Environment*. The *System* itself is further comprised of the *Heating* system, the *Indoor Environment* and the *Building Structure* itself. Each of these subsystems are also broken down into their constituents. It is important to note that the *Building Structure* components are built using certain materials which are also shown as blocks in the Domain BDD. The operations of each component of the system as well as the values (variables) associated with each of the blocks should also be shown in a BDD. However, for the purpose of readability of the diagram, these details have been suppressed.

Next, the *Environment* is also broken down into its subsystems namely, the *Domestic Cold-Water Supply*, the *External Environment*, and the *Electricity Provider*. Finally, the *Users* of the system are also shown in their own block.

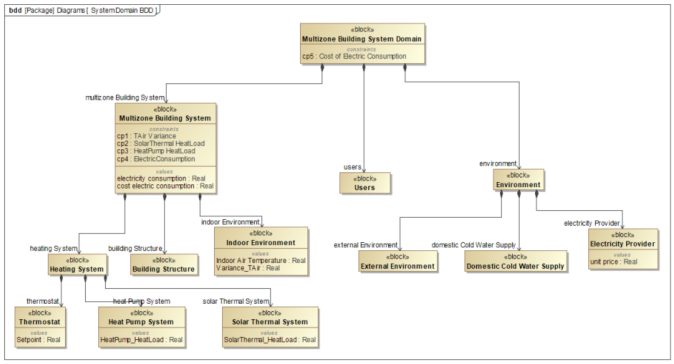

Fig. 6. System Domain Block Definition Diagram

2) Parametric Diagram

The diagram most important for creating the executable model is the SysML Parametric diagram. This diagram is used to express information about a system's constraints [19]. The constraints are in the form of mathematical models that determine the set of valid values within the running system.

This Parametric Diagram allows the previously described Simulink model to be treated as a "black box" constraint within the Cameo Systems Modeler SysML Model. This is achieved by using a special plugin called ParaMagic, developed by InterCAX Inc.

To create this parametric diagram, first all the factors and metrics have to be identified. For this model of the two-room energy efficient house, the metrics are shown in Table I.

TABLE I. TABLE OF SYSTEM METRICS AND UNITS

S No.	System Metric	Units
1.	Variance of Indoor Air Temperature	K^2
2.	Heat Pump Heat-load Per Annum	kWh
3.	Cost of Thermal Insulation	$

Although there are several different factors that affect the indoor air temperature, only few factors were selected as the design parameters. These are the parameters that can be realistically altered in order to improve the values of the metrics. Each design parameter, its associated SysML variable name and its nominal value can be found in Table II. The design parameters chosen were the thickness of the wood-fiber insulation layers used in the construction of the outer wall and the ceiling, the surface area of windows, and the indoor temperature setpoint.

TABLE II. TABLE OF DESIGN PARAMETERS, VARIABLE NAMES AND NOMINAL VALUES

S. No.	Design Parameter		Nominal Value
1.	Wood fiber insulation thickness	Outer wall	0.015 m
2.		Ceiling	0.255 m
3.	Window Area		3.0 m^2
4.	Indoor Temperature Setpoint		20 °C ~ 68 °F

Having identified the factors and metrics, a constraint block had to be first created in Cameo Systems Modeler [20]. Next all the parameters as shown in the above table were inputted into the Constraint Block as shown in Fig. 7.

