33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4-9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

# TEMPLATE-BASED SYNTHESIS OF 3D SYSTEMS FOR MBSE

Elias Allegaert[1], Yves Lemmens[1], Emre Ongut[1]

[1]Siemens Digital Industries Software, Interleuvenlaan 68, 3001 Leuven, Belgium

## Abstract

A methodology is presented to enable MBSE for systems that require CAD-based analyses by extending an existing system synthesis framework. Most importantly, a more general view of component interactions was needed. Now, geometrical information like points, axes, and curves can be exchanged between components. Furthermore, it is shown that the model-based methodology known for 1D system simulation is also applicable for CAD-based simulation synthesis. The methodology is demonstrated by synthesizing CAD, FE analyses, and dynamic simulation models in Simcenter 3D. In the first example, landing gear simulation models are synthesized by using a newly developed tool to manage system architectures and setting up simulation scenarios that can be synthesized. In the second example, a trailing-edge flap mechanism is defined and multiple configurations are synthesized to explore different modeling methods for the deployment mechanism. The methodology lowers the barrier to creating full-fledged heterogeneous system simulations for conceptual and preliminary design and investigating innovative solutions.

**Keywords:** MBSE, system architecture, model synthesis, landing gear, high-lift devices

## 1. Introduction

The 21st century has brought many challenges to the world's attention. From an engineering standpoint, many of these challenges can be relieved by technology. Aviation in particular, has an important part to play because it is connected to many problems and solutions. Aerospace engineering is a forerunner when it comes to implementing innovations. In an ongoing effort to increase product performance, design methodologies are developed to let the engineer work as efficiently as possible. Currently, model-based systems engineering (MBSE) is an attempt to improve the design process by providing better tools to handle the increasing complexity of systems.

In essence, the strategy of MBSE is to make it easier for people to collaborate by reducing the perceived system complexity. Domain experts have access to all needed information, but they should only expose the relevant information to non-experts through interfaces. This is a proven technique in software engineering of which object-oriented programming is probably the most obvious example. A secondary consequence is that the system and its engineering activities are easier to formalize. This makes it easier to structure the design process and design variants resulting in the engineering process becoming more machine-interpretable. This leads to design automation because, with a formalized definition, it is easier to express the design as a tree of decisions that can be simulated to find the most performant variant.

Although the MBSE approach is gaining traction in the aerospace industry and success stories get reported, its adoption is slow. One of the reasons might be that there are few examples of 3D systems being designed and optimized using an MBSE approach. Most MBSE-related papers focus on systems simulation, i.e. 1D simulation based on a bond-graph representation of the physics [1]. There exist frameworks where CAD and CAE models can be included. However, the CAD geometry is not able to adapt to the system it needs to fit in.

In this paper, an engineering approach is presented for using MBSE for 3D system design involving CAD, FEM, and multibody simulation. The methodology makes use of templates and aims to maximize their reuse. Therefore, the methodology is most applicable to mechanical systems with a reasonable degree of modularity. In the next section, a brief discussion on MBSE is provided with a

focus on the simulation of system performance. In the following section, our methodology is discussed after which two examples are given: simulation synthesis of landing gear and high-lift devices. The last chapter is a synthesis of how this work fits into past and current efforts in systems engineering and offers a humble guess of what the future might hold.

## 2. MBSE: Promises and Challenges

### 2.1 MBSE drives design automation

Model-Based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later lifecycle phases. The output of MBSE activities is a coherent model of the system and its configurations where the emphasis is placed on evolving and refining the model using model-based methods and tools. An MBSE approach focuses on creating a system model that would act as a single source of truth for information.

Implementing an MBSE approach should provide numerous benefits to an enterprise including:

- Better communication among teams
- Improved product quality
- Reuse of system specification and design artifacts across a product's lifecycle
- Enhanced knowledge transfer across domains
- Increased productivity

Ideally, systems modeling starts with requirements and functional definition. The complex interactions between functional and non-functional requirements can best be seen as a model by itself. This requirements model can serve as the basis to derive the components of the system. However, high-level important design decisions need to be made because there might be different system architectures possible. A formalized definition of the system components and constraints can be given to a dedicated solver to find all possible system architectures [2]. In this way, the possibility is eliminated that good architectures were missed or discarded because of a bias of the engineer. Figure 1 illustrates the generating of system architectures and simulating them in Simcenter Studio.

