Manufacturing x Digital

Final Project Report

Model-Based Systems Engineering for Digital Manufacturing: A Proof-of-Concept	
Principal Investigator / Email Address	N/A
Project Team Lead	Georgia Tech Research Corporation
Project Designation	20-11-09
MxD Contract Number	2020-09
Project Participants	Rolls-Royce North American Technologies, Inc. (LibertyWorks®) Rolls-Royce Corporation
MxD Funding Value	N/A
Project Team Cost Share	N/A
Award Date	01/27/2021
Completion Date	01/27/2022

This project was completed under the Technology Investment Agreement No. W15QKN-19-3-0003, between Army Contracting Command – New Jersey MxD. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Army.

TABLE OF CONTENTS

I. EXECUTIVE SUMMARY

New challenges brought by the increase in complexity and connectivity of new products have accelerated the transition from document-centric to model-centric approaches as a means to address some of the current limitations of PLM. Model-based systems engineering, in particular, by providing a common interface between models, data, knowledge and disciplines, is critical to enable a system-level understanding of the design's performance across domains.

To demonstrate these qualities across the *As Designed* and *As Built* phases of the product lifecycle, this research focuses on the development of a methodology aimed at capturing manufacturing and production system considerations in a model-based environment. In doing so, this research demonstrates how model-based systems engineering allows for the connection and integration of product design, manufacturing and production models and data through an authoritative source of truth (ASoT). In particular, manufacturing/production system model for a minimum viable product (UAV wing) are integrated within an existing digital enterprise to demonstrate the requirements validation and verification process, requirement traceability, and finally the ability to conduct tradeoff analyses between design, manufacturing and production concurrently.

As such, this research contributes to addressing some of the limitations of PLM and in particular the common disconnect between physical and virtual systems. It also demonstrates how model-centric approaches lead to better communication (among tools, disciplines, and decision-makers) to 1) support early design and collaborative decision-making across disciplines and 2) improve the effectiveness of engineering by advancing systems representation and modeling, cross-domain coupling, data—driven trade space exploration and analysis, and collaborative design and support. Finally, this research highlights some of the challenges associated with the realization of a MBSE model as a single source of truth.

II. PROJECT DELIVERABLES

The following list includes all deliverables created through this project. These deliverables are referenced throughout this report and should be accessible on the MxD membership portal in accordance with the rights defined by the Membership Agreement. Specific deliverable types include, but are not limited to, the following items.

- Software modules
- Workforce development materials
- User guides/instillation manuals
- Demos/case studies

Table 1. Project Deliverables

#	DELIVERABLE NAME	DESCRIPTION	FORMAT OF DELIVERY
1	Final Technical Report	A comprehensive, cumulative, and substantive summary of all technical advancements and significant accomplishments achieved during the project. Includes a detailed documentation of the methodology developed and results obtained as well as user resources to run each model developed	Electronic (MS Word and pdf documents)
2	Final Technical Presentation/Demonstration	The final presentation includes a comprehensive, cumulative, and substantive summary of all technical advancements and significant accomplishments achieved during the project, along with a demonstration of the methodology developed.	Electronic (PowerPoint presentation)
3	Modeling Environment	Includes relevant models, data and script developed as part of the effort. In addition, the folder includes ReadMe files for each model (system model in MagicDraw, structural models in NX, manufacturing models in SEER-MFG, production models in Python and Simio)	Electronic
4	Transition Plan	Written plan for successful transition of project outcomes after period of performance including distribution and follow-on efforts for phase(s) 2 & 3. Desired future industry partners are clearly identified with a plan of action for future participation, if/when relevant	Electronic (Word document
5	Educational Impact	Documentation on course/lab module presented to students demonstrating the use of technology to increase awareness in topic area will be provided. Made available to MxD Learn for future educational content.	Electronic (PowerPoint presentation and Word document)
6	Quarterly Technical Review	The quarterly technical review will include a comprehensive, cumulative and substantive summary of all technical advancement and significant accomplishments achieved to date.	Electronic (PowerPoint presentation)

III. PROJECT REVIEW

The growing interest in digital technology has profoundly impacted all disciplines of science and engineering. The integration of digital methods with traditional engineering approaches have led to conversations on how digital techniques can be best utilized to ease the process of designing and manufacturing a product (Gottschall et al., 2018). In recent times, the aerospace industry has faced challenges with regards to the increasing complexity of systems, the large amount of data that is transferred between hyperconnected systems, and the integration of tools and techniques for different types of processes (e.g., tools for managing requirements and for conducting detailed design).

Industry leaders are looking into enabling continuity and integration of the product throughout its lifecycle with the people involved under the traditional idea of a product life-cycle management (PLM). PLM is an approach of facilitating intercommunication between the different departments to ease the process of designing, building, selling, and maintaining a product (NikPakvasa 2009; Radtac). It can be used to simplify, organize, and integrate data and to follow the product through its introduction, growth, maturity, and decline. However, PLM, as it is done today, presents some limitations at the organizational level (Vertex Software, Accessed October 2, 2019). For example, teams often work independently from one another, leading to "silos." This leads to digital and physical boundaries between sub-teams of an organization and inhibits communication of knowledge and data between departments. Traditionally, processes, methods, and disciplines are fundamentally isolated based on each discipline's view of the system of interest (SOI). To address this, utilizing a "systems engineering" approach helps apply a variety of workflows, processes and best practices to create a universal solution which scales with the company, its specializations, and systems (Gottschall et al., 2018). However, a universal solution that can work across teams and organizations, to the team's knowledge, does not yet exist. Furthermore, a PLM-software driven perspective can be limited in adapting quickly to changes (Vertex Software, Accessed October 2, 2019). For example, legacy data is integrated with other data types and aligned to the PLM data model. However, since there is the possibility of incompatible files, additional administrative and curative efforts are required to handle different terminologies and formalisms (Menshenin, Knoll, Brovar, & Fortin, 2020). Therefore, there is a need to improve PLM by providing a trustworthy infrastructure that supports knowledge and data transfer, enables better integration and management of data and models as well as facilitates collaboration over organizational boundaries.

To address the limitations of PLM, Digital Enterprise visions, methods and technologies have emerged to support verification and validation, knowledge-based engineering, and communication and collaborative procedures (Mourtzis, Maropoulos, & Chryssolouris, 2015) during the "As designed" "As Built" and "As Operated" phases of the product lifecycle.

The digital enterprise "connects the digital end physical worlds across a system's lifecycle. The end-to-end digital enterprise will incorporate a model-based approach in a digitally connected environment enabled by advanced technologies to conduct lifecycle activities from concept to disposal (Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018). It enables data and model consistency across teams during the design phase and enables agility by representing the product on high abstraction levels not as a collection of static specification documents (Gottschall et al., 2018). As exemplified by the digital enterprise visions formulated by industry leaders (Boeing (Hatakeyama, Seal, Farr, & Haase, 2018), Rolls-Royce (Figure 1) (A. F. Donaldson et al., 2019) and others), the digital enterprise seamlessly connects virtual and

physical systems to support knowledge and data transfer in physical and virtual realms and enable communication between and within the teams involved in the lifecycle of a product.

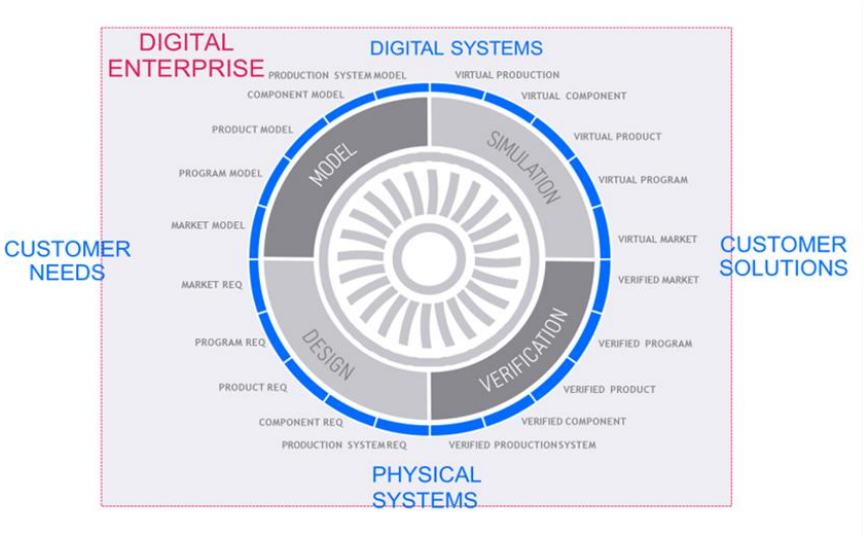


Figure 1. Rolls Royce Digital Enterprise "O" Vision

The critical task in digital enterprise is to digitalize engineering artifacts and enable information sharing, traceability, and accountability across the lifecycle and across domains. The advantages of the digital enterprise are far reaching once understood in these terms. This includes fast infusion of novel digital technologies, unique identification with information traceability, accountability across life cycles, tracing dependency relations among engineering artifacts, model reproducibility and replicability and increasing trust in evaluation of artifacts.

To follow through on the above while keeping in mind that digital enterprise is still a rising domain, there are a few key areas which are crucial (Huang et al., 2020). First, it is important to identify the challenges involved in enabling the transition from traditional engineering to digital engineering. Second, it is important to define core concepts such as digitalization, unique identification, digitalized artifacts, digital augmentation, etc. Third, a big picture of the digital enterprise must be understood in terms of vision, strategy, action and foundation.

Efforts to transition from document-centric to model-centric approaches are underway (Akundi & Lopez, 2021; Singh, 2018) as a means to improve the sharing of product and process definitions, enable the capture of knowledge, improve communication, and enable traceability of requirements and decisions. Hence, as the digital transformation penetrates all phases of the product lifecycle from concept through disposal (Mordecai, de Weck, & Crawley, 2020), Model-Based Systems Engineering (MBSE) has emerged as a critical enabler of Digital Engineering and the design of complex, interdisciplinary systems (Promyoo, Alai, & El-Mounayri, 2019). MBSE is defined as 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” (INCOSE, 2014). Hence, by enabling the communication of system information via descriptive and collaborative models, MBSE ultimately aims to provide a common interface between models, data, knowledge and disciplines to enable a system-level understanding of the design’s performance across domains. By guiding the development of complex systems and products from the early stages of the

lifecycle, it allows for better model and data integration, management and flow (Akundi & Lopez, 2021; Bretz, Tschirner, & Dumitrescu, 2016; Menshenin et al., 2020).

Project Scope

This research focuses on the development of a methodology aimed at capturing manufacturing and production system considerations in a model-based environment, hence allowing for the connection and integration of product design, manufacturing and production models and data through an authoritative source of truth (ASoT).

Digital manufacturing is often viewed as the integration of tools and methods with product definition data to support both design and manufacturing activities (Singh, 2018) and address the challenges that arise during the development of complex products and systems (Promyoo et al., 2019). It is an active area of research, even after decades since its inception in the automotive industry. It is the concept that enables digital and material advancements in much shorter time scales while providing a holistic view of process manufacturing design. Its significance is known to increase with the complexity of the product (Paritala, Manchikatla, & Yarlagadda, 2017). The automotive industry has been quicker at adopting the concepts of digital enterprise and digital manufacturing. Integration and management of the whole lifecycle data in a digital manner is carried out under the headings of product, process and resource (Myung, 2003). By going digital earlier, manufacturers can provide cheaper, faster and better products to their customers. According to (Reinhard Geissbauer, stefan Schrauf, Philipp Berttram, & Farboud Cheraghi, 2020), manufacturing companies are shifting entire supply lines to digital manufacturing and several have moved beyond pilot projects. However, finding adequately qualified employees to handle digital manufacturing is one of the major difficulties. The integration of man and machine is another key challenge. While a digital enterprise can provide a relief from repetitive tasks, it can be difficult to secure funding and engage stakeholders by moving beyond limited proof of concepts. Pilot programs can help in identifying project-specific issues and finding task-specific solutions.

The research discussed herein builds on the following past efforts: 1) the formulation, development and implementation of a Digital Enterprise to enable the sharing of models, data and analyses across the “As Designed”, “As Built”, and “As Used” phases of a product’s lifecycle, and 2) the implementation of a methodology that integrates parametric, physics-based structural design and aircraft performance models alongside process-based costing and discrete-event simulation to facilitate affordability-based design exploration during the early stages of the design. Hence, as part of this research manufacturing/production system model for a minimum viable product (UAV wing) are integrated within the aforementioned digital enterprise (data, models, etc.) to demonstrate the requirements validation & verification process, requirement traceability, and finally the ability to conduct tradeoff analyses between design, manufacturing and production concurrently.

Research Objectives

This research integrates the manufacturing and production modeling components of the framework developed in (Siedlak, Pinon, Schlais, Schmidt, & Mavris, 2018) with the RFLP (Requirement, Functional, Logical and Physical) approach and MBSE models developed in (Kargin et al., 2021) to satisfy the following objectives:

At a capability level:

- Utilize the requirement-Functional-Logical-Physical (RFLP) approach to track requirements throughout lifecycle.
- Develop architectures and frameworks for Digital Manufacturing.
- Illustrate the integration of system models with other analytical tools.
- Develop the link between manufacturing capabilities and design.

At a programmatic level:

- Address some of the limitations of PLM and in particular the common disconnect between physical and virtual systems.
- Provide better communication (among tools, disciplines, and decision-makers) to support early design and collaborative decision-making across disciplines.
- Improve the effectiveness of engineering by advancing systems representation and modeling, cross-domain coupling, data—driven trade space exploration and analysis, and collaborative design and support.
- Leverage on the knowledge infrastructure built during past efforts to develop a greater focus on the digital manufacturing aspects of Rolls-Royce digital enterprise vision.
- Provide an opportunity to demonstrate 1) capacity for reuse of models, data and knowledge, and 2) knowledge transfer through the sharing and joint access of models and data, and the ability to learn around common tools and representations.

At the enterprise level:

- Contribute to increasing the awareness of MBSE benefits and value.
- Address the recognized need to develop a workforce with knowledge and skills relevant to Digital Engineering and MBSE.

Major Tasks

The major tasks are provided in Table 2.

Table 2. *List of Major Tasks*

#	MAJOR TASKS
1	Knowledge transfer: familiarization with existing MBSE architecture and models
2	Requirement identification and use case definition
3	System functionality identification and definition of derived requirements
4	Modeling: structural analysis, manufacturing processes of interest and production system
5	Integration of modeling and analytical capabilities within MBSE environment
6	Verification and validation of requirements via simulation through demonstration of trade-offs between design, manufacturing and production considerations

The Technical approach and specific tasks are discussed in Section IV. Section V describes the implementation of the approach in detail while Section VI discusses the capabilities developed and knowledge gained as a result. Section VII concludes as to the benefits of this research and provides avenues for enhancements and future work.

IV. TECHNICAL APPROACH

The sections below describe the methodology and approach used to address the aforementioned objectives by focusing on the creation of a Minimum Viable Product (MVP).

The goal of this investigation was to build on the existing digital enterprise proof-of-concept to demonstrate the seamless integration of design, manufacturing, and production. The methodology is built to break down the steps of the implementation.

Rolls-Royce's Digital Enterprise Vision

Figure 2 describes Rolls-Royce's Digital Enterprise vision for the three following phases: As Designed, As Built, and As Used. The goal of this vision is to improve upon traditional PLM processes by extending each phase into both physical and virtual spaces through the Digital Thread, where the Digital Thread is defined as the "linked authoritative information pertaining to a process, product, or system enabling the capture and communication of knowledge and decision-making throughout its life cycle" (AIAA Digital Engineering Integration Committee, To be published in 2022).

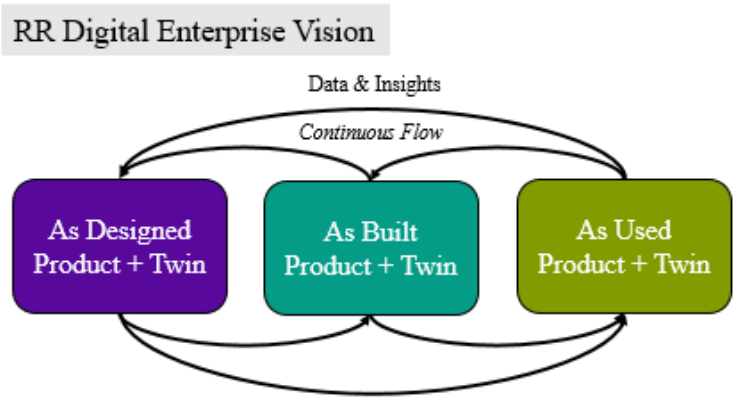


Figure 2. Rolls-Royce's Digital Enterprise Vision

A detailed explanation of each phase is provided below.