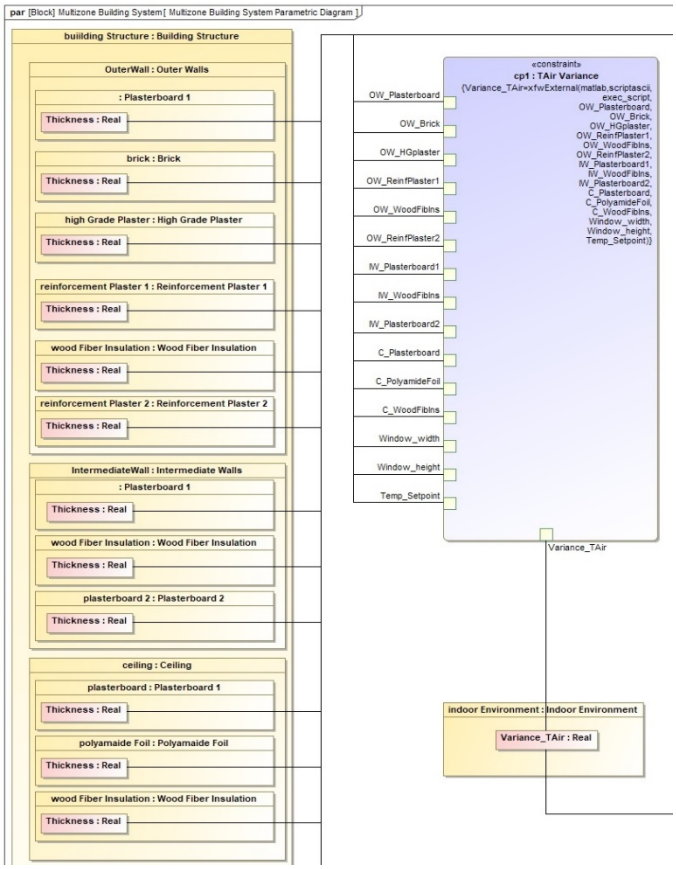

Fig. 7. SysML Parametric Diagram - Cropped

C. *Dymola Model of Two-Room Energy Efficient House*

The Dymola model of the two-room energy efficient house is shown below in Fig. 8. This Dymola Model contains all the equations, constraints and relations pertaining to calculating the values of the system metrics for a given set of input parameter values. This main model is named *SystemModel*.

This model can be broadly categorized into 3 separate component clusters. These are the, *Building* model, the *Ambient* model and the *House-Heating* model. These components are highlighted in Fig. 8. The *Building* block for this paper was developed by the authors using the *BuildingTemplate* template model class that can be found in the *BuildingSystems* Library at the path *BuildingSystems.Buildings. BaseClasses. BuildingTemplate*. The *Ambient* block was instantiated by the authors from *BuildingSystems.Buildings.Ambient* and configured. Finally, the *Home-Heating* cluster of blocks was adapted from an existing example in the *BuildingSystems* Library from the path *BuildingSystems.Applications. HeatingSystems.SolarHeatingSystem*.

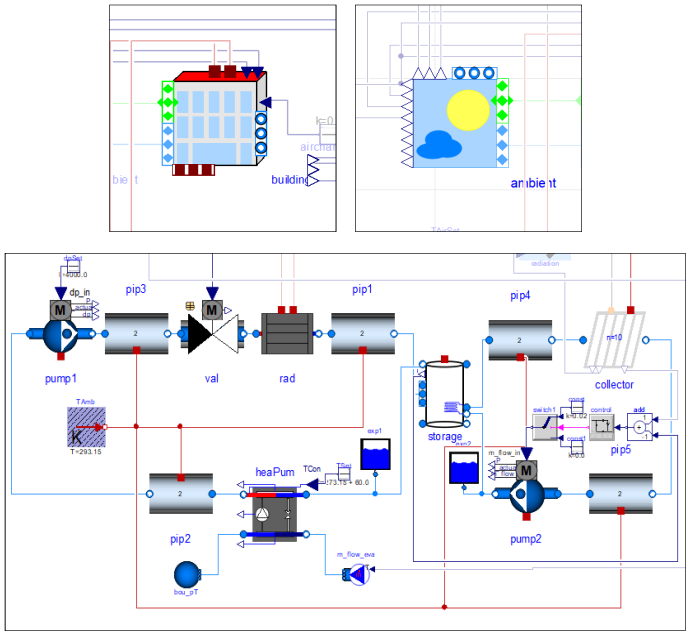

Fig. 8. The *Building* block (top left), the *Ambient* block (top right), the *House-Heating* block cluster (bottom).

At this stage, the Dymola model was ready, and the Functional Mock-up Unit (FMU) of the model was ready to be generated.