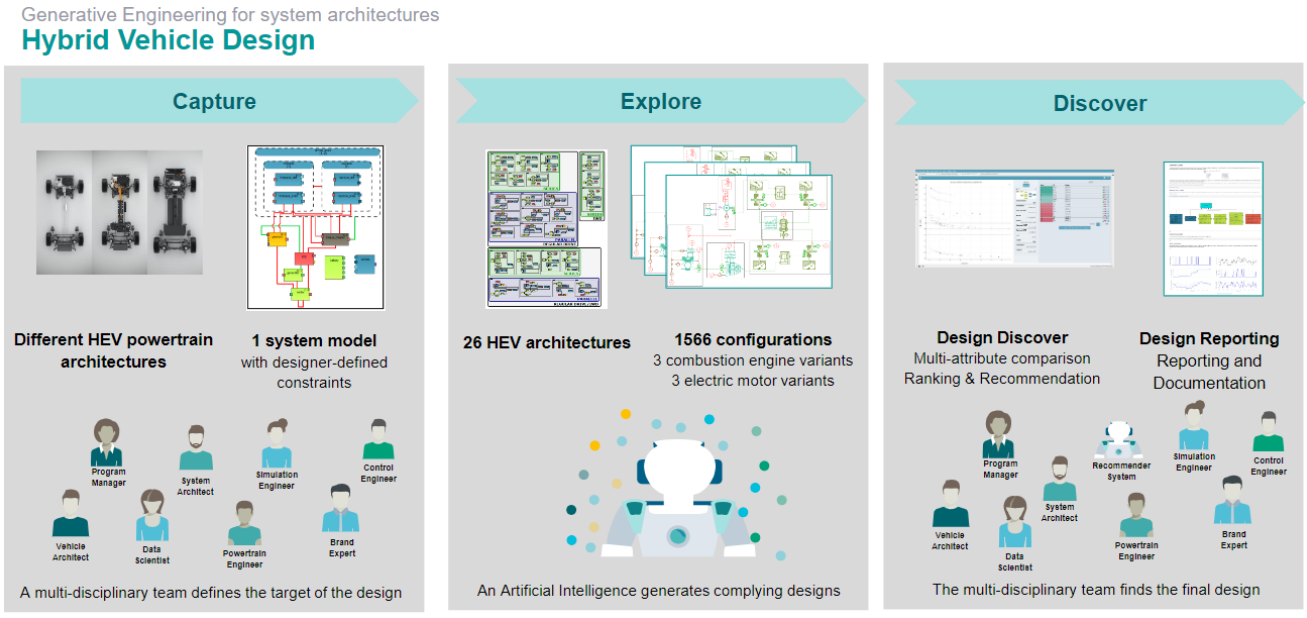



Figure 1: In Simcenter Studio, the user can define the system based on existing architectures or a meta-model of the system. A solver finds all possible architectures which can intelligently be evaluated through simulation to find the most optimal design.

The modeling of requirements, functions, and logical decomposition, a.k.a. system architecture, can be done in the SysML language [3]. However, a model language alone helps little in the next steps of the design process: the modeling of physical objects and their behavior. The reason is that requirements, functions, and components are abstractions. The internal details are left out to make the modeling manageable. When the system modeler creates a requirement in the model stating that a component shall be corrosion resistant, is it expected that the modeling language is so rich that it contains the concepts of redox reactions down to the level of quantum mechanics? How else can the requirement be verified? This leads us to a couple of challenges that need to be addressed.

## 2.2 Technical challenges of MBSE

Obviously, physical reality is so complex that there exists different analysis and simulation software for every aspect of it. Each simulation tool is just “one” implementation to represent the physical behavior of an object. In developing the physical theory and software, countless assumptions and nomenclatures have been adopted. Here the formalization ends and things get messier. The human engineer needs to interpret the requirements, functions, and system architecture and then create or retrieve the simulation models that correctly represent the component’s behavior. This process is sometimes abbreviated to RFLP [4][5]. Finally, the engineer needs to map the abstract concepts of the system model to the concrete attributes of the simulation model. For example, parameters that were defined in the SysML model need to be linked to the parameters of the simulation model. When the simulation is finished, and a result is retrieved, it can be verified if the design satisfies the constraints. Full-fledged optimizations are often needed, and setting these up can be time-consuming. Nowadays, there exist frameworks that can automatically set up the optimization based on the abstract system model [6][7]. Still, much time needs to be spent on tedious “bookkeeping” tasks to set up a simulation workflow, despite having a formalized representation of requirements, functions, and system architectures.

In order to perform a Design Space Exploration as efficiently as possible, some automation can be achieved by building a library of template models that can represent the components. Most system simulation software, e.g. Simcenter Amesim and Simulink are built around such an extensive library. However, it is possible to combine component models into super-component models. Simcenter System Architect and Simcenter Studio provide the means to convert a system architecture into a simulation model of the complete system by a so-called synthesis. A lot of time can be saved by this integrated approach, especially when the system is modular and models can be reused [8].

This work must be seen as an extension of existing architecture-based synthesis approaches that exist for 1D system simulation [2][9]. In this paper, we define 3D systems as systems that contain components for which geometry is essential for performance. Therefore, a CAD model is needed to perform analyses such as Finite Element Analysis (FEA) and multibody simulation. Two important differences between 1D and 3D systems are:

- Components depend on other components for their geometry. Where 1D models often abstract the geometry away, 3D models often require exchanging points and orientations to connect.

- Geometry can be infeasible when CAD models need to connect to the neighboring components. In general, CAD models are less robust than their 1D counterparts. Therefore, a lot of attention needs to be spent on the synthesis code of these simulation models but also on the modeling itself.

Two ways can be identified to automate the generation of the 3D simulation models: from code or templates. Specifying the model as code can be done by directly using the CAD program’s API or by using a Knowledge-Based Engineering (KBE) system that indirectly uses a CAD kernel’s API by providing high-level functionality to the modeler [10]. The work described in this paper uses the template-based approach to synthesize the simulation models. The main advantage is that the expert modeler can use the Graphical User Interface (GUI) of the CAD program instead of

programming. The disadvantage is that the adaptability of the model is limited because all design variation needs to be expressed as parameters [7]. However, visual scripting in modern CAD software partially removes this limitation.