As Designed

The *As Designed* phase captures the structural and CAD models as well as requirements. In this phase, the structure of the physical product and its digital twin is created. Additionally, analysis of structural capability and meeting of dimensional as well as capability requirements are seen in this phase.

As Built

The *As Built* phase captures all of the physical and digital elements that contribute to the manufacturing and production of the product. In this phase, validated and calibrated models are updated based on experimental information from Test and Evaluation (T&E).

As Used

The *As Used* phase captures all physical assets as well as digital methods that simulate the behavior of the actual physical product. The physical product provides real-life data from operations and the digital twin provides simulation results for the most optimal operational settings in future missions.

As mentioned in Section III, this research builds on past efforts conducted by GT/ASDL and sponsored by Rolls-Royce. The following section describes the work conducted then to provide additional context to this research.

Review of Previous Efforts at GT/ASDL

In 2019/20, an ASDL team advised by Rolls Royce developed an approach to support the definition and implementation of a Digital Enterprise proof-of-concept across the *As Designed*, *As Built* and *As Used* phases for a small Unmanned Aerial Vehicle (Figure 3). The proof-of-concept leveraged the concepts of Digital Twin and Digital Thread, Model Based Systems Engineering (MBSE), the Requirements-Functional-Logical-Physical (RFLP) approach, and a graph database (Neo4j) as the data infrastructure. The digital enterprise created by the team can be seen in Figure 4 and is discussed in detail in (Kargin et al., 2021).

Figure 3. UAV used as part of 2019/220 Efforts

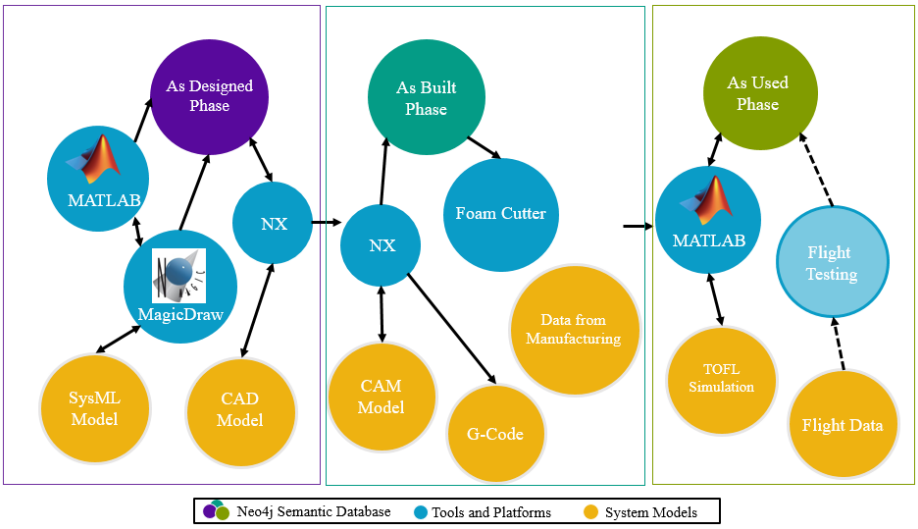


Figure 4. Digital Enterprise Proof-of-Concept

The team's goals and accomplishments are summarized in the Table below.

Table 3. Past Research Goals and Accomplishments

Goals	Methods
Create a design model and integrate design analysis/simulation.	Created MBSE models integrated with analysis capabilities in MATLAB and parametric geometry in NX.
Model the manufacturing process and validate against manufacturing data.	Created a manufacturing simulation in NX; corresponding tool path was used in actual manufacturing.
Connect data flow between all phases. Expose the data infrastructure to end-users.	Created a data infrastructure that saved the gold layers and connected each portion of the infrastructure. Developed easy-to-use desktop app for seamless model-app handoff.
Develop and operational digital twin and connect it to design models using MBSE.	Performed simulations of flight with Fixed Wing in MathWorks, aerodynamics with AVL, propulsion with ground test data, and integrated all with MATLAB. Calibrated battery model using flight data

Knowledge Transfer

To build on past efforts requires that knowledge be transferred to the new team of researchers undertaking the efforts discussed below. In particular, the largest driver of this step is learning the MBSE-specific tools used: MagicDraw (MBSE architecture) and Neo4j (enabler of the digital thread that connects all the information passed from the different models and constituent programs, like NX and MATLAB). Similar to learning a new programming language, the knowledge transfer step had a somewhat steep learning curve.

To facilitate this process, the past team walked the new team through the MBSE models and architecture, the Neo4j graph database, and the MATLAB scripts that they developed. Tutorials and demonstrations were also recorded as part of virtual meetings and constitute a library of knowledge that can be accessed and reviewed as needed. In addition, as skills are best learned through practice, the new team completed a number of SysML tutorials made accessible to them by GT researchers.

Definition and Identification of Minimum Viable Product

Building a complete and scalable digital manufacturing infrastructure/environment is a rather challenging endeavor that includes many dynamic models. Given the time constraint of this research, a minimum viable product was chosen to demonstrate the integration of design and digital manufacturing. A minimum viable product (MVP) is a concept from lean that stresses on the impact of learning in a new product development (Agile Alliance). It involves the prioritization of product requirements to deliver core functionalities of a simple system. The MVP is minimum because it includes a basic set of features and capabilities for the overall solution. It is viable because it is able to work and function as intended. It is a product because it is something tangible that customers can see and use.

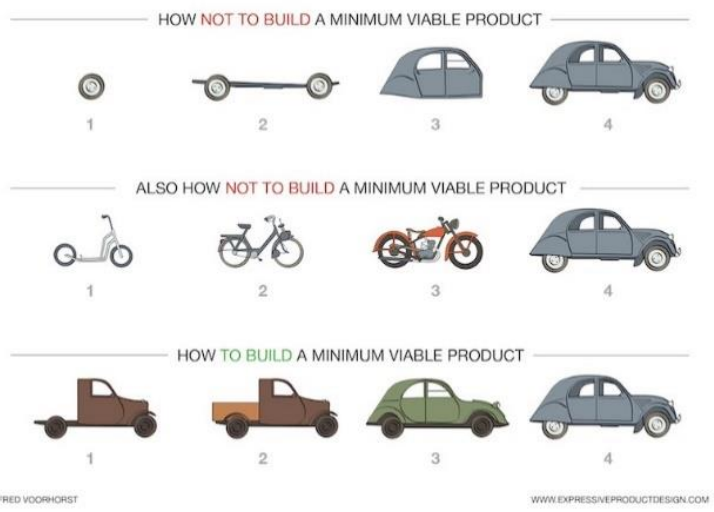


Figure 5. How to Build an MVP (Agile Alliance)

An MVP should have the ability to be scaled up and upgraded to a more complex system. Most importantly, it provides a digital foundation that can be enhanced as capabilities and technologies become more mature and tools/software become available.

In this research, the fully complex system is a representation of successful integration of the design and digital manufacturing processes of a fully engineered product. Following specific criteria discussed in the next section, the team selected the wing as the MVP (or use-case) for this research.

MVP Determination

A component of the aircraft had to be selected as the MVP for this research. The team leveraged the AIAA Design, Build, Fly (DBF) competition requirements as well as the aircraft used in past efforts to determine the use case. This fixed wing aircraft was the winning design from the 2016 AIAA DBF competition. The structural design and CAD packages were also available to the team.

Four alternatives were generated and benchmarked against four different criteria. The four alternatives are the wing, fuselage, landing gear, and empennage. The selection criteria are shown in Table 4.

Table 4. Subsystem Selection Criteria

Physical Component Complexity	Available understanding of the component design.
	Understanding of complexity associated with the structural analysis and simulation.
Physical Component Flexibility	Room for changes: how much can the component be modified for improvement (e.g., change in material, change in the design/geometry).
Component Manufacturing	Familiarity with modeling and simulation of the manufacturing process (e.g., modeling of milling machines).

	Availability of manufacturing/production requirements.
	Cost: how much will it cost to build the component.
Component Digital System Characteristics	Availability of tools for modeling and simulation of component design.
	Availability of modeling and simulation tools for manufacturing.
	Team familiarity with software for modeling and simulation of manufacturing process and structural analysis.
	Ability to demonstrate requirement traceability: Potential impact of geometry/manufacturing process on performance requirement.

Table 5 shows the final benchmarking matrix. To convey the logic followed to assign color intensities to it, the landing gear and the wing will be described in greater depth below.

Landing Gear

The landing gear is an aircraft component for which the detailed design takes about a week to complete and approximately one day to manufacture. The total cost of manufacturing is $40 and the process is as follows.

- A strip of metal is bent into a shape using a sheet metal (this is simulated in NX Computer-Aided Manufacturing (CAM) and done in actuality) (Figure 6).
- Holes are drilled.
- NX CAM defines the tool path, and this is executed in the physical product as well.

Figure 6. NX CAM Sheet Metal Bending Modelling

Though the design of the landing gear (Figure 7) is relatively simple, and the manufacturing process is relatively easy, there are also some challenges associated with this part. Due to its simplicity, it does not allow for flexibility in the design or manufacturing process. Therefore, testing the robustness of the methodology is not as feasible as it could be if another physical component were used as MVP. Additionally, there is little literature on the design and requirements to be tested against are non-existent.

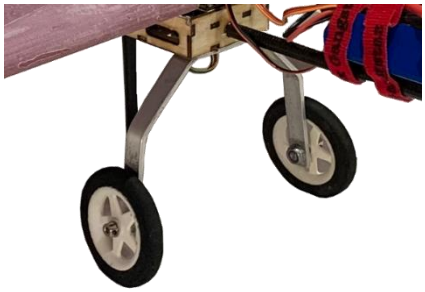


Figure 7. Year 1 Landing Gear

Wing

Another aircraft component is the wing for which the detailed design takes about a week to complete and about three days for manufacturing. The manufacturing time of the wing is longer than the landing gear because it is a larger component and the curing time for the epoxy used for it is longer. It costs approximately $50 and the manufacturing process is as follows.

- The wing and spar hole are cut from a solid block of polystyrene foam (as is illustrated in Figure 8).
- The wing is covered in 2-part epoxy to strengthen it.
- The ailerons are cut out and wrapped around with fiber glass.
- Holes are cut into the wing to attach servos.
- The wing and fuselage are joined together with epoxy.

The benefits of choosing the wing (see Figure 9 for the model as designed in NX) as the use case are that requirements are readily available, it has a moderately complex design, and it is relatively easy to collect data from it. For example, strain gauges could be added to its upper surface to measure strains and stresses so they could be verified against the *As Built* phase. Additionally, because different processes and materials are needed, there is room and flexibility in the manufacturing process to integrate new capabilities. However, there are challenges associated with using the wing as a use case as well: many different materials (e.g., carbon fiber and epoxy) result in different manufacturing processes. Furthermore, the cost of manufacturing the wing is slightly higher than the landing gear.

Figure 8. Cutting Wing from Foam

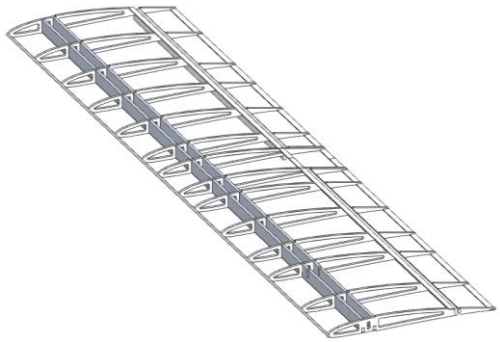


Figure 9. Wing Model on NX

The alternatives are benchmarked based on the criteria in Table 4 and Table 5. Some criteria are more important than others, such as room for changes and availability of requirements, since they need to be carried through the research and validated towards the end of the project. The wing was chosen as the MVP since it met the set criteria the best.

Table 5. Proof of Concept Selection Benchmarking

Criteria		Landing Gear	Wing	Fuselage	Empennage
Physical Component Complexity	Understanding of Component Design				
	Understanding of Analysis and Simulation				
Physical Component Flexibility	Room for Changes				
Component Manufacturing	Modeling and Simulation Expertise				
	Availability of Requirements				
	Cost				
Component Digital System Characteristics	Availability of M&S Tools for design				
	Availability of M&S Tools for manufacturing				
	Team familiarity with modeling and simulation software				
	Demonstrate requirements traceability				

1 3 5

Determination of Tools and Software

Next, the tools and software used to model and simulate the wing as the proof of concept needed to be benchmarked and chosen. The required tools were broken down into MBSE, Requirements management, CAD modeling and structural analysis, manufacturing and production modeling and simulation and Digital Thread. The tools chosen were benchmarked against some key criteria, as shown in Table 6.

Table 6. Software and Tools Trade Study

Disciplines	Tools/ Platforms	Used by RR	License Availability to the Team	Community Acceptance	User-Friendliness	Level of integration	Plug-and-play	Learning Curve	Used in DEAL GC Year 1
MBSE	MagicDraw								
Requirements	DOORS								
	Cameo								
CAD and Structural Analysis	3DX								
	NX								
	Ansys								
Digital Manufacturing/ Production	SEER-MFG								
	SEER-3D								
	TruePlanning								
	SimPy								
	Simio								
Digital Thread	Neo4j								

From this benchmarking exercise, the following software and tools were chosen – MagicDraw for MBSE, Cameo for Requirements, NX for CAD and structural analysis, SEER-MFG for manufacturing modeling, SimPy and Simio for production modeling, and Neo4j as the digital thread.

Approach and Methodology

After determining which software and tools to use for the purposes of this research, the methodology and approach steps were developed.

An overview of the methodology proposed for integrating the existing system model, parametric structural design, process-based costing, and discrete-event simulation to facilitate trade studies during the early stages of the design is presented below.

- **Step 1 – Define the Use-Case:** Wing.
- **Step 2 – Identify the tools and platforms to be used:** Neo4j Graph Platform, MagicDraw, Cameo, NX CAE, NX CAM, SimPy, Simio, and SEER.
- **Step 3 – Define the technical baseline** (existing software, research, technology, etc.) from which technical development will begin.
- **Step 4 – Leverage the Digital Enterprise concepts** to enable the sharing of models, data and analyses across the *As Designed* and *As Built* phases of a product's lifecycle. In particular, the MBSE environment enables:
    - The sharing of aircraft design and manufacturing properties within a system model.
    - The verification of requirements at each abstraction level.
    - The concurrent development of design and production system and the integration of analytical tools and visualization capabilities to support informed decision making.

Additionally, the Requirements, Functional, Logical, and Physical (RFLP) approach, shown in Figure 10, is used to leverage software or tools to model the manufacturing and production processes. Using this approach helps augment the capabilities, data infrastructure, SysML models, etc., that were developed previously.

Requirements	Functional	Logical	Physical
• Identify design requirements • Develop manufacturing requirements • Develop production system requirements (i.e. factory layout, takt time) • Verify and validate requirements through test cases.	• Identify system functionality to enable manufacturing processes to occur. • Derive requirements from identified functions.	• Determine manufacturing processes to satisfy system functionality.	• Use Design of Experiments (DoEs) to parameterize the production system model. • Interface with manufacturing simulation tools.

Figure 10. RFLP Approach

The methodology and RFLP approach enable the implementation of the digital manufacturing vision. This includes six models of interest – requirements, design, structural, manufacturing, production, and operation. For the purpose of this study, only the models in the *As Designed* and *As Built* phases are considered. A representation of the framework is given in Figure 11. The *As Built* phase represented in Figure 11 focuses on the connection between the system model and the structural, manufacturing, and production models. In particular, requirements are collected from different sources such as the DBF competition, FAA regulations, and customer requirements. Then, the requirements are flowed into MagicDraw to develop use cases for the design model. The design model is then connected with the structural, manufacturing, and production models through SEER, Simio, and SimPy. The model can then share aircraft design and manufacturing/production properties within the system model. The sharing of models, data and analysis ultimately facilitate trade studies between design, manufacturing and production.