D. *Simulink as an Intermediate Model*

To integrate the Dymola model with Cameo Systems Modeler, Simulink was required to be used as an intermediate model. This is because the Dymola model was unable to be linked with SysML model either directly or through the use of the FMI standard. Since Matlab does have an existing integration with Cameo Systems Modeler, the idea was to somehow import the Dymola Model into Simulink which would then be programmatically accessible to Cameo through a Matlab script. This meant that the Simulink model would just be an

empty "shell" containing the full Dymola model without any alteration to it. The only value it adds to the model is the ability to link with SysML by exploiting the existing Cameo-Matlab integration. For importing the Dymola model into Simulink, a special block called the FMI block was required. This special block could be found in the FMI Kit for Simulink that is made available by Dassault Systems with Dymola. The FMI Kit enables embedding FMUs into Simulink [21]. The Simulink Model with the FMU loaded into it is shown in Fig. 9.

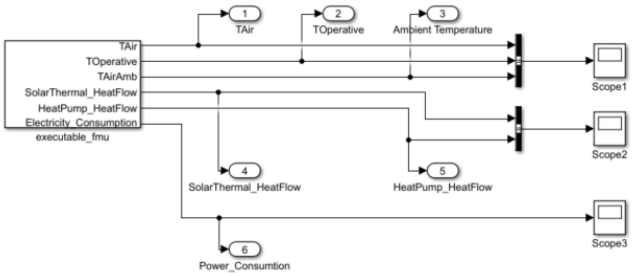

Fig. 9. Simulink Model

E. Integration Procedure

The way the integration works is that the user will input values for the design parameters in SysML in an "instance" of the model (Instances will be explained in detail below). These inputted values are then exported as a text file with the name *input.txt* by the SysML ParaMagic Plugin. The text file is then read by the Matlab script, assigning the inputted values to their corresponding variables and thereby inputting them into the Matlab workspace. This Matlab script then programmatically launches the Simulink model that was discussed earlier. Next, the script obtains the default parameters for the entire Simulink model from the FMU that resides in the Simulink model. Subsequently, these parameters are replaced with new parameters as specified in the input.txt file that is outputted by Cameo Systems Modeler. Following this step, using string manipulation methods, the old parameters string is edited to include the new parameters as mentioned by the input.txt file.

This is done by first, converting the old parameters string, which is saved as a `char` to a `string` for the purpose of string manipulation. Then the string is split into a `cell array` and the relevant cells are edited. The `cell array` is then converted back to a `string` and then back into `char` and fed back into the Simulink model to simulate.

Finally, the Simulink model is re-evaluated through the simulation and the metric is calculated over the period of the simulation time, in this case, one year. Lastly, the final value of the metric at the end of the simulation time is outputted from Matlab as a text file called *output.exe*. This file is then read by the ParaMagic Plugin in the SysML model and the stored metric value is then displayed in SysML and updated into the SysML model.

To conduct a trade study, several instances are created in SysML, each with a different set of input parameters. Based on the resulting values of the metric, a suitable set of design parameters can be chosen.

VI. Multi-Objective Trade-Off Analysis

For this system, since there are multiple conflicting metrics that define the quality of the system, a multi-objective trade-off analysis needs to be conducted in order to satisfy the objective function. Consol Optcad is one such multi-criteria optimization tool that uses a Feasible Sequential Quadratic Programming (FSQP) algorithm that would be best suited for a project like this [20]. The biggest advantage of this tool is its ability to change the parameters during the simulation in order to satisfy the constraints of the objective function, after having provided an initial parameter set. Another major benefit of such a tool is its ability to handle non-linear objective functions. Spyropoulos was able to utilize this tool in his thesis work to perform a multi-objective tradeoff analysis for an electric micro-grid system [23, [24]. Despite making continued efforts to recreate Spyropoulos's work to integrate Consol Optcad with SysML for performing the trade-off analysis in this project, it couldn't be performed for a variety of reasons. Mainly because of a lack of comprehensive documentation of the integration procedure as well as a lack of time, resources and expertise to perform the integration from the beginning. Yet, to demonstrate the power of the method, the trade-off analysis was performed in Matlab by performing a Pareto Frontier Analysis.

Taking various configurations of the design parameters of the system and performing a Pareto Frontier Analysis would provide a set of Pareto Optimal points that form the frontier and also demarcate the dominated region in the feasible solution set.

The first step to performing this analysis was to select discrete values of the previously defined design parameters and creating all possible configurations of them. Thus three levels of discrete values were chosen from the design parameters as shown in Table III.