## 3. Template-based synthesis of 3D simulation models

### 3.1 Extending the concept of connections and joints

The presented template-based methodology for 3D systems builds further on existing frameworks [2][6]. Many concepts from 1D (bond graph) model synthesis can be transferred to 3D (CAD-based) synthesis. This is logical because the 1D simulation models had an abstract representation in the system model which only needs a more general representation for 3D models. Figure 2 shows the relation between the system, components, abstract models, and the specific behavioral model in Simcenter 3D which can be a CAD, FEM, or multi-body simulation model. Two additions were needed. First, a type of connection between 3D geometry was made. Second, a joint type of element was added to represent physical joints that constrain degrees of freedom.

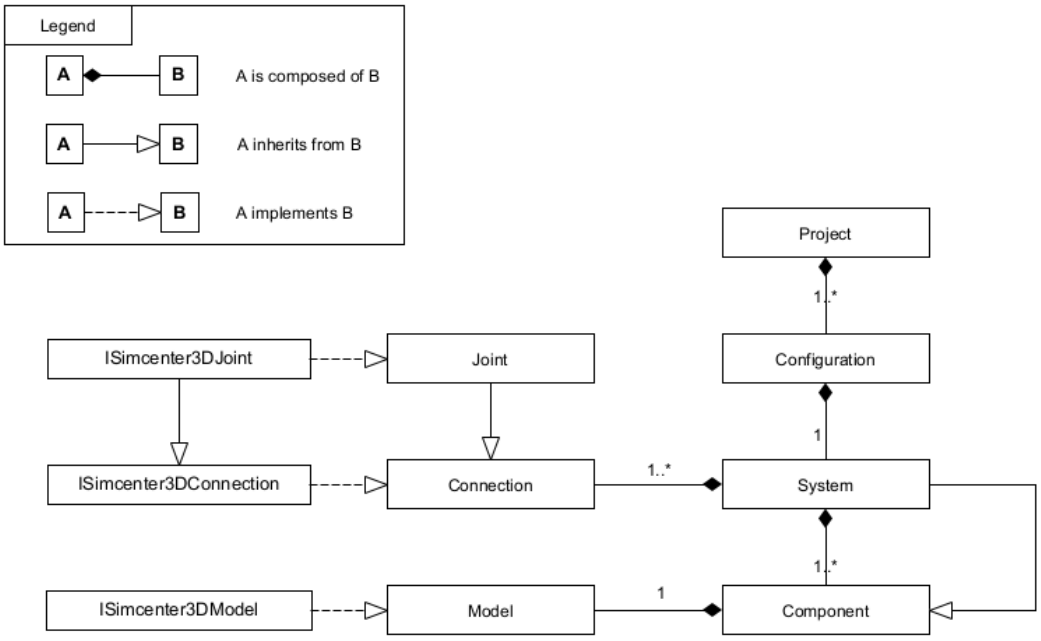



Figure 2: UML class diagram defining the ontology of the system.

Systems consist of components that interact with each other through connections. For 1D system modeling, synthesis only requires the connection of the ports of components so that effort, flow, and data can be exchanged [11]. However, for 3D systems that need CAD models to represent physical behavior, connections are not a manifestation in the simulation domain but an essential part of the design. Therefore, component connections need to exchange geometrical information like points, axes and curves between components. Model templates use the shared geometrical elements to build on and provide new elements for other models. Additionally, a group of geometrical elements can be assigned to represent a physical joint. For example, a point and an axis can be one of the inputs of a model on which other geometry is constructed, see Figure 3. When that CAD model is the basis for a FE model, that point and the axis are used to create a joint with one rotational degree of freedom denoted by the axis. It was found that interfaces between components should be defined as generally as possible to increase the reuse of templates. This agrees with the findings of Bussemaker et al. [12].

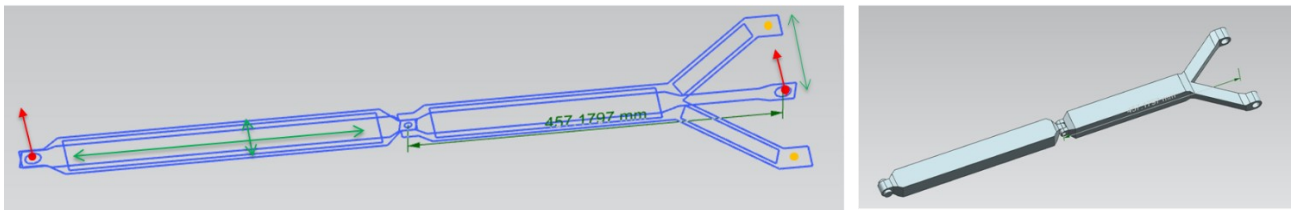



Figure 3: (left) Sketches on which the extrusions are built which are visualized on the right. Red: input geometry, Green: parameters, Orange: output geometry which cannot be directly controlled because it is the result of input geometry and parameters. The orange points are used to make a joint between the brace and the airframe.

### 3.2 Compressing the template library by introducing hierarchy

In the spirit of Object-Oriented Programming (OOP), it should be encouraged that detailed models are an extension of simpler models. This becomes concrete when looking at the structure of CAD parts and assemblies. It is possible to import a simple CAD model with stick geometry into a new CAD part and let it be the child of a parent part where the 3D geometry lives. The stick geometry of the simplified models is used as a skeleton for the extrusions and other features of the detailed CAD model. In NX, the geometry can be shared between parts through WAVE links. It is important to note that creating robust CAD models requires practice [13]. Figure 4 shows the FE model of the housing and its detailed CAD model. When a change is made to the stick model, the update propagates automatically to the detailed model. Analogously, this also happens for Finite Element (FE) models and multi-body simulation models which have a CAD part as a child.