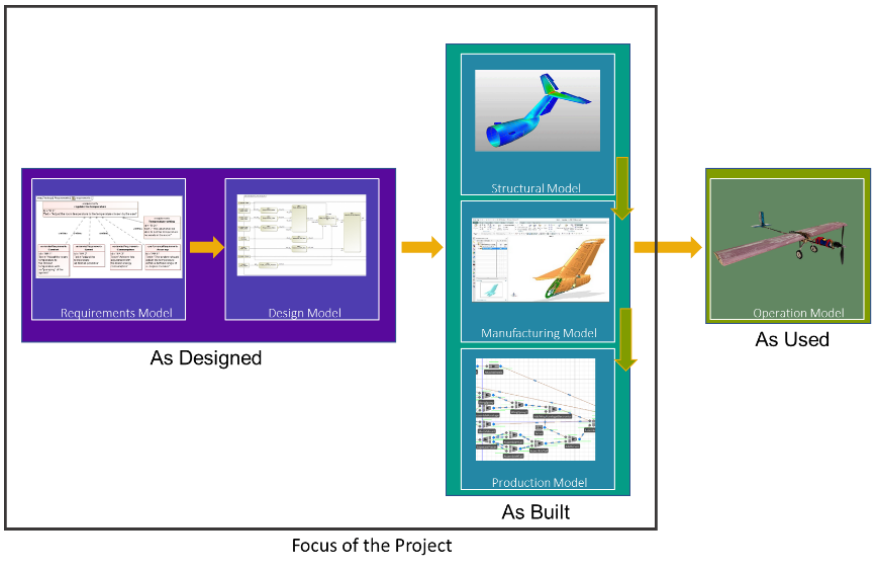


Figure 11. *Representation of the Framework*

The following section discusses the implementation of the aforementioned framework. Additional information and detail pertaining to the installation and development of the various models as well as their integration can be found in Appendix.

V. IMPLEMENTATION

This section details the technical work behind the implementation of the MBSE, structural, manufacturing, and production models for the *As Designed* and *As Built* phases of a product's lifecycle. Additionally, the integration of external tools with the MBSE environment, MagicDraw, is demonstrated. Finally, the Data Infrastructure is presented.

As Designed

The *As Designed* phase focuses on the requirements and models of the overall system structure. The design analysis is coupled with authoritative system models to decompose and describe the system to its full extent. Then, the system structure is verified against the requirements at all stages to ensure that intended functions are fulfilled. The validation process is enabled by the connection between the design models and requirements highlighted in this segment. The new additions and changes reflect the addition of digital manufacturing to the existing system model.

In particular the team added the design of the wing as well as the structural analysis that ensure meeting structural requirements. The data from the CAD model of the wing as well as the finite element analysis were integrated to the overall system structure to enable the passing of data down to the *As Built* or manufacturing and production portion of the work.

Architectural Framework and Context Diagrams

An architecture framework is an encapsulation of a minimum set of practices and requirements for artifacts that describe a system's architecture (The MITRE Corporation) within a specific domain of application and/or a community of stakeholders. Models are representations of how objects in a system fit structurally in and behave as part of the system.

For this system, the model infrastructure is housed within the data infrastructure, as defined by the use case diagram. Within the model infrastructure, the inputs, outputs, and connections between each model are shown (Figure 12).

For example, from the CAD model to the structural model, wing properties are passed along. This includes information such as the wing geometry, structural elements, material, loads, and boundary conditions. The purpose of this view is to show the inputs and outputs that are being transferred between each model.

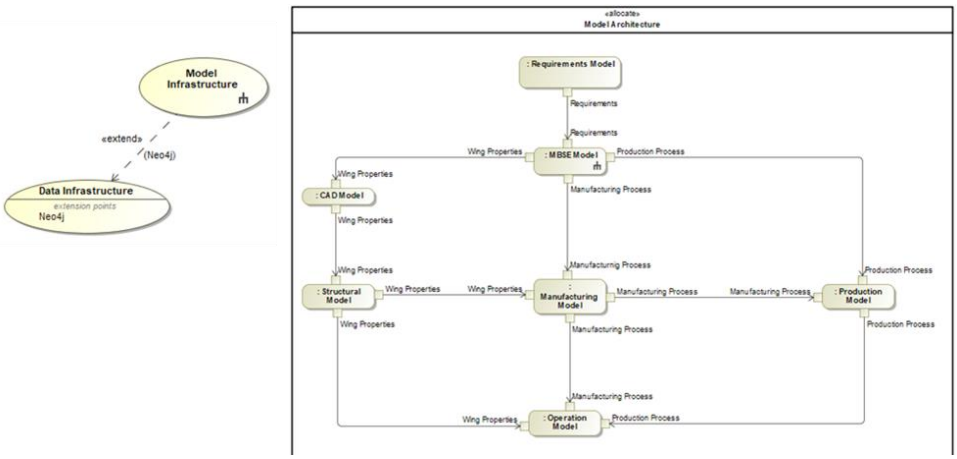


Figure 12. Framework Architecture

Furthermore, a block definition diagram is used to define the parts (in this case, the tools and software) that each model will be using (Figure 13). The CAD and Structural models use NX, the manufacturing model uses SEER-MFG, and the production model uses SimPy and Simio. The use of those two modeling environments to develop production models is motivated by the desire to illustrate the integration of the system model with a variety of tools and languages.

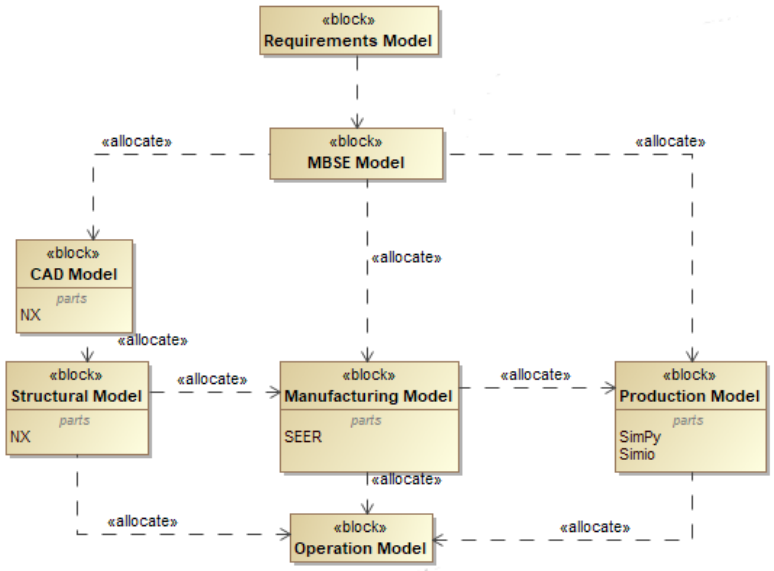


Figure 13. Block Definition Diagram

Lastly, an overall airplane program structure is defined using a block definition diagram as well. This includes the airplane, the production system, and the data system (Figure 14). This view shows the breakdown structure of how the specified use case (the wing) and each model fits into an airplane program.

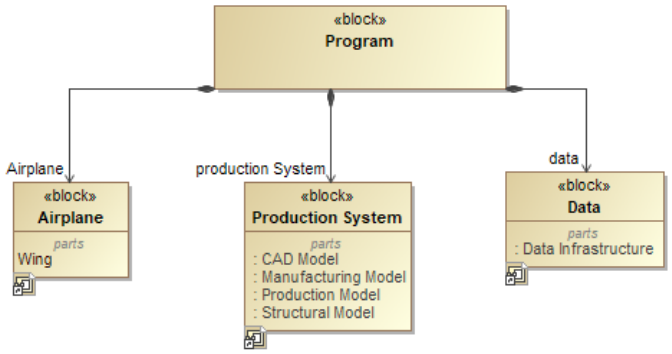


Figure 14. Program Structure

Stakeholder Needs

Stakeholder needs and requirements represent the views at the enterprise operations level. This includes the users, customers, regulatory agencies, etc. The primary user requirements were extracted from the Design, Build, Fly competition ruleset as well as regulatory agencies such as the Federal Aviation Administration (FAA). The requirements of interest developed as part of the past efforts (Kargin et al., 2021) were modified to conform to the SysML syntax. The requirements table is shown below.

1	SN1 Stakeholder Need 1	Grouping stakeholder need (may be left empty)
4	1.0 Banned Configurations	The aircraft shall be any configuration except rotary wing or lighter- than-air.
5	2.0 Structural Integrity Definition	The aircraft shall have strong structural integrity such that no components are dropped during flight.
6	3.0 Takeoff Assistance	The aircraft shall take-off using energy from on-board propulsion battery pack(s).
7	4.0 Propulsion System Definition	The aircraft's motors shall be a commercial brush or brushless electric motor.
10	5.0 Propeller Definition	The aircraft shall have the capability to change the propeller diameter/pitch for each flight attempt.
11	6.0 Battery and Fuse	The aircraft's motors and batteries shall be limited in current draw.
16	7.0 Pilot Verification	The Aircraft and pilot shall be AMA legal.
17	8.0 Physical Inspection	The aircraft shall be physically inspected before flight.
18	8.0.1 Component Security	The aircraft components shall be adequately secured to the vehicle..
19	8.0.2 Structural Integrity	The aircraft's propeller shall have structural and attachment integrity.
20	8.0.3 Wiring	The aircraft's electronic wiring shall have adequate wire gauges and connectors.
21	8.0.4 Radio Check	The aircraft shall have the capability to perform a radio range check, motor off and motor on.
22	8.0.5 Control Check	The aircraft shall have controls that move properly.
23	8.0.6 Payload Integrity	The payload shall have structural integrity.
24	9.0 Structural Verification	The payload shall have to capability to verify the structure.
28	10.0 Production Aircraft Requirements	The Production Aircraft shall be used for operations.

Figure 15. Requirements Table

With the focus of this research effort on digital manufacturing and production, additional requirements were developed to address stakeholder needs. These requirements relate to

manufacturing, production, and structural (design) processes. An external Excel file (Figure 16) was created and synced into MagicDraw to allow for continuous traceability of changes. The file includes information regarding the requirement ID, requirement text, rationale for the requirements, and the model that it corresponds to.

#	Text	Rationale	Related Requirements
DI01	The Data Infrastructure shall connect all models to allow for user interfacing.	The user needs to be able to interface with all of the models from the MBSE environment. The connection between the models is critical to developing the digital thread.	
MS01	The Manufacturing systems shall manufacture wings using metal manufacturing techniques.	Wing will be made of metal materials	
MS02	The Manufacturing System shall produce X wings every month.	The factory needs to be able to produce a specified amount of wings every month to meet customer demands and stakeholder needs. This number is set by the company and can scale up once operations ramps up.	
MS03	The Manufacturing System shall manufacture a wingspan that does not exceed 5 feet.	Design, Build, Fly Requirement	
MS04	The aircraft take-off gross weight with payload shall be less than X lbs.	In order for the weight to not overpower all of the other forces on the aircraft, the take-off gross weight needs to be constrained.	
MS05	The aircraft shall use commercially produced propeller blades.	Design, Build, Fly Requirement	
MS06	The Manufacturing System shall have a throughput of at least 3.	Derived requirement related to MS02	MS02

***Figure 16**. Structure of the External Excel File*

#	△ Name	Text
1	R DI01	The Data Infrastructure shall connect all models to allow for user interfacing.
2	R MS01	The Manufacturing systems shall manufacture wings using metal manufacturing techniques.
3	R MS02	The Manufacturing System shall produce X wings every month.
4	R MS03	The Manufacturing System shall manufacture a wingspan that does not exceed 5 feet.
5	R MS04	The aircraft take-off gross weight with payload shall be less than X lbs.
6	R MS05	The aircraft shall use commercially produced propeller blades.
7	R MS06	The Manufacturing System shall have a throughput of at least 3.

Figure 17. Digital Manufacturing Requirements

Use Cases

Stakeholder needs are further refined with use cases and use case scenarios. Similar to the stakeholder requirements shown in Figure 17, work has been previously done to create functional use cases. The goal was to provide measurable value to the user by creating scenarios about how the system interacts with the user and other systems. The use cases modeled for the UAV were: Start, Takeoff, Climb, Flight Control, and Landing (Figure 18) and the analysis was conducted through activity diagrams.

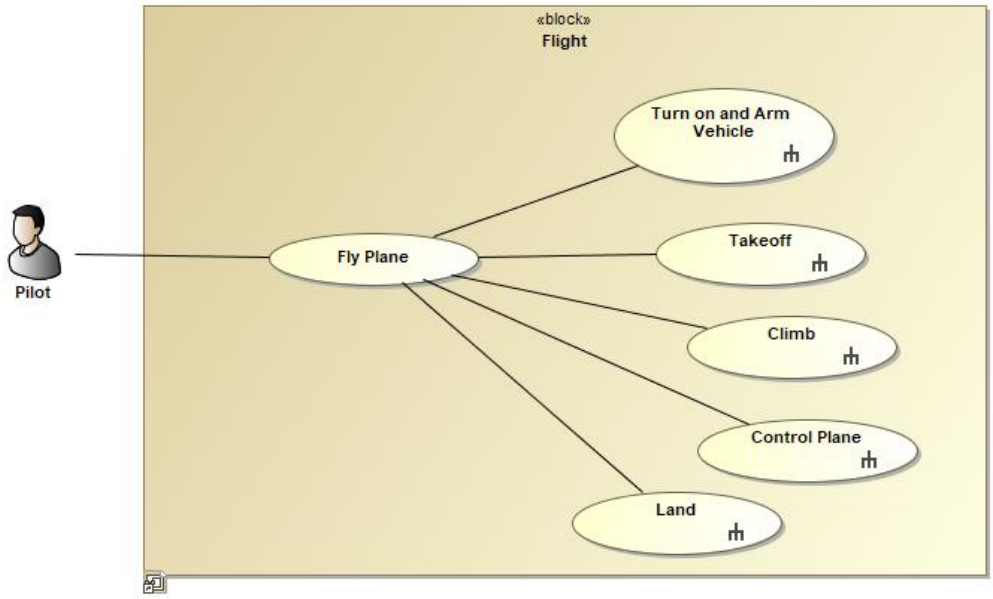


Figure 18. Use Case Diagram

The activity diagrams (Figure 19) for these use cases were modified to ensure that all required elements such as start and stop nodes, signals, allocated swimlanes, etc. were present and also better organized.

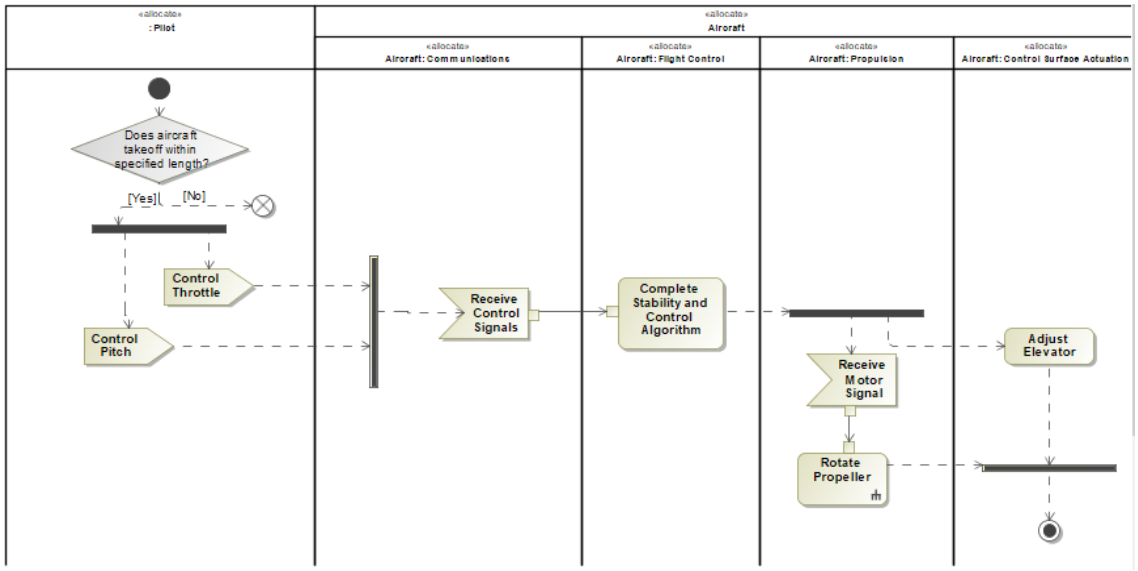


Figure 19. Example Activity Diagrams

In addition to the uses cases that relate to the phases of flight, new use cases relating to the design, manufacturing and production were created, as shown in Figure 20.