TABLE III. Table of Design Parameters and Discrete Value Levels

	Outer wall Wood fiber insulation thickness	**Ceiling Wood fiber insulation thickness**	**Window Area**	**Indoor Temperature Setpoint**
Low	0.16 m	0.255 m	1 m^2	20 °C
Med	0.24 m	0.3825 m	2 m^2	21 °C
High	0.32 m	0.510 m	3 m^2	22 °C

The objective function used in the Matlab script was supposed to minimize all three of the metrics. The Variance of Indoor Air Temperature was minimized since it is desired for the indoor temperature to stay as close to the user-defined setpoint as possible, thereby making this a classic regulation problem. Next, the Heat Pump Heat-load Per Annum was minimized to ensure that the heat pump isn't being used more than necessary. Since it was very electricity intensive to run and was designed to be a backup to the solar system, minimizing the amount of heating produced by the heat pump was a desirable trait. Finally, the Cost of the Thermal Insulation used in the building construction was also minimized in order to keep the first costs of constructing the house low.

Thus, solving the simulation for the aforementioned metrics yielded the following results. The Pareto Analysis using these three metrics yielded 46 Pareto Optimal points as shown in Fig. 10.

However, out of 81 possible design configuration points, having 46 Pareto Optimal points isn't very helpful from a trade-off analysis point of view. Thus, some pruning of the Pareto Points was required to reduce the number of the choices to be presented. The first method of reducing the number of Pareto Optimals was simply by trimming the extreme values in all three dimensions. Since all three of the metrics have to be minimized, the numerically higher values of each metric could be deleted. Thus, any Pareto Point in the top 60% of the values of the Variance of the Indoor Temperature, or the top 60% of the values of the Heat Pump Heat-Load, or the top 85% of the Cost of Insulation was eliminated. This reduced the Pareto Points from 46 to only 12.

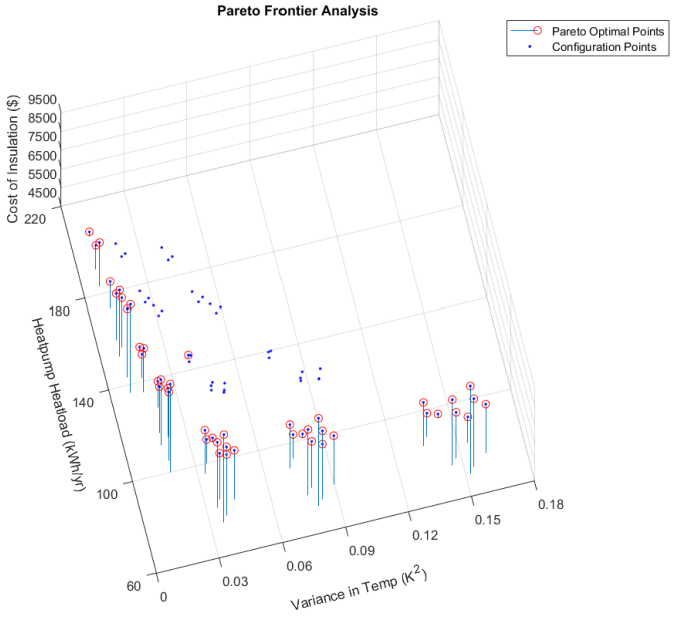

Fig. 10. Pareto Optimal Solutions – Tilted View

The next level of thinning down the Pareto Points was by finding points very close to each other in clusters and picking only one from each cluster to represent the whole cluster as a single design configuration, , leaving us with 6 Pareto Points.

From these last 6 points, 3 more points were eliminated as they provided a very small increment in the reduction of the heat-load and variance of indoor temperature in comparison to other points while costing close to $1200 more.

The final 3 points and their respective design configurations and metric values are shown in Tab. IV and Tab. V. They provide a good spread between the three metrics. Each of these points favors a particular metric and is a fair representation of the possible choices a customer would want to make based on the customer's preferences.