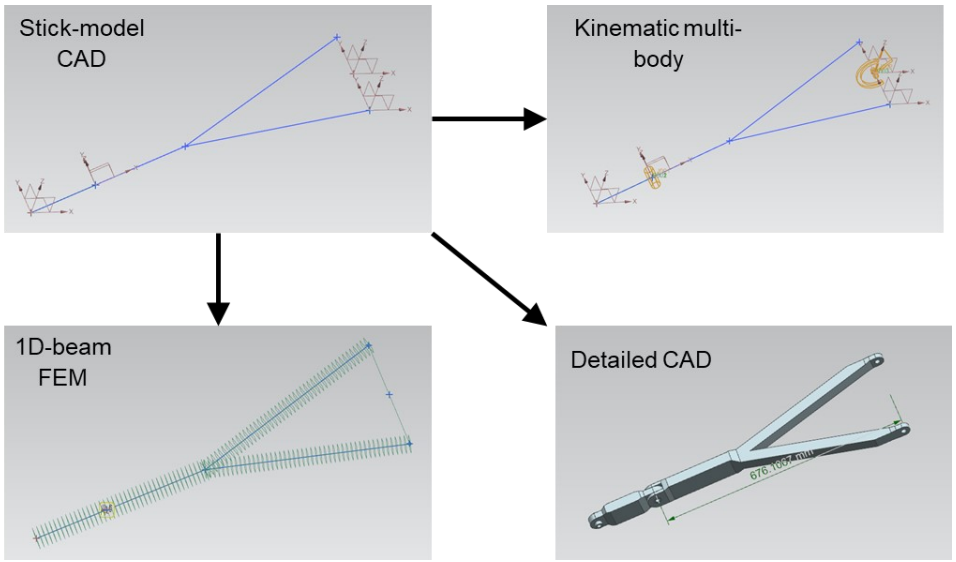



Figure 4: The simple CAD model serves as basis for the FEM, multi-body and detailed CAD model. Templates should be built in this way to minimize modeling (re-)work.

## 4. Example 1: landing gear synthesis

The presented methodology was used for the creation of a tool that enables a non-expert user to select a landing gear architecture, assign models to the components, change parameters, add extra boundary conditions that are not already in the model template, request analysis outputs, and add engineering handbook methods that can use simulation results for calculations. Landing gear models are synthesized for CAD, Finite Element Analysis, and multibody simulation in Simcenter 3D. A screenshot of the tool can be seen in Figure 5. The tool comes with a graphical user interface (GUI) which guides the user in the process and simplifies the provision of user inputs. However, the tool does not limit itself to the synthesis of landing gear models because all design-specific information such as system architecture and template models can be specified in JSON files. Therefore, virtually any scenario can be synthesized if it can be expressed in parametrized template models. The user is guided through the tool in a logical manner by moving through tabs.