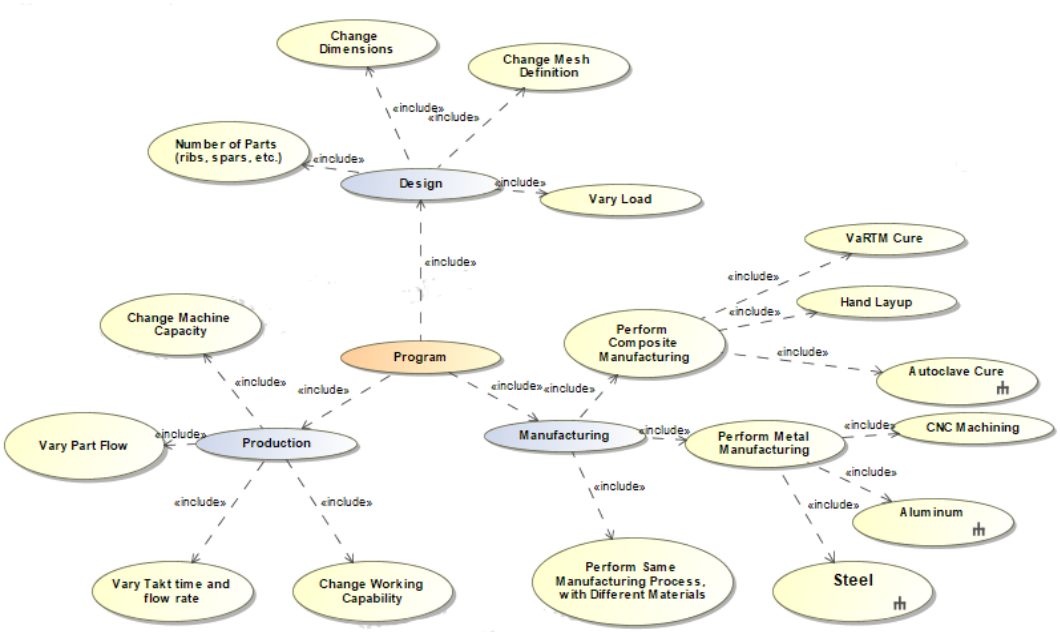


Figure 20. Design, Production, and Manufacturing Use Cases

A description of each use case, its purpose, as well as which requirements would be satisfied through the development of the use case are shown in Table 7 through Table 9.

Table 7. Design Use Cases

Use Case	Description	Purpose	Satisfied Requirements
Vary Number of Parts (ribs, spars, etc.)	The ribs, spars, stringers, are principal structural parts of an aircraft's wing. This looks at how changes in the part quantity change the overall structural integrity.	A change in the number of parts, based on stakeholder needs or requirements, can change the overall structure of the wing.	DR02 – Principal structure
Change Dimensions	There is a range of dimensions that the wing can have.	This determines what the best overall dimensions of the wing should be for the structural design. These outputs will provide insight into the manufacturing and production models.	DR03 – Capability to change parameters DR05 – Output results
Change Mesh Definition	Meshing involves weaving open fabric of string, rope, or wire together at regular intervals.	Based on variations in the CAD model, the mesh definition can be updated accordingly.	DR04 – Update Mesh
Vary Load	The wing structure needs to be able to remain its structural integrity under a certain load factor.	After completing structural analysis, it needs to be confirmed that the wing structure is able to withstand a certain load. Otherwise, there is means for structural damage.	DR01 – Load factor

Table 8. Manufacturing Use Cases

Use Case	Description	Purpose	Satisfied Requirements
Perform Composite Manufacturing	Composite manufacturing involves the combination of two or more dissimilar materials that are used together to combine best properties. It consists of three possible facilities and tools- autoclave cure, hand layup, and VaRTM cure	The process of composite manufacturing involves more complex steps that would be useful in documenting. e.g. functionality of different systems, machines, etc.	MS02 – Composite manufacturing MS03 – Produce number of wings MS04 – Wingspan MS05 –TOGW MS07 – Minimize defects MS08 – Minimize late parts MS09 – Limit WIP inventory MS10 – Reduce traveled work MS14 – Tooling Cost MS15 – Rework MS16 – Machine capability
Perform Metal Manufacturing	The creation of metal structures by cutting, bending, and assembling. It consists of two possible facilities and tools- sheet metal or CNC machining	The process of metal manufacturing can be documented to understand the flow of activities needed to produce a product.	MS01 – Metal manufacturing MS11 – Milling machine MS12 – Heat treatment MS13 – Riveting MS03-MS10, M14-M16
Perform Same Manufacturing Process, with Different Materials	This use case looks at the differences in process flow between different materials, while using the same manufacturing process (composite or metal)	Determines how the process flow of activities will change if the same process is used (e.g. composite) but different materials are used to create the final product.	M03-M6

Table 9. Production Use Cases

Use Case	Description	Purpose	Satisfied Requirements
Change Machine Capacity	A short-term study in which a fixed sample of workpiece is produced under constant conditions.	Machines have varying capability levels that can change the overall quality of a production process. Influences by different batches of material, operator, or surrounding conditions can be considered.	PS03 – Planning, scheduling, capacity PS06 – Produce 10 wings per month
Vary Takt Time and Flow Rate	At any point of production, the number of wings produced per month may ramp up or ramp down.	Determining how the production process changes as takt time and flow rate increase or decrease is important to analyze to see what needs to be changed, in order to meet demand.	PS01 – Takt time PS06 – produce 10 wings per month
Change Working Capability	Workers may be shifted around to different lines, there may be new employees, employees leaving, etc.	The working capability may change as new factors are introduced to the system.	PS04 – Transportation & Logistics PS05 – On time Delivery
Vary Part flow	The process of part flow from outside or inside suppliers to the line. This also includes keeping track and forecasting inventory	In order for parts to show up to the line on time, the production system needs to be able to plan and schedule the capacity for each shift.	PS02 – Square footage PS03 – Planning, scheduling, capacity PS04 – Transportation & Logistics

Activity Diagrams

The aforementioned use cases can be decomposed in MagicDraw using an activity diagram. The purpose of these views is to provide a high-level understanding of different wing manufacturing

processes and integrate the information into other tools. To properly capture process tasks, geometric detail and material selection is needed and is generally provided by structural design and sizing. The composite manufacturing process with VARTM is shown in Figure 21, the composite manufacturing process with Autoclave is shown in Figure 22, the sheet metal manufacturing process is shown in Figure 23, and the aluminum process is shown in Figure 24.

In general, sizing for composite materials requires more detailed analysis than for metallic ones. The composite processes using VaRTM cure and autoclave, as well metallic processes using aluminum and sheet metal were modeled and compared for the proof of concept.

Figure 21. Composite Manufacturing Process with VaRTM Cure

Figure 22. Composite Manufacturing Process with Autoclave

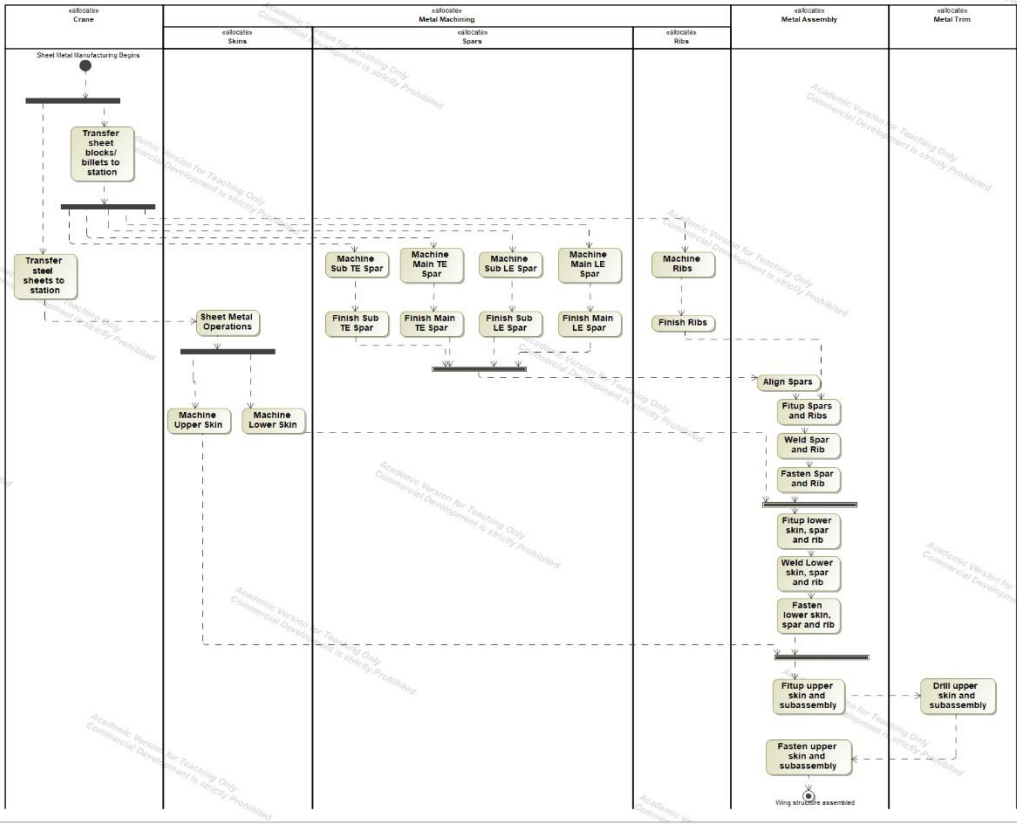


Figure 23. Sheet Metal Manufacturing Process

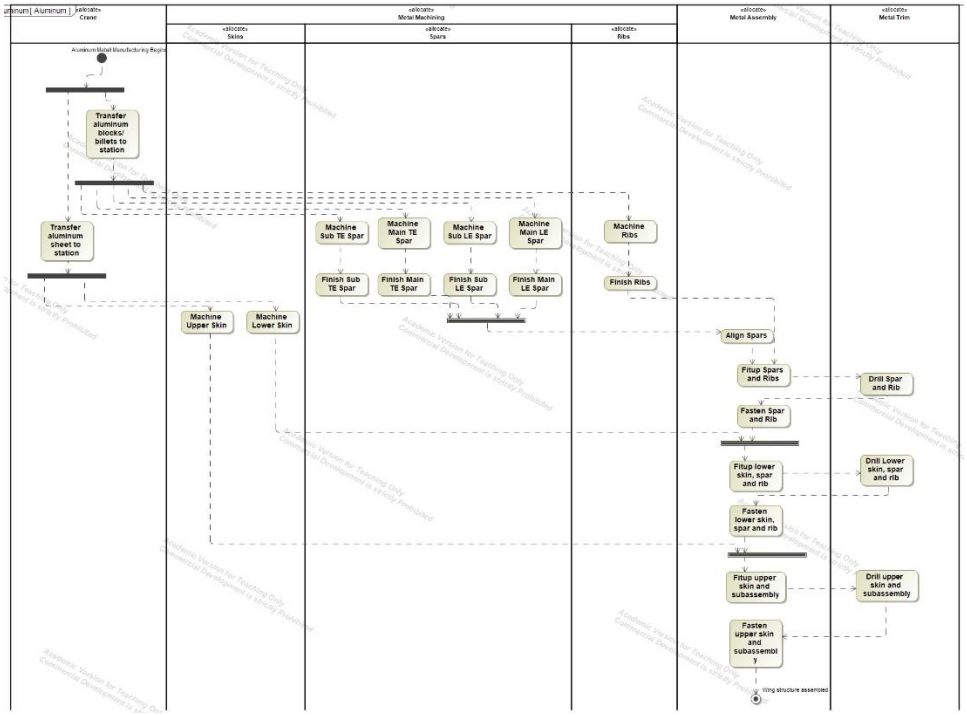


Figure 24. Aluminum Manufacturing Process

Each “swimlane” in the activity diagram is identified as a function, such as metal machining and metal assembly. The activities under each swimlane dictate the process in which the aircraft components are created and come together to manufacture a wing.

Block Definition Diagram

A Block Definition Diagram (BDD) is a static structural diagram that shows system components, their contents (Properties, Behaviors, Constraints), Interfaces, and relationships. Several BDDs were created with regards to the manufacturing structure to apply value and part properties to all of the different processes (Figure 25 and Figure 26).

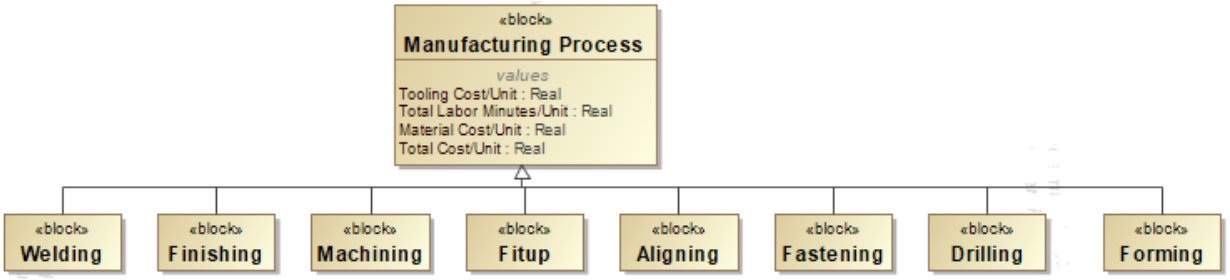


Figure 25.Manufacturing Specialization

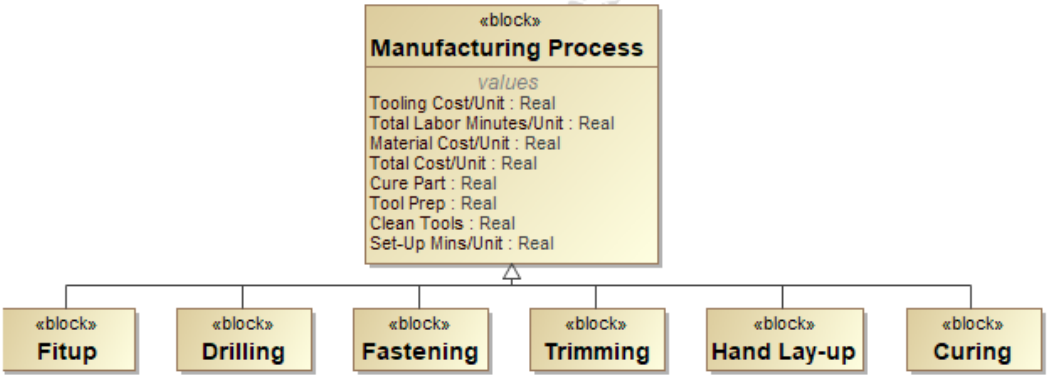


Figure 26. Composite Manufacturing Specialization

As shown in the figures above, the manufacturing process block has value properties of tooling cost, total labor minutes, material cost, and total cost, as well as cure part, tool prep, clean tools, and set-up minutes for composites. It is also “generalized” by the different manufacturing processes that are used to build an aircraft wing, such as finishing, machining, drilling, forming, etc. This allows all of the value properties to be passed along to the different manufacturing processes and provide traceability.

Furthermore, each part that is manufactured (e.g. ribs, sub-main spar, main front spar, upper skin, etc.) has a block diagram associated with it as well. An example of the rib/spar subassembly breakdown structure is shown in Figure 27. This allows for development of parametric diagrams.