TABLE IV. TABLE OF FINALIZED DESIGN OPTIONS AND ASSOCIATED PARAMETER VALUES

	Parameters			
ID	**OW Insulation**	**C Insulation**	**Window Area**	**Temp Setpoint**
9	0.16 m	0.255 m	1 m^2	20 °
36	0.24 m	0.255 m	1 m^2	20 °
43	0.24 m	0.3825 m	1 m^2	21 °

TABLE V. TABLE OF FINALIZED DESIGN OPTIONS AND ASSOCIATED METRIC VALUES

	Metrics		
ID	**Variance in TAir**	**Heat pump Heat load**	**Cost of Insulation**
9	0.041 K^2	108.18 kWh	$ 4727
36	0.042 K^2	90.76 kWh	$ 5915
43	0.018 K^2	117.31 kWh	$ 6967

VII. CONCLUSIONS AND FUTURE WORK

Although the integration of SysML with a modeling and simulation tool for the purpose of reducing inconsistencies, enabling automation and supporting early and continual verification by analysis, was successful, the method of integration has a large scope for improvement. Since there was no way to simulate the FMU in Cameo Systems Modeler itself, the usage of Simulink as an intermediate model between SysML and the FMU caused the overall procedure to be fairly "clunky". This was very evident in the part where string manipulations to replace the default values for the design parameters, embedded in the FMU, with the new input parameters specified by the user in SysML, had to be performed in the Matlab script responsible for calling the Simulink model. If no changes are made to the FMU for any part of this integration, the string manipulations would not pose any problems. However, even if a single change is made to the FMU through Dymola, the string manipulations would have to be performed again to ensure that the right design parameter inputted by the user is mapped to the right variable residing in the FMU.

Next, the ParaMagic plugin that even made this integration possible in the first place also proves to be a bottleneck to its capabilities in some instances. The major drawback of this tool is observed when any constraint block in the SysML architecture calls a Matlab function or script to evaluate a system metric value. Although the Matlab script is capable to calculate multiple metric values in the same script, only a single metric value could be outputted back into the SysML model to be displayed to the user. Hence, for different system metrics to be computed by a single Matlab script, the same script had to be solved repeatedly, each outputting the value of a different metric. In this paper, this issue was overcome by setting the first Matlab script as the main calculation module and setting the subsequent Matlab scripts to just pull the calculated values of different system metrics from the first Matlab script. However, it was still a major drawback of the ParaMagic tool and a cause for inefficiency in the integration procedure.

As for finding a better integration technique in the future, the FMU import Cameo Systems Modeler described in Section IV, Subsection B of this paper shows the most promise. This method is still in development and if successfully completed, will eliminate the need to use an intermediate modeling tool like SysML to execute FMUs. The Model-Exchange enabled FMU could be directly imported into the SysML architecture of any system and could interact with the rest of the system architecture in a streamlined manner.

Another area worth exploring is the recreation of the OMG SysML-Modelica Transformation Specification mentioned in Section IV, Subsection B. For seemingly unknown reasons, this OMG standard is not compatible with the latest versions of Cameo Systems Modeler or Dymola. It would be of great value if the transformation was brought back to life, allowing for Modelica models to be transformed into SysML constructs. This method would in-fact eliminate the need for the FMU block altogether as the Modelica model would itself transform into SysML artifacts preserving all the data, equations, constraints and connections that are part of the Modelica model. However, this would only be most useful to those who are only working with SysML and Modelica as this method doesn't help with the integration of other modeling tools.

ACKNOWLEDGMENTS

Research was partially supported by ONR grant N00014-17-1-2622, DARPA STTR contracts through AnthroTronix Inc. and Boston Engineering Inc., and by the Lockheed Martin Chair in Systems Engineering.

REFERENCES

[1] S. Balestrini-Robinson, D. F. Freeman and D. C. Browne, "An Object-oriented and Executable SysML Framework for Rapid Model Development," *Procedia Computer Science,* vol. 44, p. 424, 2015.

[2] J. E. MacCarthy, *Approaches to Agile MBSE - 180326,* College Park, MD: University of Maryland, College Park, 2018.

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

[4] "Model-Based Systems Engineering Overview," MBSE.Works, [Online]. Available: http://mbse.works/mbse-overview/. [Accessed 28 February 2018].