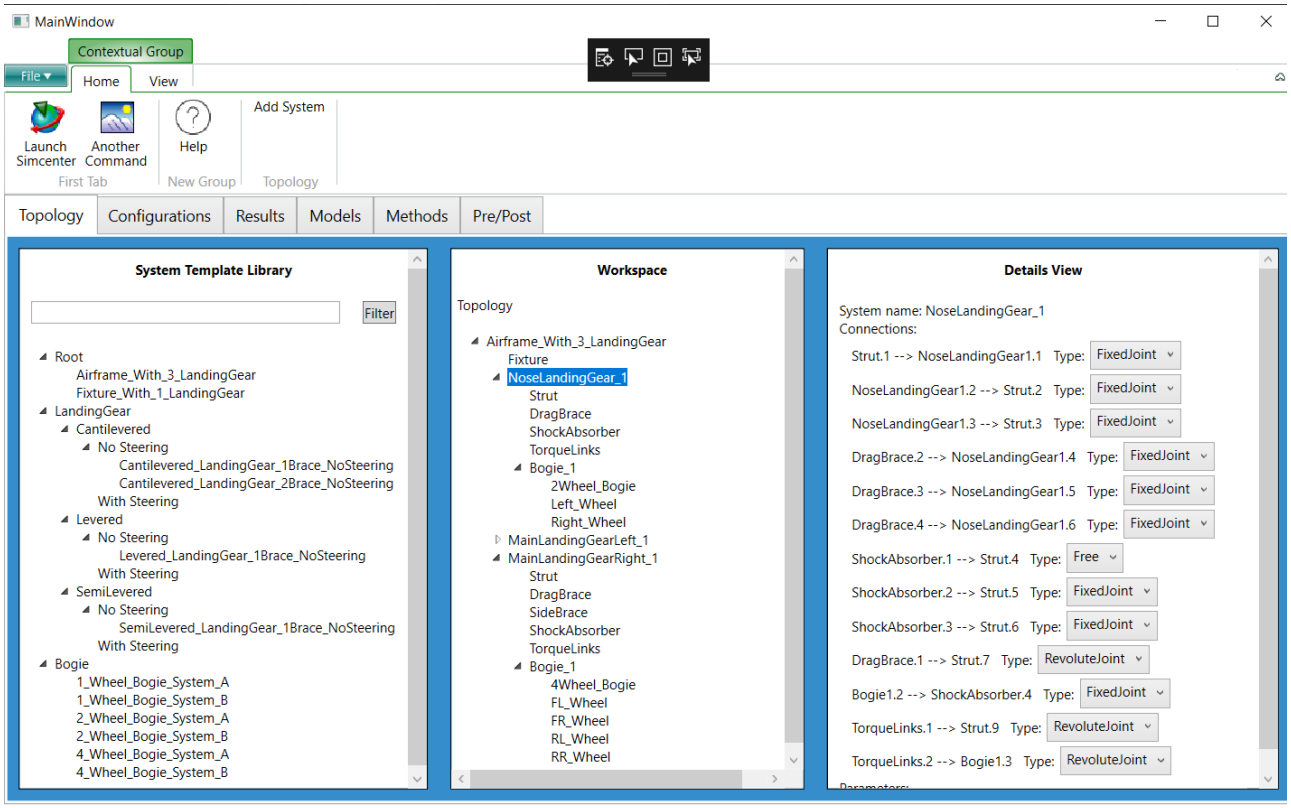



Figure 5 - Screenshot of the tool showing the selection of a landing gear topology on the left pane, the component tree in the middle pane, and details of the selected component on the right pane.

In the first tab, a system topology is built dynamically by dragging and dropping (sub-)system definitions onto the workspace. Figure 6 shows 3 top-level architectures that can be created for a landing gear. Initially, the bogie component is left open for the user to decide to drag a bogie with multiple wheels or a fork with one or two wheels. Note that on the highest level a choice needs to be made between an architecture of a mounting with a single landing gear or an aircraft with multiple landing gear.

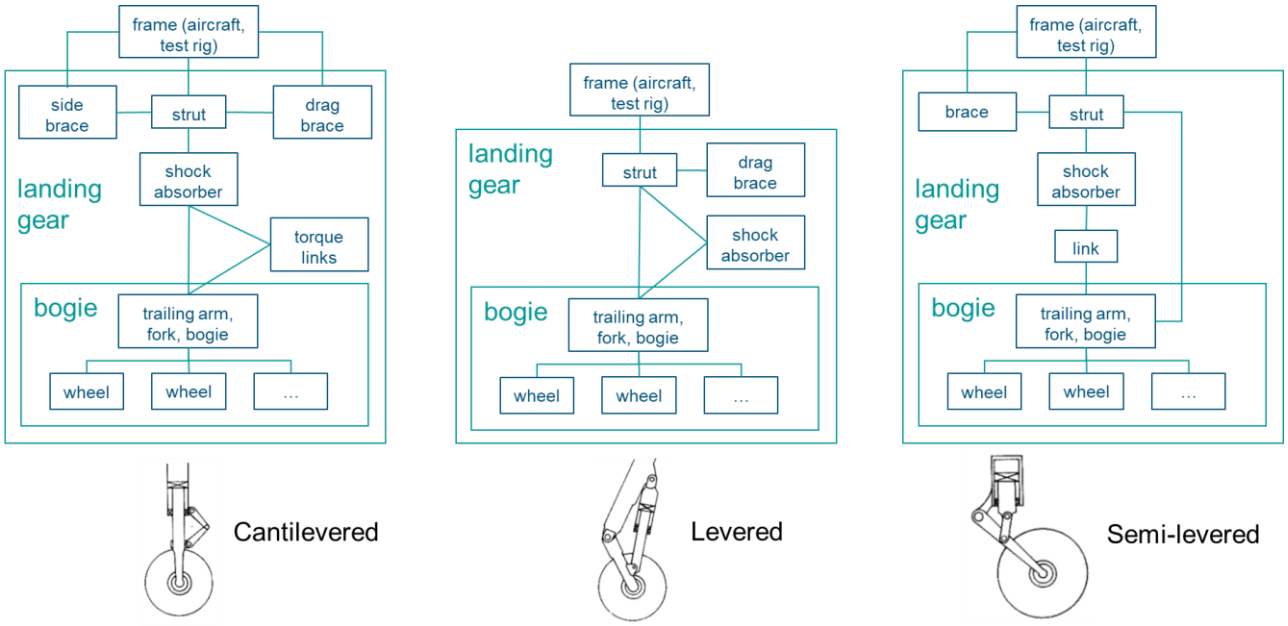
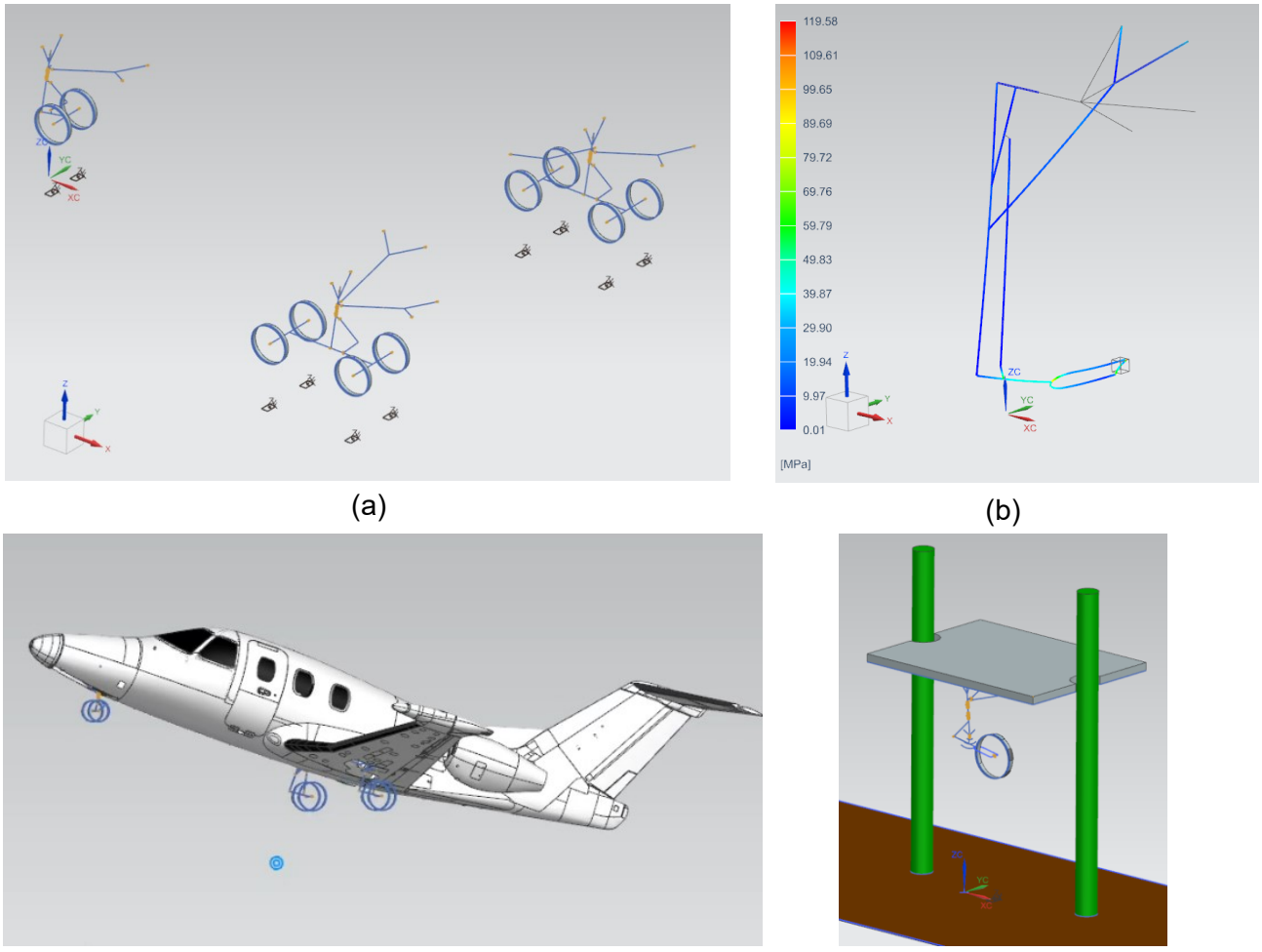