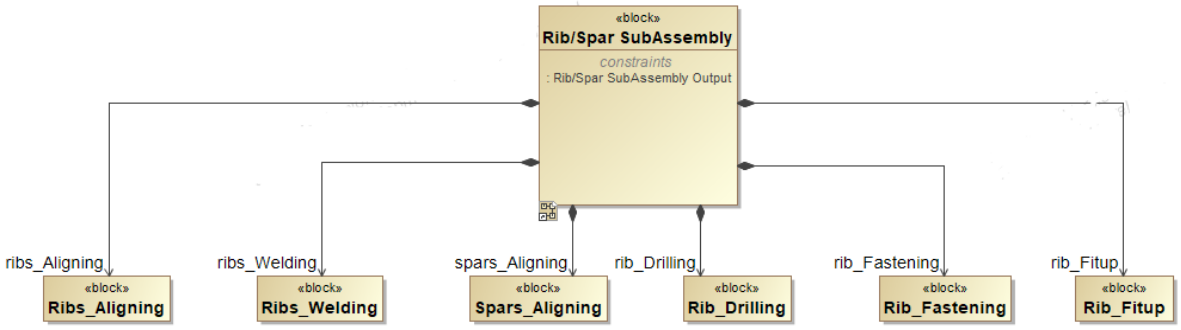


Figure 27. Rib/Spar Subassembly

Requirements Verification and Validation (V&V)

Requirement verification is a quality check of the analyzed requirements. This task involves ensuring that each requirement is correct, complete, and meet the quality standards defined for them. Requirement validation is the actual testing techniques to ensure that the requirements are correctly implemented.

The four fundamental methods of verification are Inspection, Demonstration, Test, and Analysis. In the context of this research, the primary method of verification is through analysis. Analysis is the verification of a system using models, calculations and testing equipment. It allows for predictive statements about the performance of the system based on a series of test results.

To ensure that V&V is traceable, requirements are refined by using constraint blocks. This enables automatic requirements verification by running a simulation and inputting relevant values such as spar thickness, takt time, etc. This determines whether or not the requirement is satisfied based on its specified conditions.

An example of a constraint diagram and its corresponding parametric diagram are shown in Figure 28 and Figure 29.

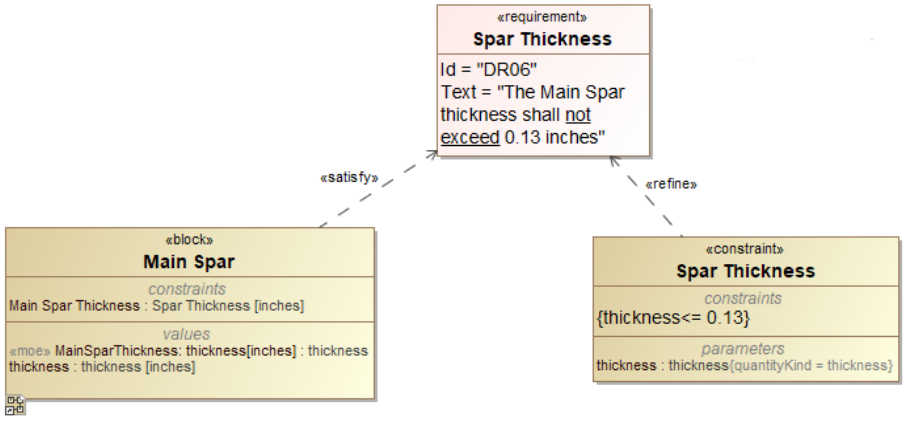


Figure 28. Requirements Verification

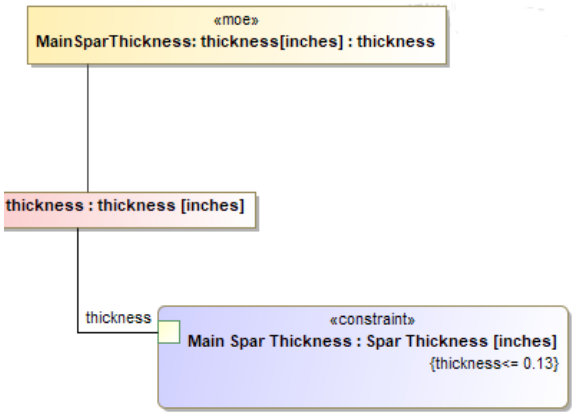


Figure 29. Requirements Parametric Diagram

After identifying the input port (green outlined square) to the constraint block, the requirement is ready for verification. To do so, the simulation button is pressed, which opens a simulation window. Then, a value can be given to the constraint property and this determines whether the requirement is satisfied (highlighted in green, as shown in Figure 30) or not satisfied (highlighted in red as shown in Figure 31). These values can be taken from the results of the structural, manufacturing, and production analyses.

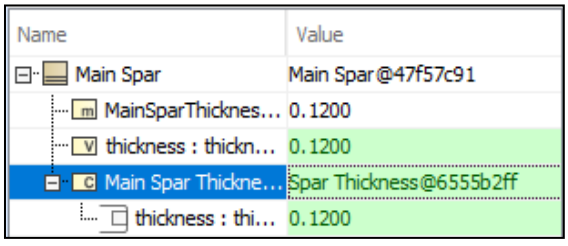


Figure 30. Requirements Simulation Window - Pass

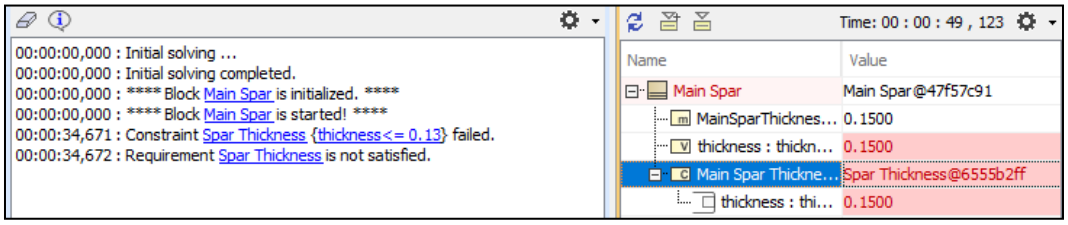


Figure 31. Requirements Simulation Window - Fail

The full results from requirements verification and validation are shown in Section VI.

Computer Aided Design of Wing

Figure 32 shows a view of the wing model as designed for metallic processes. Figure 33 shows a view of the wing model as designed for composite processes, with an aileron back spar included to more accurately capture how the physical part was constructed. These models were used for the structural analysis that ensures meeting DBF's structural requirements. From these CAD models, it is possible to extract essential information for the *As Built* or manufacturing/production phase, such as material information, number of components, and volumes of each part.

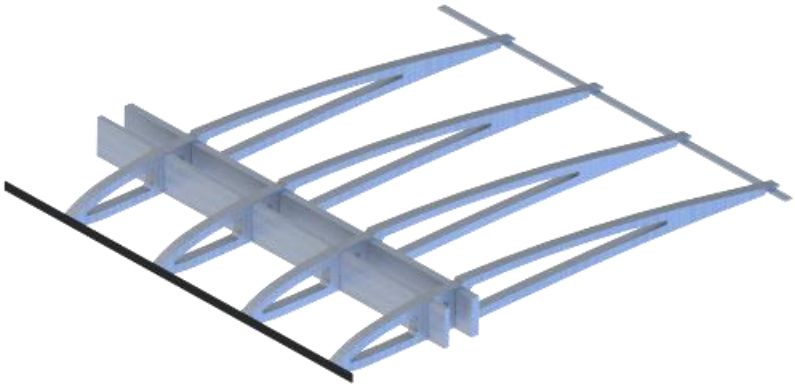


Figure 32. View of Metallic Wing

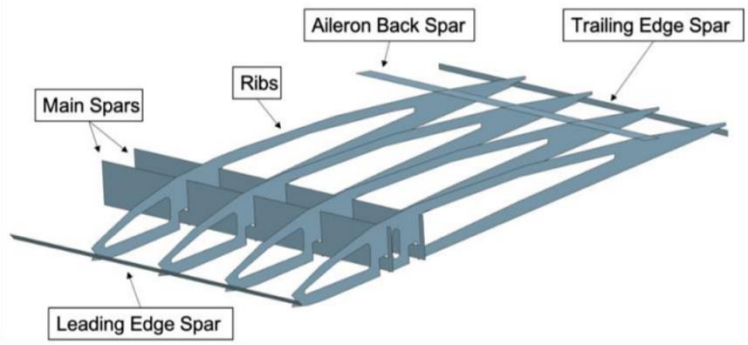


Figure 33. View of Composite Wing and Part Nomenclature

The following section discusses the results of the structural analysis. The intent is to have the geometry data from the wing CAD and from its structural analysis be shared and exchanged with the manufacturing model and with the MBSE environment.

Structural Model

The structural analysis allows the engineer to know what loads and boundary conditions can be applied to the actual part for a given margin of safety before structural failure onsets. Structural failure for metallic materials, such as aluminum or steel, is usually said to have occurred when the material reaches its yield strength. Structural failure for composite materials, such as carbon fiber, is more complex and includes many different modes related to tension, compression, and shear interactions. However, in this initial demonstration, this is simplified so that failure is said to have occurred when the material reaches its yield strength. This is done to keep the methodology

as simple as possible in its initial iteration to showcase it is able to make material tradeoffs and not become needlessly convoluted with additional failure modes.

For the structural analysis setup for the composite wing, the material used is carbon fiber/epoxy material. The wing has a 10 lbs. load distributed over the ribs to ensure it is able to at least support its own weight. The boundary condition degrees of freedom (DOF) are all fixed. The same boundary conditions are used for the metallic wing. 2D CQUAD4 shell elements are used for all the parts in the composite model. For the metallic wing, the material used is aluminum 6061 for the aluminum wing, with the ribs being made of steel instead of aluminum for the aluminum/steel hybrid wing. The loads and their distributions are the same as with the composite wing. Furthermore, the 3D mesh size is 0.33 inches and there are 1D elements between flaps and main wing. Figure 34 shows views of the loads and boundary conditions (right) as well as the Von-Misses stress values (left) from the preliminary FEM of the composite wing, with the setup described as in the previous paragraph. The loads, boundary conditions, and stress values for the metallic wing are not shown due to being qualitatively similar.

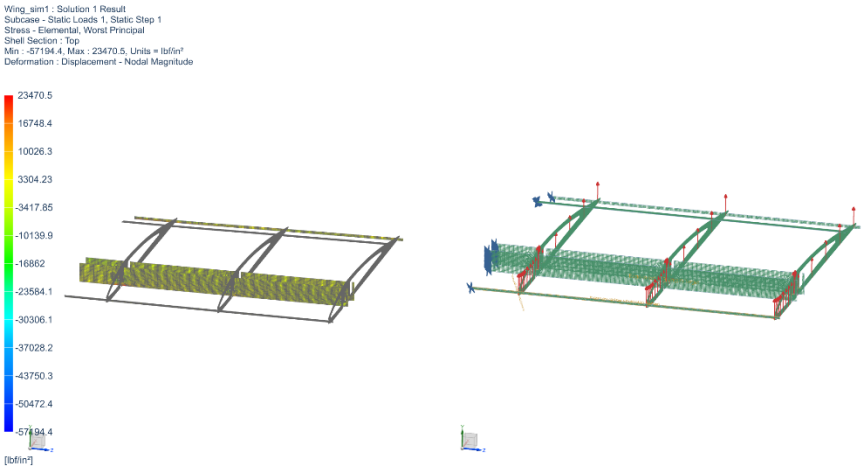


Figure 34. Von-Misses Stress Distribution for the Composite Wing

This analysis produces a .dat file, which defines the mesh, loads, and boundary conditions. To create a parametric structural model, the team went beyond making this file parametric. Modifying the .dat is not a trivial task given that it has approximately two million lines. An automated and systematic process that updates the geometry and runs the FE analysis is therefore required. The chosen design variables of interest for the wings were wingspan, rib thickness and spacing, as well as quantity of ribs.

In this process, the design variables can be manipulated to provide information on various configurations of a system. The script that is used for running the structural analysis is integrated into MagicDraw through the MATLAB language to provide a way to start and complete structural analyses without opening an external tool. With the completion of this task, a way of parametrically updating the finite element model was developed. Figure 35 illustrates how this process occurred, which is the same for both the metallic and composite wings.

MATLAB is used to loop through the Design of Experiments (DoE) created in an Excel file. The features and their ranges are the wingspan (affects all disciplines): 26 – 36’’, rib spacing (only affects structural response): 1.6 – 3.2”, rib thickness (affects all disciplines): 0.16 – 0.3” and

quantity of ribs (affects all disciplines): 6 – 8. General dimensions and 98th percentile stress are extracted during the process to satisfy the model requirements. The reason for using the 98th percentile stress instead of the 100th percentile stress is explained further in the results section but, put simply, pertains to stress risers and non-physical values. The stress data produced by running the DoE is then matched with their respective inputs and both these sets of data are used to create surrogate models. These surrogate models allow for easier integration with MagicDraw and also allow future prediction of stresses to be quicker as well, assuming the inputs are within the ranges used for the DoE.

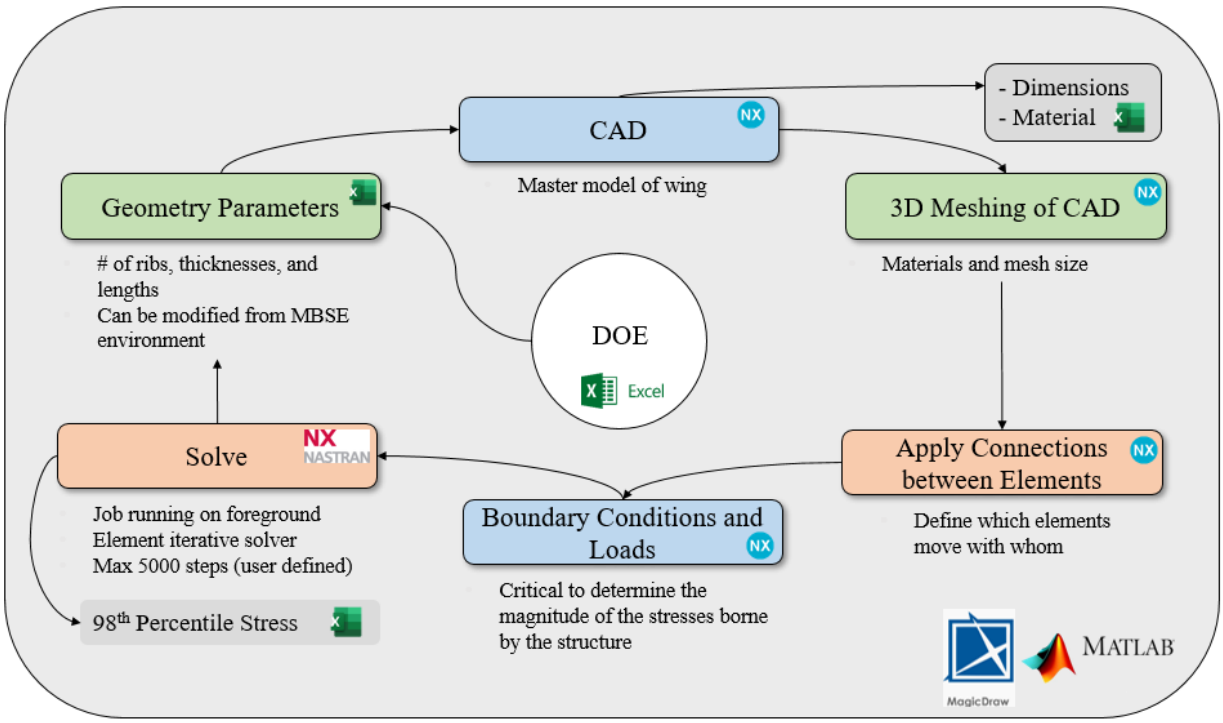


Figure 35. Overview of Automated Capability of Parametric Structural Model

As Built

The *As Built* phase focuses on the connections between the system model with the manufacturing, production, and structural models. These enable the sharing of models, data and analysis to ultimately facilitate trade studies. This section details how each model is created in its specified tool/platform and connected to the *As Designed* phase.

Manufacturing Model

There are three manufacturing models of interest that can be used to manufacture a wing. This includes composites, metals, and additive manufacturing. Table 10 through Table 12 provide the matrix of alternatives for composites, metallics and additive manufacturing, respectively.