[5] J. S. Baras and M. A. Austin, "Development of a Framework for CPS Open Standards and Platforms," *ISR Techn. Report* 2014-02, Univ. of Maryland 2014.

[6] Y . Zhou and J. S. Baras, "CPS Modeling Integration Hub and Design Space Exploration with Applications to Microrobotics," Chapter in the Volume *Control of Cyber-Physical Systems*, D. C. Tarraf (ed.), Lecture Notes in Control and Information Sciences 449, pp. 23-42, Springer 2013.

[7] No Magic Inc., "Cameo Systems Modeler," No Magic, [Online]. Available: https://www.nomagic.com/products/cameo-systems-modeler#intro. [Accessed 02 March 2018].

[8] L. Delligatti, "Chapter 2: Overview of the Systems Modeling Language," in *SysML Distilled: A Brief Guide to the Systems Modeling Language*, Crawfordsville, Indiana, Addison-Wesley, 2014.

[9] No Magic Inc., "Modeling SysML Diagrams," No Magic, [Online]. Available: https://docs.nomagic.com/display/SYSMLP182/Modeling+SysML+Diagrams. [Accessed 02 March 2018].

[10] Dassault Systemes, "Catia Systems Engineering - Dymola," Dassault Systemes, [Online]. Available: https://www.3ds.com/products-services/catia/products/dymola/key-advantages/. [Accessed 02 March 2018].

[11] M. Otter, "Modelica Overview," Modelica Association, 28 August 2013. [Online]. Available: https://www.modelica.org/education/educational-material/lecture-material/english/ModelicaOverview.pdf. [Accessed 03 March 2018].

[12] C. Nytsch-Geusen, "BuildingSystems," Universität der Künste Berlin, [Online]. Available: http://modelica-buildingsystems.de/index.html. [Accessed 03 March 2018].

[13] Modelica Association Project, "Functional Mock-up Interface," [Online]. Available: http://fmi-standard.org/. [Accessed 15 March 2018].

[14] Modelica Association Project, "Functional Mock-up Interface for Model Exchange and Co-Simulation," 25 July 2014. [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/v2.0/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf. [Accessed 15 March 2018].

[15] C. D. Bodemann and F. D. Rose, "The Successful Development Process with Matlab Simulink in the Framework of ESA's ATV Project," [Online]. [Accessed 15 December 2017].

[16] Object Management Group, "About The Sysml-Modelica Transformation Specification Version 1.0," November 2012. [Online]. Available: http://www.omg.org/spec/SyM/. [Accessed 04 January 2018].

[17] C. Paredis and A. Reichwein, "Sysml-Modelica Integration," Model-Based Systems Engineering Center, Georgia Tech, [Online]. Available: http://www.mbsec.gatech.edu/research/projects/active/sysml-modelica-integration. [Accessed 29 November 2017].

[18] No Magic Inc., "Simulation of SysML models," No Magic Inc., [Online]. Available: https://docs.nomagic.com/display/CST190/Simulation+of+SysML+models. [Accessed 13 January 2018].

[19] L. Delligatti, "Chapter 9: Paramteric Diagrams," in *SysML Distilled: A Brief Guide to the Systems Modeling Language*, Crawfordsville, Indiana, Addison-Wesley, 2014.

[20] InterCAX LLC., "SysML Parametrics Tutorial - HomeHeating," in *ParaMagic® 18.0 - Tutorials*, Atlanta, Georgia, 2014, p. 65

[21] InterCAX LLC., "SysML Parametrics Tutorial - HomeHeating," in *ParaMagic® 18.0 - Tutorials*, Atlanta, Georgia, 2014, p. 65

[22] D. R. Daily, "Trade-off Based Design and Implementation of Energy Efficiency Retrofits In Residential Homes," MS Thesis, MSSE Program, University of Maryland, College Park, MD, 2014.

[23] D. Spyropoulos, "Integration of SysML with Trade-off Analysis Tools," MS Thesis, MSSE Program, University of Maryland, College Park, MD, 2012.

[24] D. Spyropoulos and J.S. Baras, "Extending Design Capabilities of SysML with Trade-off Analysis: Electrical Microgrid Case Study," *Proceedings Conference on Systems Engineering Research* (*CSER'13*), pp. 108-117, 2013.