Figure 6: Three landing gear architectures that are available in the synthesis tool. Many variations on these three are possible by adding more or less braces and wheels.

The second tab is used for creating and managing configurations of the topologies. The other tabs serve to define the configuration: assign models to components, link handbook methods and scripts, request results, add load cases and initial conditions.

The tool can accelerate the setup of full system analyses such as FE ground load cases and dynamic multi-body simulations of maneuvers like landing, taxi, and towing. Figure 7 shows the wide variety of scenarios that can be synthesized. Multibody simulation models can be structurally flexible and can express complex dynamics e.g., tire models. Templates for the shock absorber can include a co-simulation with physics simulation software. The focus of the tool is to create the simulation models needed for Design Space Exploration (DSE) and optimization studies.




(a) (b)

(c) (d)

Figure 7 - (a) Synthesized airframe with a nose landing gear and 2 main landing gear in Simcenter 3D Motion. (b) static linear FE analysis of a single landing gear. (c) Landing of a business jet (d) Simulated drop test of a single landing gear with initial wheel spin.

## 5. Example 2: high-lift device synthesis

In this simple example, the kinematic design of the moveable mechanism of a trailing-edge flap is considered. This study aimed to identify a system architecture and a 3D modeling approach that allows the swapping of different deployment mechanisms, see Figure 8. Two types are considered: the dropped hinge mechanism and the four-bar mechanisms.