Table 10. Composite Manufacturing Options

Composite Layup	Composite Cure	Trim	Assembly	Nondestructive Inspection
3D Weaving	Autoclave	5- Axis CNC	Co-Curing	Pulse Echo
Braiding	Electron Beam	Hand Router	Co-Bonding	Hyscan

Filament Winding	Resin Transfer Molding	Water-jet	Paste Bonding	Theodolite
Automated Fiber Placement	Vacuum Assisted Resin Transfer Molding (VaRTM)	Ultrasonic	Electron Beam	Coordinate Measuring Machine
Hand Layup			Stitching	X-Ray
Tow Placement			Z-pins	
Pultrusion			Drill and Fasten	

Table 11. Metal Manufacturing Options

Trim	Assembly	Machining
Turning	Folding	Lathe
Lasers	Welding	CNC Machining
Plasma torches		
Water jet		
Drilling		
Grinding		
Shearing		
Punching		

Table 12. Additive Manufacturing Options (Ben Redwood)

Powder Bed Fusion	Vat Polymerization	Sheet Lamination	Binding	Material Extrusion	Jetting	Direct Energy Deposition
Multi Jet Fusion	Stereolithography (SLA)	Laminated Object Manufacturing	3D printing	Fused Deposition Modeling	Material Jetting	Laser Engineered Net Shape (LENS)
Selective Laser Sintering	Direct Light Processing (DLP)	Sheet Lamination	Binder jetting		Nano Particle Jetting	Electron Beam Additive Manufacture (EBAM)
Electron Beam Melting	Continuous DLP				Drop-on-Demand	
Direct Metal Laser Sintering						
Selective Laser Melting						

Furthermore, design and manufacturing scenarios for the skin and rib design can be defined from the different alternatives shown in Figure 36.

	Aluminum Design #1	Aluminum Design # 2	Composite Design #1	Composite Design #2:
Material	Aluminum	Aluminum/Steel (rib)	Carbon Fiber	Carbon Fiber
Fabrication Technique	CNC Machining	CNC Machining	Hand Lay-up Autoclave Cure	Hand Lay-up VaRTM Cure
Integration	Very High Integration	Very High Integration	Low Integration	High Integration
Structural Data	Industry Practices & First Order Structural Analysis	Industry Practices & First Order Structural Analysis	Industry Practices	Industry Practices
Joining Technique	Fasteners (Assembly Only)	Fasteners (Assembly Only)	Fasteners (Assembly Only)	Co-Bonding/Paste Bonding

Figure 36. Design and Manufacturing Scenarios

After determining which alternatives will be used with the manufacturing process, the corresponding activity diagram is selected from MagicDraw and translated into SEER-MFG. SEER-MFG from Galorath, Inc. is a commercially available software that leverages a cooperative effort between government and industry to reduce acquisition cost of aero-composite structure under the Composite Affordability Initiative (CAI). It enables:

- **Greater design freedom**
  - There is a wide breadth and depth of manufacturing data
  - It is empirically based aerospace libraries
  - The tool is non-proprietary and commercially available
- **Production planning trades**
  - Allows for process-based cost modeling
  - Configurable at the activity level
- **Preliminary level structures, manufacturing trades, and beyond**
  - Non-weight based
  - Preliminary level fidelity (e.g. Number of parts, thickness, number of fasteners)
- **Parametric capability**

SEER-MFG utilizes a bottom-up approach, which allows for granularity in modeling individual tasks. Additionally, the estimates provided are reasonably accurate since it is backed with a plethora of validated data. The required level of detail is not also too high, making this tool amenable to be used in the early design phase. A screenshot of the aluminum manufacturing process model is shown in Figure 37.

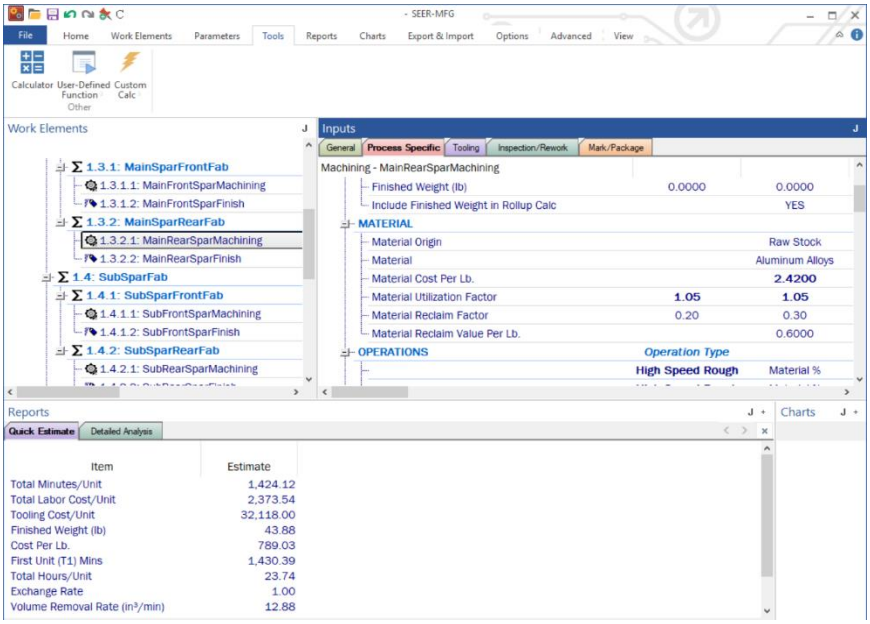


Figure 37. Screenshot of SEER-MFG model for aluminum manufacturing process

Some major assumptions were made when developing the metallic manufacturing models. These include:

- Blind rivets are used for fastening due to possible accessibility issues and small size of aircraft not requiring fasteners to be extremely robust
- Parts are generally of adequate thickness for riveting; parts that are too thin are attached via riveting to tabs that add insignificant cost and time
- Steel rib fabrication tasks are based off of aluminum ones except that machining processes occur more slowly
- Due to the small size of parts, the same set of tooling was assumed adequate for the entire production run of 1000 wings
- The tooling cost was amortized over 1000 wings
- Due to the small size of the parts, a single setup can be used to produce ten parts

A separate set of assumptions were used when developing the composite manufacturing models. These include:

- Assembly steps for Hand Layup/Autoclave process are the same as metallic ones except that different materials are being used
- Ribs have a 1'' wide flange around their perimeter to aid in their assembly to the spars and skins
- Hand layup steps are the same between the Autoclave and VARTM processes
- The composite layups are quasi-isotropic, meaning each part consists of 25% 0° plies, 25% 90° plies, 25% 45° plies, and 25% -45° plies
- Each part type is cured together in the same batch, i.e., all types of spars are cured together, upper and lower skins are cured together, more than one rib is cured at a time (when applicable)
- Due to the small size of parts, the same set of tooling assumed adequate for the entire production run of 1000 wings
- The tooling cost was amortized over 1000 wings

- Parts in the Autoclave process spend at least 6 hours curing, while parts in the VARTM process spend at least 90 minutes curing
- Due to the small size of the parts, a single setup can be used to produce ten parts

SEER-MFG receives inputs from the structural model after its structural analysis is completed. These inputs include part geometry and materials used. Additionally, it takes as user inputs information about the process type, part size, fabrication technique, technique details, etc., to estimate the total time needed. It also supports functionality for estimating tooling cost, material cost, and finished weight. The outputs from SEER, which correspond to a specific manufacturing process, are then fed into the production model in SimPy (for metallics) and in Simio (for metallics and composites). For example, in forming the upper skin, there is a total cost/unit, material cost/unit, tooling cost/unit, and total labor minutes/unit value associated with it. The complete list of the inputs, manufacturing activities and outputs are shown in Table 13 for the metallic wing and Table 14 for the composite one. The additional outputs for the composite wing are used to better separate workstations and workstation times in the production modeling due to intricacies only applicable for composite processes, such as curing.

Table 13. Metallic Wing Input and Output Information

Inputs	Manufacturing Activities	Outputs
BMainSpar_M_Rlength	UpperSkinForm	Material Cost/Unit
BMainSpar_M_Rwidth	UpperSkinMachining	Tooling Cost/Unit
BMainSpar_M_Rthickness	LowerSkinForm	Total Cost/Unit
FMainSpar_M_Rlength	LowerSkinMachining	Total Labor Minutes/Unit
FMainSpar_M_Rwidth	RibMachining	
FMainSpar_M_Rthickness	RibFinishing	
LE_Spar_M_Rlength	MainFrontSparMachining	
LE_Spar_M_Rwidth	MainFrontSparFinish	
LE_Spar_M_Rthickness	MainRearSparMachining	
LSkin_SF_Length	MainRearSparFinish	
LSkin_SF_Width	SubFrontSparMachining	
LSkin_SF_Thickness	SubFrontSparFinish	
Rib_M_Qty	SubRearSparMachining	
Rib_M_Rlength	SubRearSparFinish	
Rib_M_Rwidth	SparAlignment	
Rib_M_Rthickness	FitupRibs	
TE_Spar_M_Rlength	DrillRibstoSpars	
TE_Spar_M_Rwidth	FastenRibstoSpars	
TE_Spar_M_Rthickness	FitupLowerSkin	
USkin_SF_Length	DrillLowerSkin	
USkin_SF_Width	FastenLowerSkin	
USkin_SF_Thickness	FitupUpperSkin	
	DrillUpperSkin	

Table 14: Composite Wing Input and Output Information

Inputs	Manufacturing Activities	Outputs
BMainSpar_M_Rlength	Upper/LowerSkinHLU	Material Cost/Unit
BMainSpar_M_Rwidth	Upper/LowerSkinCure	Tooling Cost/Unit
BMainSpar_M_Rthickness	Upper/LowerSkinTrim	Total Cost/Unit
FMainSpar_M_Rlength	RibHLU	Total Labor Minutes/Unit
FMainSpar_M_Rwidth	RibCure	Part Cure Minutes/Unit (Autoclave cure only)
FMainSpar_M_Rthickness	RibTrim	Tool Prep Minutes/Unit
LE_Spar_M_Rlength	MainSparForward/RearHLU	Clean Tools Minutes/Unit
LE_Spar_M_Rwidth	MainSparForward/RearCure	Setup Minutes/Unit
LE_Spar_M_Rthickness	MainSparForward/RearTrim	
LSkin_SF_Length	Leading/TrailingSparHLU	
LSkin_SF_Width	Leading/TrailingSparCure	
LSkin_SF_Thickness	Leading/TrailingSparTrim	
Rib_M_Qty	AileronSparHLU	
Rib_M_Rlength	AileronSparCure	
Rib_M_Rwidth	AileronSparTrim	
Rib_M_Rthickness	FitupSparRibs	
TE_Spar_M_Rlength	DrillSparRibs (Autoclave only)	
TE_Spar_M_Rwidth	FastenSparRibs (Autoclave only)	
TE_Spar_M_Rthickness	PasteBondSparRibs (VARTM only)	
USkin_SF_Length	FitupLowerSkin	
USkin_SF_Width	DrillLowerSkin (Autoclave only)	
USkin_SF_Thickness	FastenLowerSkin (Autoclave only)	
Aileron_Spar_length	PasteBondLowerSkin (VARTM only)	
Aileron_Spar_width	FitupUpperSkin	
Aileron_Spar_thickness	DrillUpperSkin (Autoclave only)	
percent_zero	FastenUpperSkin (Autoclave only)	
percent_ninety	PasteBondUpperSkin (VARTM only)	
percent_forty_five		
Aileron_Spar_length		

Production Model

The production model simulates the sequence of the manufacturing steps, queuing, storage, supply, and assemblies. For this reason, the production model depends on the selected manufacturing model and processes. The production model builds on a number of data sources: the CAD Model, the information from SEER-MFG, structural analysis, and facility characteristics. Some of the information that is required from each data source is shown in Figure 38.

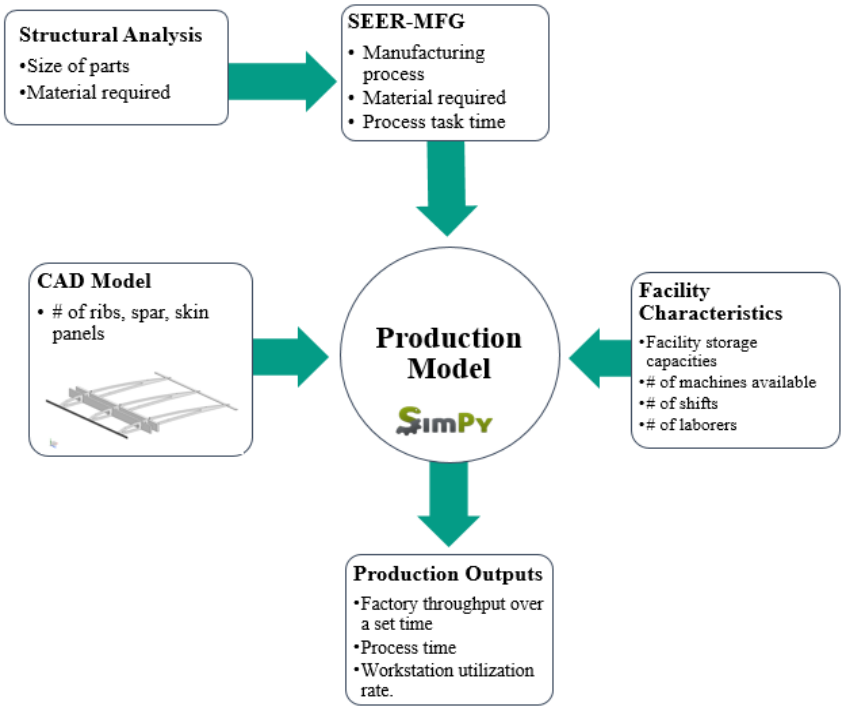


Figure 38. Production Model Data Sources

An example of the production model for the metallic wings is shown in Figure 39, which is a more comprehensive model of the manufacturing process. It starts with materials being supplied by a supplier and the process of creating rib, skin panels, and spars for a wing. After the wing is assembled, the entire structure is sent to the aircraft assembly location in the factory. Figure 40 and Figure 41 show the production/manufacturing steps for the two types of composite wings.

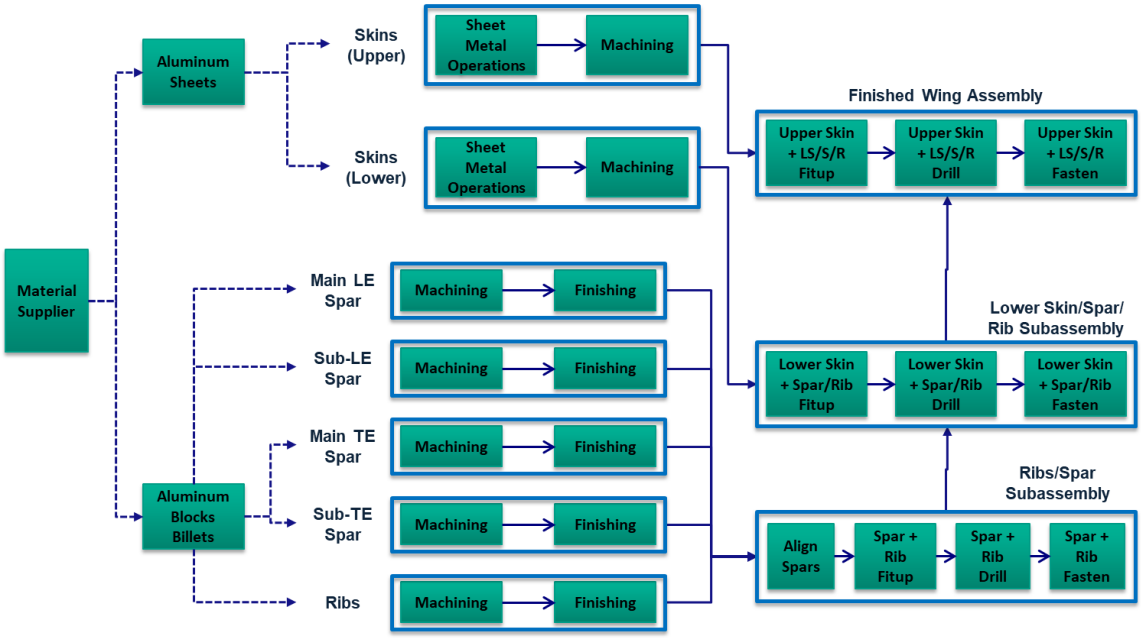


Figure 39. Production Model for the Manufacturing of an Aluminum Wing (Siedlak et al., 2018)

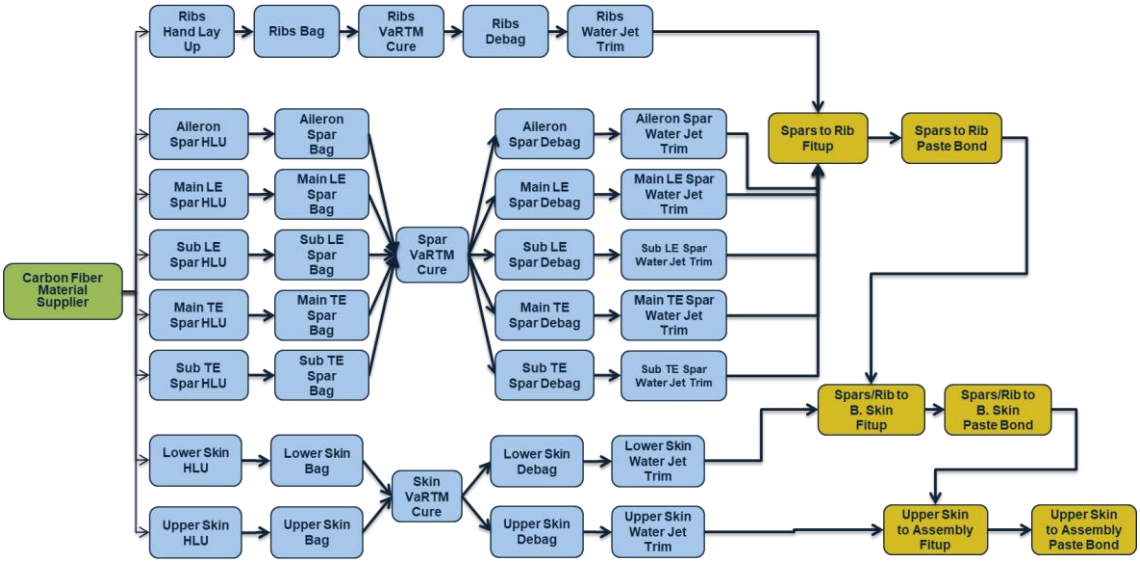


Figure 40. Production Model for the Manufacturing of a VARTM Composite Wing

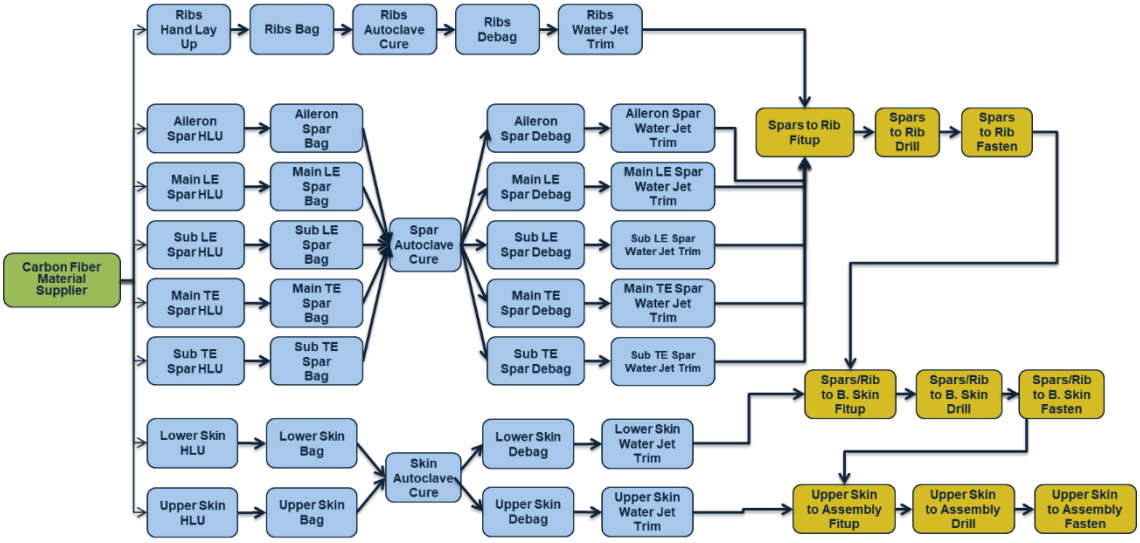


Figure 41. Production Model for the Manufacturing of an Autoclave Composite Wing

The production model outputs the factory throughput over a set time, process time, and workstation utilization rate. How this occurs and the calculations involved, however, is slightly different depending on whether SimPy or Simio is used to model the production of the wing.

The SimPy model is the more simplistic of the two. In SimPy, each of the individual activities inside the blue-green boxes in Figure 39 is represented by a task. The amount of time it takes to complete a task is determined by the respective outputs obtained from SEER-MFG. Once a task is finished, the part involved in a task moves on to the next one until all the parts from all tasks

are assembled together through the assembly tasks. A complete wing is produced when the last assembly task is finished. All of the tasks are constantly being carried out as in a production line and thus all are essentially being done in parallel; the fact that the final assembly task for, say, the 37th wing is being carried out does not prevent the ribs for, say, the 44th wing from being machined out and the spars for the 40th wing from being aligned. The primary limits on production rate are thus the length of the individual task times as well as the fact that each task has to wait until the tasks immediately preceding it are finished. This is to say that the spar and rib fitup task for, for example, the 37th wing cannot start until the align spars task for the 37th wing have been finished. Each task is associated with a workstation in SimPy and the amount of time a workstation spends working on a task vs. being idle and waiting for the preceding tasks is what determines the utilization for that station. To help simplify the SimPy model, only four stations are used: a station for all the spar fabrication tasks, one for all the rib fabrication tasks, one for all the skin fabrication tasks, and one for all the assembly tasks. Changing the number of these stations can reduce or increase the amount of time the proceeding station spends waiting for a part to be finished.

The Simio model is significantly more complex than the SimPy model in that it tracks far more resources needed to complete each task. All aspects of the SimPy model also hold true for the Simio model except that Simio also tracks the number of workers and tools in circulation. Unlike SimPy, where a task only needs to wait for the immediately preceding tasks to finish before it is able to start, in Simio a task also needs to wait until there is an available worker to carry out the task as well as an available tool to transport the part from the preceding task. It is thus significantly easier for a workstation to be idle in Simio due to waiting for available workers or tools and therefore create a bottleneck that limits throughput. Additionally, Simio is more flexible and optimized than SimPy, allowing for the workstations to be more discrete and easily changed. In Simio, a workstation can consist of only a single task as opposed to Simpy where a workstation is a collection of tasks. This is due to Simio's more streamlined calculation capabilities. Overall, without optimal settings, the throughput outputs from Simio tend to be lower than the SimPy ones because Simio keeps track of more throughput-limiting factors than SimPy. An example of a small portion of the Simio interface depicting the implementation of the fabrication tasks is shown in Figure 42. More detailed settings can be found in Appendix.

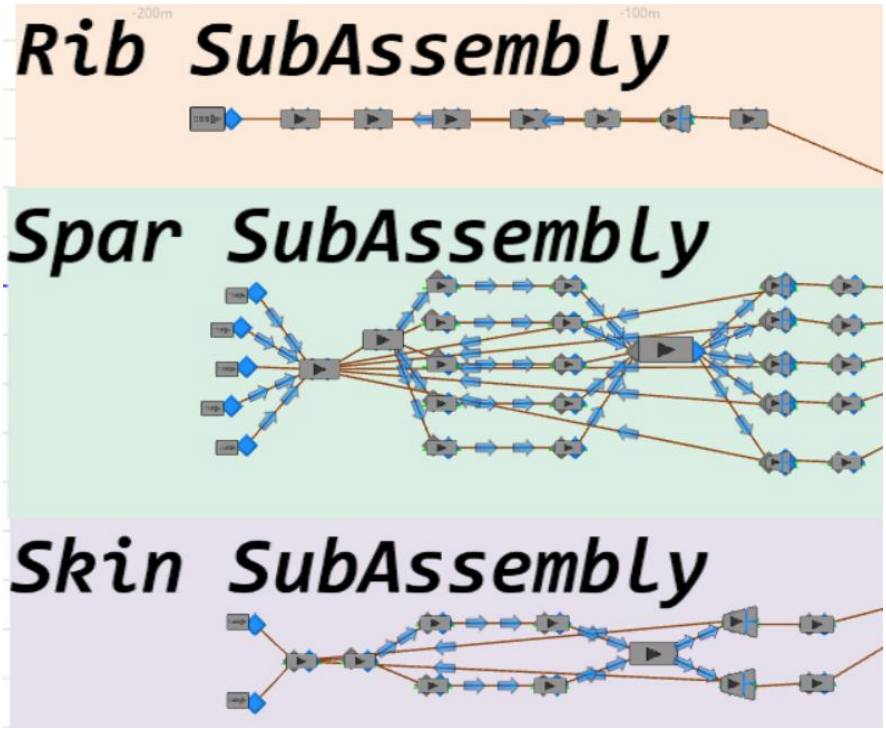


Figure 42. Simio Implementation and Visualization of Fabrication Tasks

To facilitate a one-to-one comparison with the composite wings, a separate Simio model was also constructed for the metallic wings. The primary difference between the metallic wing model in SimPy and the one developed in Simio, besides the differences endemic to the two being different programs, is that the tasks that were consolidated into one station in the SimPy model are split up into their own stations in the Simio model. As an example, instead of all the spar tasks for all four type of metallic spars being condensed into one station, the spar machining and finishing tasks for all four spar types now each belong to their own stations. This metallic Simio model will be used as the main point of comparison with the composites model.

External Tool Integration with MagicDraw

With the completion of the structural, manufacturing, and production models, a way to integrate the external tools (NX, SEER-MFG, SimPy and Simio) into the MBSE environment had to be architected. The goal was to create and demonstrate the capability of a consolidated infrastructure that could run analysis easily.

Parametric Diagram

A parametric diagram is used to constrain the properties of a block as inputs are passed along from analysis. As shown in Figure 43, a parametric diagram is used to link the structural model (in yellow), to the manufacturing model (in blue) and production model (in orange).

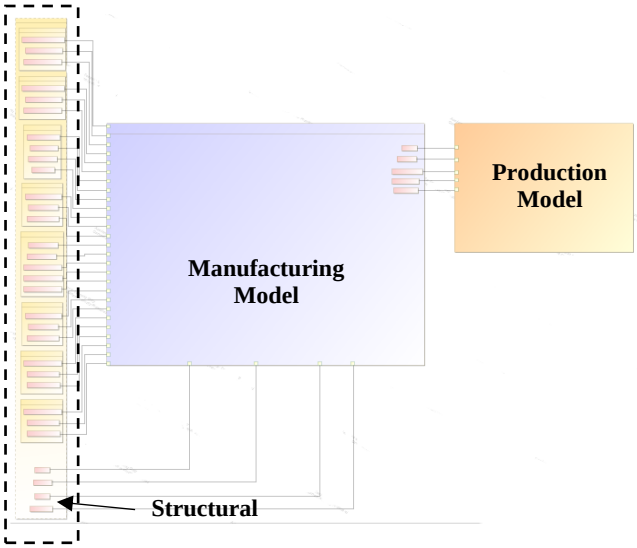


Figure 43. Parametric Diagram Connecting Structural, Manufacturing, and Production

The value properties in the structural model are directly related to the outputs that are generated from the structural analysis in NX (See Table 13). These properties are tied to its corresponding instance and then fed into the manufacturing model. The manufacturing model takes in the inputs and conducts analysis to generate properties that are fed into the production model. Finally, the production model will run its analysis and output rates such as flow time, process time, worker utilization, factory throughput, and workstation utilization.

Note: The purpose of this parametric diagram is to provide a visualization of how the inputs and outputs feed into each model. To conduct the analysis for the structural, manufacturing, and production models, please refer to the sections below for more details.

NX and MATLAB

Output files from the structural analysis are integrated into the MBSE environment via parametric diagrams as shown in Figure 44. Here, surrogate model equations which were created through structural analysis are implemented as a script in a constraint block to automatically calculate what the 98th percentile stress would be based on different processes. There are also several input variables that are used to calculate this stress, and can be changed to show how different configurations of the aircraft affect the stress. These inputs are the thickness of the ribs, the rib span spacing, amount of ribs, the half wingspan, main spar thickness, and main spar front length depending on which manufacturing process is used. Ranges for the input parameters are:

- RibSpan_Spacing: 1.6 - 3.2”
- Rib_Thickness: 0.16 - 0.3”
- Amount_Ribs: 6 – 8
- MainSparFront_Length (half of the wingspan): 13 – 18”

Both the metallic and composite process constraint blocks are able to determine the rib, leading spar, main spar, and trailing spar stress, with the composite equations calculating an additional stress for the aileron back spar.

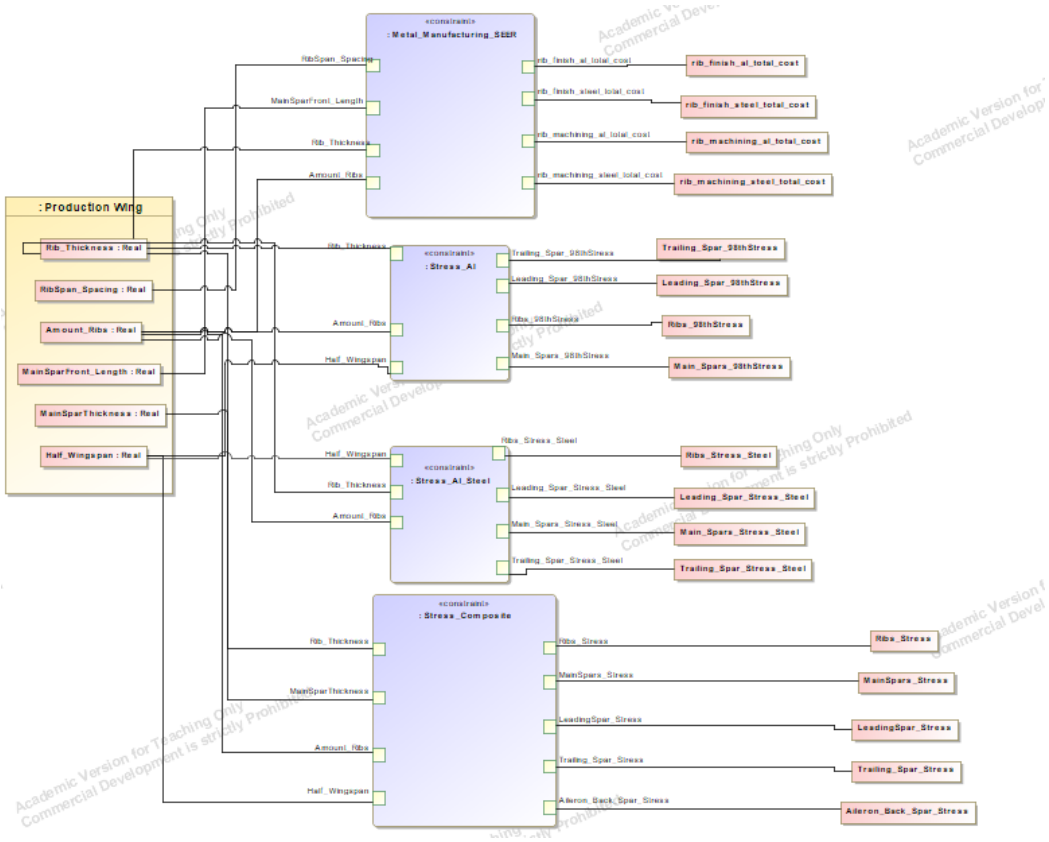


Figure 44. Structural Stress Parametric Diagram

After connecting the output files to MagicDraw, an opaque behavior (Figure 45 and Figure 46) can also be used to integrate a MATLAB script to a block. This provides a way for the structural analysis to be run completely in MagicDraw, without having to open an external tool. The script used to run the structural analysis is wrapped using a MATLAB wrapper function. This script can modify the geometry of the wing and run the simulation for several cases, from where surrogate models could then be built. The MATLAB script is defined as the “body” and MATLAB is defined as the “language” in the opaque behavior.