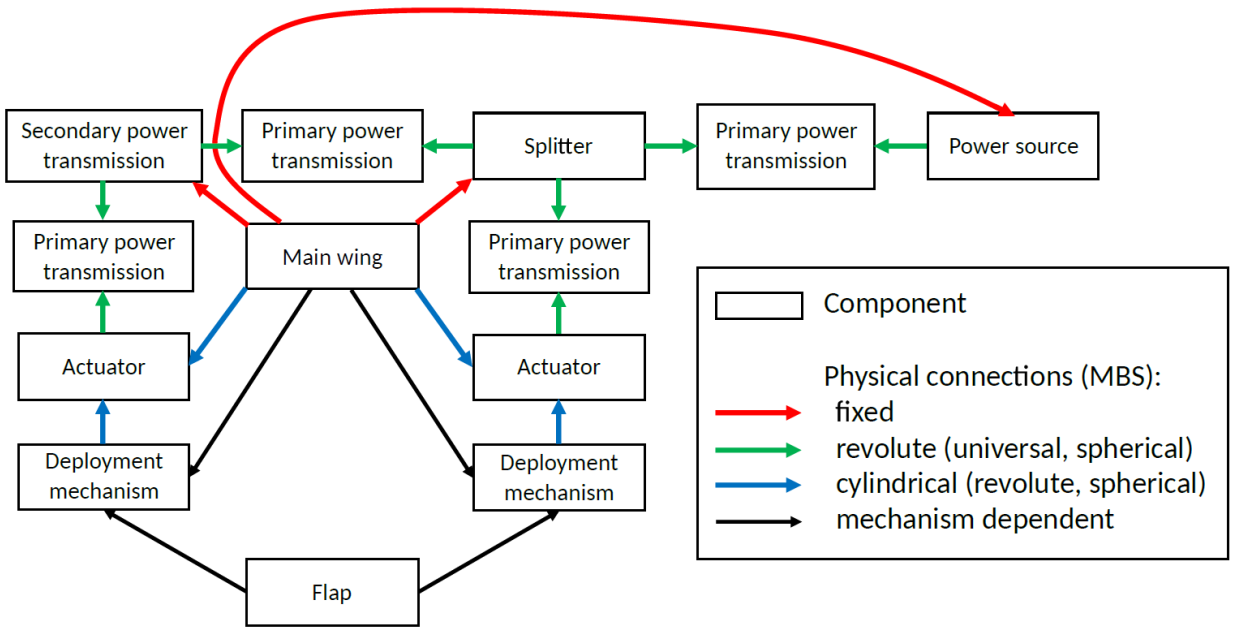



Figure 8 - Schematic of the trailing-edge flap system architecture

The main wing and flap geometry are considered fixed but the positioning parameters of the attachment points for the deployment mechanisms can be changed. The position and orientation of the flap for the different positions are parametrized in the same model file. The task of the deployment mechanism modeler is as follows: create a mechanism where the attachment point with the flap moves on a trajectory through the required flap positions. By having two attachment points to the flap per deployment mechanism, the correct orientation of the flap is guaranteed. When the flap makes a cylindrical or conical motion, there exist construction techniques for the deployment mechanism to match the trajectory. However, when more complex motion of the flap is allowed these techniques break down. On the one hand, one could resort to approximations that often rely on projecting the motion onto a plane. On the other hand, one can give this problem to an optimization algorithm to find the most optimal trajectory.

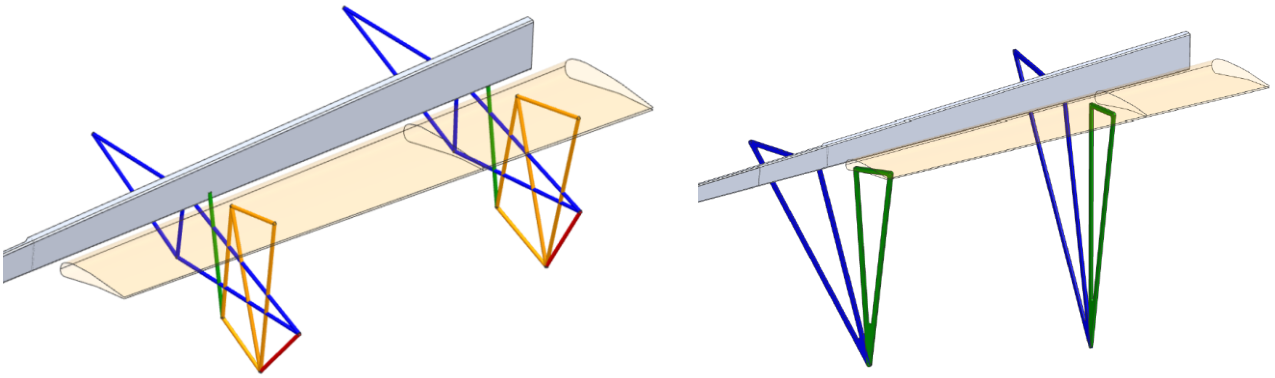


Figure 9 - (left) Optimized design of 4-bar deployment mechanisms and (right) optimized dropped hinge mechanism. The desired take-off and landing positions of the flap lead to unrealistic designs.

The performed optimizations yield usable results for a preliminary design of a high-lift device, demonstrating the usability and the added value of the template-based design, see Figure 9. The architecture-based synthesis approach made it possible to experiment efficiently with different modeling techniques of the deployment mechanisms. In essence, the designer has more time to come up with creative designs by reducing the time needed for linking the CAD models.

## 6. Discussion

People use templates throughout the day for all kinds of activities to avoid repetitive work. Similarly, engineers frequently resort to templates to speed up their jobs. Think about code snippets that software developers use and simulation templates that are becoming more popular for simulation engineers. They all can be seen as shortcuts to obtaining a higher-level objective.

This work contributes to this strategy of using templates to speed up the process of modeling and simulating mechanisms. Experience with the template-based design for 3D systems indicates that a reduction of the modeling effort can be achieved. First, there is the direct advantage of the automatic synthesis of simulation models for the configurations which become significant when there are about 4 or more configurations to be investigated. Second, there are additional advantages that the formalized approach of MBSE brings: better documentation of modeling intent, better collaboration and fewer errors, and a better transfer of knowledge to future projects.

Furthermore, the formal model of the system facilitates extensions beyond synthesis of simulation models. For example, work is ongoing to add assembly assessment functionality to the tool that was presented in this paper. Topological information of the system together with joint types and the addition of assemblability knowledge are very useful information for a code that assesses (dis-)assembly rules. It is expected that this will speed up the exhaustive search for collisions when simulating the assembly of the product.

There is a limit on the design space that a template model can span. This is because a large part of the design intent is not coming forth from parameter values but it results from the feature tree that was created in the CAD modeler or the choice of simulation elements that were added in the simulation environment. Currently, a popular way to circumvent the limitations of parametric models is by writing a script that can generate the model. The script can take input arguments that can influence high-level decisions by loops and conditional statements. Giving more importance to the code can be taken further and one could view Knowledge-Based Engineering (KBE) applications as a manifestation of templating the code. There is even an unexplored hybrid form possible where parametric models are enriched with visual scripting features, e.g. the algorithmic feature in NX.

Clearly, script-based modeling is powerful. However, industry has a clear preference for the parametric CAD modeling approach in combination with a GUI. This preference is not unjust because code can be difficult to understand, especially when the author spent little attention on coding etiquette. For these reasons, advances in code synthesis might enhance engineering design because they can facilitate new methodologies where CAD models are generated from code. It is interesting to note that the key idea of machine learning is to let an algorithm find templates, i.e. patterns, in data.

The methodology as presented is already powerful and in principle, any modular system can be synthesized. Generating models from code would lead to a compression of the model library and allow more complex designs. However, the examples have shown that it was possible to already synthesize a large variety of landing gear simulations from a library of model templates.

## 7. Conclusion

A methodology was presented to enable MBSE for systems that require CAD-based analyses. When taking 3D geometry into account, the design of mechanisms becomes more complex due to physical connections and other geometrical requirements. Most importantly, the component connections needed to be able to exchange geometrical information like points, axes, and curves between components. Model templates use the shared geometrical elements to build on and provide new elements for other models to connect to. This chain of causality can be defined in the abstract system architecture but it is also possible to resolve the dependency chain during synthesis.

The methodology makes the advantages of MBSE available for CAD-based simulations. Templates make it possible to reuse simulation models of components and use software to automatically synthesize them. This leads to a large reduction in modeling effort when there are many configurations that need to be modeled and optimized. Moreover, the formalized approach makes it possible to employ a mathematical solver to generate system architectures.

The methodology is demonstrated by synthesizing CAD, FE analyses, and dynamic simulation models in Simcenter 3D. In the first example, landing gear simulation models are synthesized by a newly developed tool to manage system architectures and set up simulation scenarios that can be synthesized. The tool is open to modification through JSON files to fit in existing design workflows. Virtually any scenario can be simulated if it can be expressed in parametrized template models. In a second example, a trailing-edge flap system is defined and multiple configurations are synthesized to explore different modeling methods for the deployment mechanism. These examples should demonstrate that an MBSE approach for the design of 3D systems is feasible and has many advantages. Future software solutions might remove some of the extra work that currently has to be spent on modeling the templates.

## 8. Contact & Acknowledgment

The authors can be contacted at elias.allegaert@siemens.com, yves.lemmens@siemens.com and emre.ongut@siemens.com.

The research presented in this paper has been supported by the Soft Landings project (Solution for Fast Landing Gear Simulation) and has received funding from the Flemish government through the VLAIO O&O Programme under grant agreement (ref. HBC.2020.2011) and the Assisted DfA project (Automatic Assisted Design for Assembly) through the VLAIO ICON Programme (ref. HBC.2021.0014)

## 9. Copyright Statement



## References

[1] Zeigler B. P., Mittal S., Traore M.K.. MBSE with/out Simulation: State of the Art and Way Forward. *Systems*, Vol 6, no. 4, 2018.

[2] Menu J. and Nicolai M. A framework for automated design, verification, and simulation of electrical power systems for aircraft. *International workshop on aircraft system technologies*, Hamburg, Germany, 2012

[3] J. Holt and S. Perry. *SysML for systems engineering, 2nd edition*. Institution of Engineering and Technology, 2013.

[4] Li, T; Lockett, H and Lawson, C. Using requirement-functional-logical-physical models to support early assembly process planning for complex aircraft systems integration. *Journal of Manufacturing Systems*, Vol. 54, pp. 242–257, 2020.

[5] Promyoo, R., Alai, S. and El-Mounayri, H.. Innovative digital manufacturing curriculum for industry 4.0. *Procedia manufacturing*, Vol. 34, pp. 1043-1050, 2019.

[6] van Gent I, La Rocca G. Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach. *Aerospace Science and Technology*, Vol. 90, pp.410-433, 2019.

[7] Allegaert E, Menu J, Lemmens Y, Dutré S, Ongut E, Wilson W. Architecture-based conceptual design for mechanical systems applied to landing gear. *Proc of 1st Aerospace Europe Conference, Bordeaux, pp. 1-9,* 2020.

[8] Menu J, Nicolai M, Zeller M. Designing Fail-Safe Architectures for Aircraft Electrical Power Systems. *Proc of AIAA/IEEE Electric Aircraft Technologies Symposium (EATS)*, Cincinnati, pp. 1-14. 2018.

[9] Fengnian T., and Voskuijl M. Automated generation of multiphysics simulation models to support multidisciplinary design optimization. *Advanced Engineering Informatics*, Vol. 29, no. 4, pp 1110-1125, 2015.

[10] La Rocca, G., Knowledge-based engineering: Between AI and CAD. Review of a language-based technology to support engineering design. *Advanced engineering informatics*, Vol. 26, no. 2, pp.159-179, 2012.

[11] Gawthrop, P.J. and Bevan, G.P., Bond-graph modeling: a tutorial introduction for control engineers, *IEEE Control Systems*, Vol. 27, no. 2, pp. 24-45, 2007.

[12] Bussemaker, J., Ciampa, P.D. and Nagel, B. SYSTEM ARCHITECTURE DESIGN SPACE MODELING AND OPTIMIZATION ELEMENTS, *Proc of 32nd Congress of the International Council of the Aeronautical Sciences,* Shanghai, pp 1-16, 2021.

[13] Camba, J.D., Contero, M. and Company, P. Parametric CAD modeling: An analysis of strategies for design reusability. *Computer-Aided Design*, Vol.74, pp.18-31, 2016.