Opaque Behavior

Name	Run matlab script
Qualified Name	8 Structural Domain::Run Structural Analysis::Structural Analysis::Run m
Owner	Structural Analysis [8 Structural Domain::Run Structural Analysis]

```
%% dos Santos, Marcos. V 1.0. Mar 08 2021
% This script is to be used by MagicDraw to run text filse that run
NX Journal scripts for either 3
% or 4 ribs

clc
clear all
filepath= "C:\Users\msantos43\OneDrive - Georgia Institute of
Technology\My files - Fischer, Olivia J's files\RR Digital Twin Year
2\Modeling Folders\CAD and Structural Analysis\Parameters.xlsx";
r=readtable(filepath);
Amount_of_ribs=table2array(r(6,2));

if Amount_of_ribs==3
   tic
   [status,cmdout]=system('cmd < 3Ribs.txt');
   toc
else
```

Figure 45. Opaque Behavior Setup

Then, the code can be executed in its evaluation window. An example is shown in the Figure below.

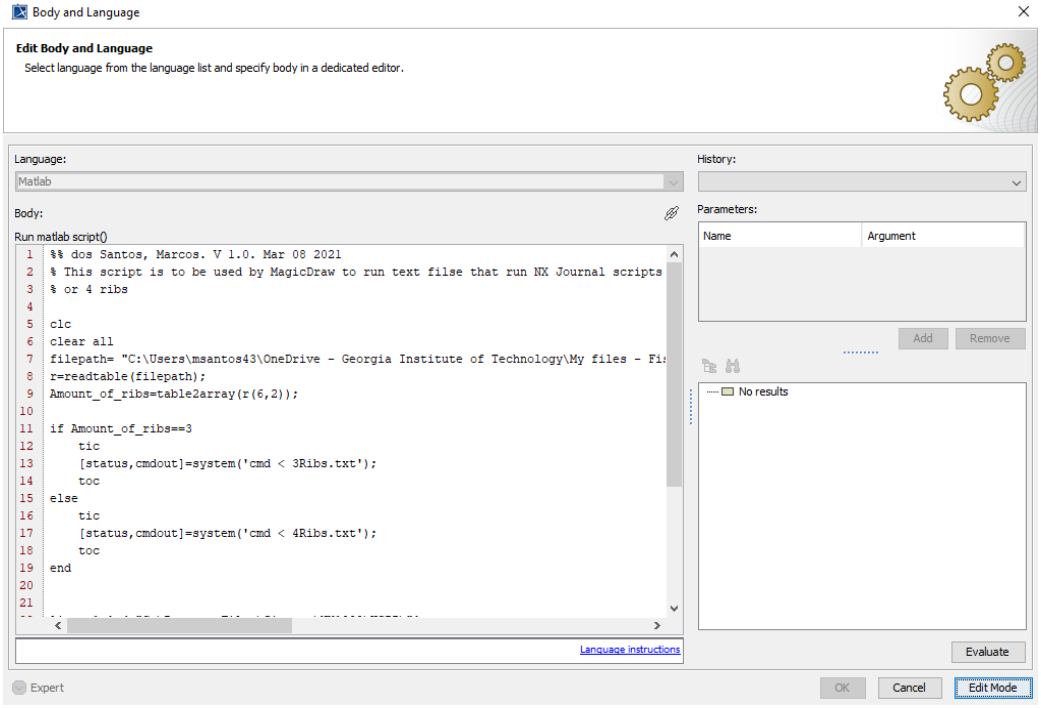


Figure 46. Body and Language Window of an Opaque Behavior

After the opaque behavior is specified, the block is ready for simulation. The simulation is run and a window (Figure 47) appears that will run the opaque behavior, and therefore the MATLAB script (and structural analysis)

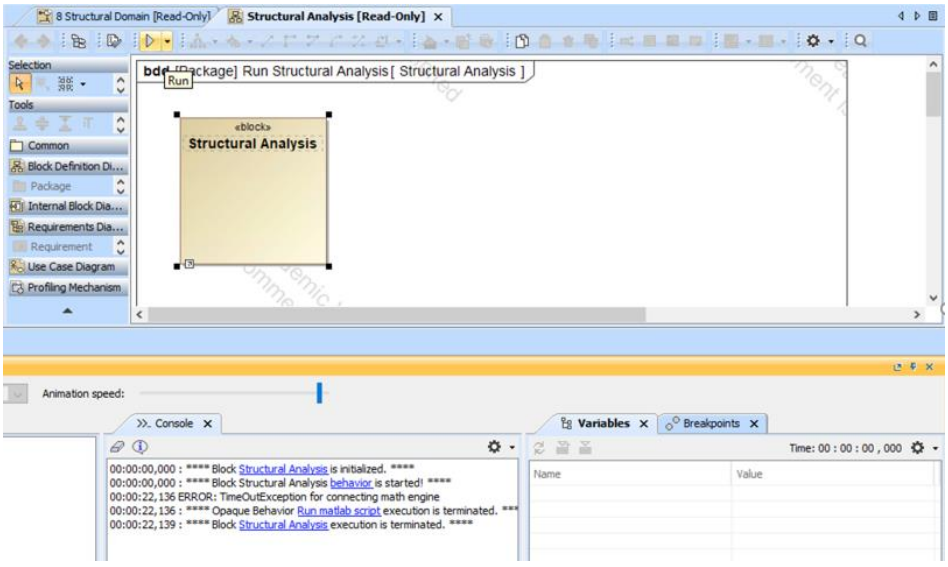


Figure 47. Structural Analysis Opaque Behavior

The analysis is complete after the opaque behavior execution is terminated. Additionally, analysis can be run once a DoE input file feeds the input geometry parameters and several cases can be run.

SEER-MFG

The manufacturing model is created in SEER-MFG. A .txt file with all the necessary inputs was created and is modified accordingly once the design variables are changed.

In the SysML model, the results from the SEER-MFG analysis are read via component specific (e.g. Main Front Spar, Sub Rear Spar, Upper Skin, Ribs, etc.) parametric diagrams, such as in Figure 48 for metallic manufacturing and Figure 49 for composite manufacturing. It also contains a decomposition of the manufacturing activities and parts that are used. The information that is read (tool cost, material cost, total cost, labor time per unit as well as set up time, curing the part, cleaning the tools for composite materials are consolidated instance table.

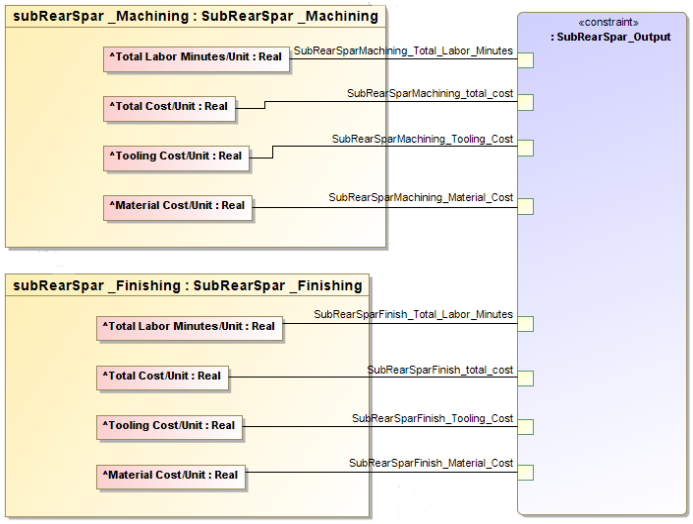


Figure 48. Sub Rear Spar Parametric Diagram

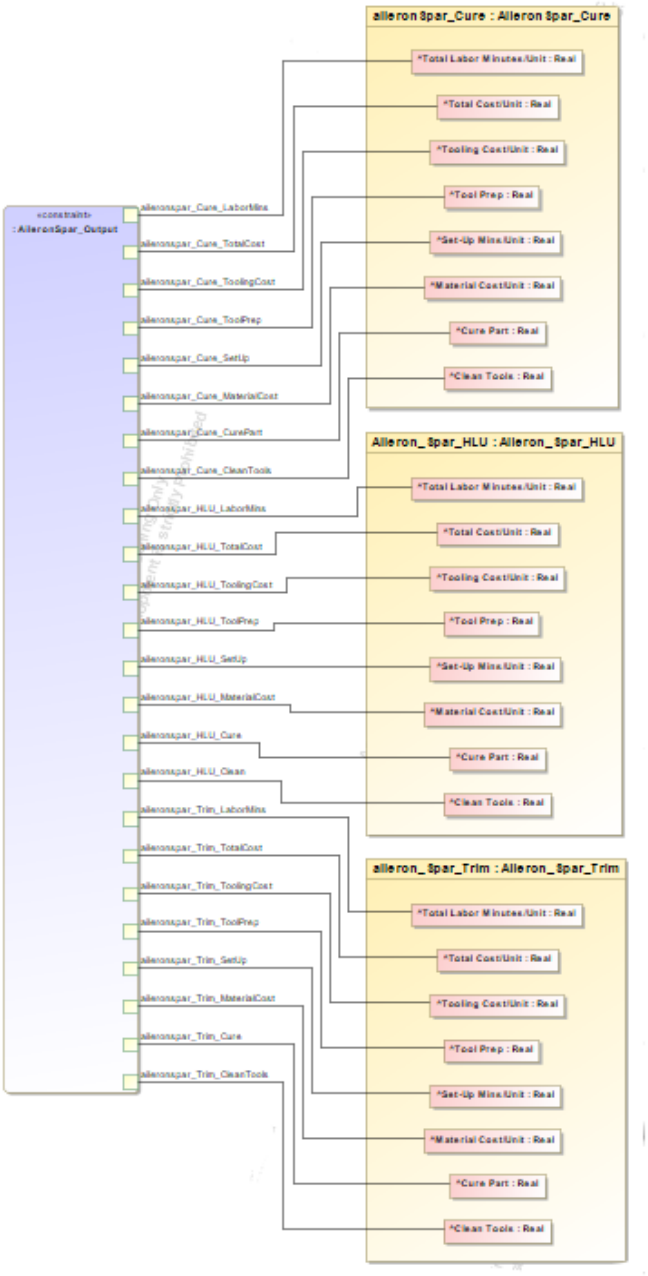


Figure 49. Aileron Spar Parametric Diagram

To read in new instances from different runs into MagicDraw, a simulation of each parametric diagram needs to be done and new instances need to be saved to the containment tree.

Python

Python is a popular, interpreted, object-oriented, high-level programming language with dynamic semantics. Coupled with SimPy, a process-based discrete-event simulation framework, Python provides teams with better modeling capabilities.

MagicDraw contains a handful of script tool APIs that allows the user to evaluate or run codes in certain programming languages. One of these supported languages is Groovy, which is a Java-syntax-compatible language. This means it has access to all of Java's classes.

Java's Runtime class allows for users to execute shell commands or run command prompts. Using Java (and ∴ Groovy) and the Runtime class, MagicDraw can execute any program from the command line, like a Python script. The outputs of the Python script can be output to a file available to MagicDraw.

These capabilities are linked to elements in the MBSE environment through the use of “opaque behaviors.” Groovy scripts can be input in the "Body and Language" section in the Specification window of the element, as shown in Figure 45. The Groovy script that is used to execute command line programs is as follows:

Runtime.getRuntime().exec "path\\to\\program.exe path\\to\\scripts.py"

After the opaque behavior is specified, the block is ready for simulation. The simulation window runs the command line program, and therefore the SimPy script.

The production analysis is complete after the opaque behavior execution is terminated. The total run time is approximately 1 second, and at this point, the output file updates and reflects the results from production analysis.

Simio

To integrate the throughput results from Simio into MagicDraw, a simple parametric diagram (Figure 50) is created to read the corresponding All-Aluminum, Aluminum-Steel, VaRTM, and Autoclave .csv files.

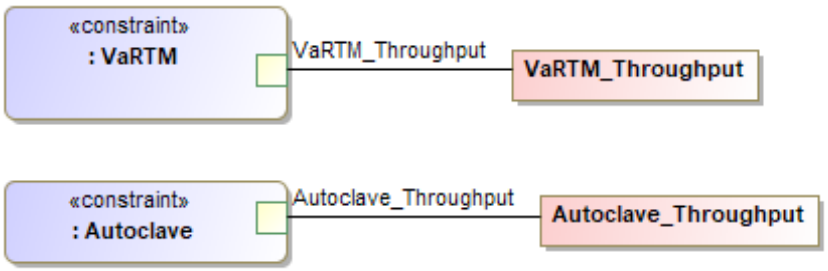


Figure 50. Throughput Parametric Diagram

Data Infrastructure

The Digital Thread brings together the *As Designed*, *As Built*, and *As Used* phases and enables all design phases to be connected seamlessly to each other. The Digital Thread is needed to enable continuous data flow, update models in real time, communicate change in requirements,

integrate and update models instantly. The Digital Thread can also store information relating to data ownership.

Previous work focused on creating a data infrastructure in Neo4j where the data is semantically linked and stored in raw format, allowing data and associated information about the data to be stored (Kargin et al., 2021). The infrastructure is shown in the figure below.

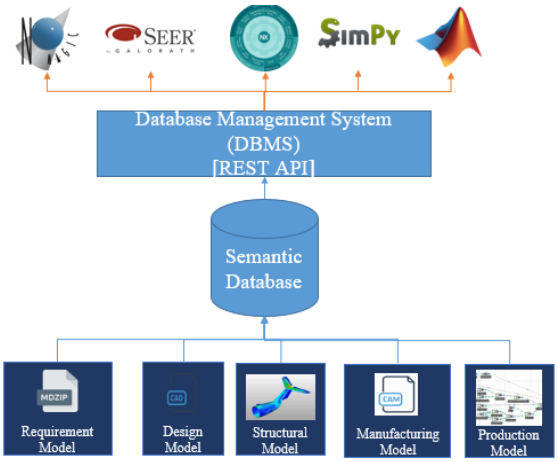


Figure 51. Data Infrastructure Schematic

The semantic database allows for the storage and version control of all data that goes through the digital thread. The Database Management System (DBMS) and REST API allow for the execution of third-party apps for continuous data and model updates.

The team leveraged the past Neo4j environment to trace data about which analyses were run and who was responsible for the creation of the models. Queries can be made whereby the desired information can be retrieved. Each model relation is traceable through the Neo4j browser.

Neo4j consists of four elements: Nodes, Labels, Relationships and Properties. Nodes are the elements that represent entities. Labels are the shapes for the domain by grouping nodes. Relationships connect two nodes and properties use Name-Value pairs to add qualities. An example infrastructure can be seen in Figure 52. Each circle is a node. Each verb such as "Manages" or "Owns" is a label. The arrows are relationships and any additional information, such as the ones in the red box, are properties.

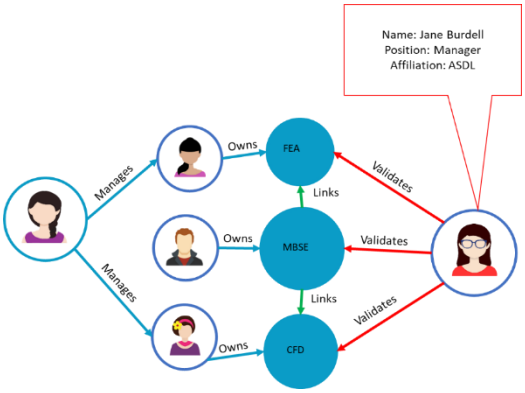


Figure 52. Example Neo4j Infrastructure

The team leveraged this data infrastructure to save the gold layers and connect each portion of the infrastructure together.

The logical links between the engineering tools are provided in a notional example in Figure 53. This illustrates the requirements and use cases being populated in MagicDraw and reflected to the Structural Analysis in NX and the Design Simulation in MATLAB. These tools direct their outputs to the manufacturing model in SEER-MFG, which in turn produces the manufacturing processes and materials required for production. Lastly, Simio hosts the Production Model, where critical parameters such as flow time, processing time and worker utilization are computed. All of these models are represented as nodes and the parameter transfer between the models are represented in relationships within the graph database, Neo4j. This allows for the querying of specific designs, such as a manufacturing model which is generated with a specific takeoff field length as its affiliated requirement input.

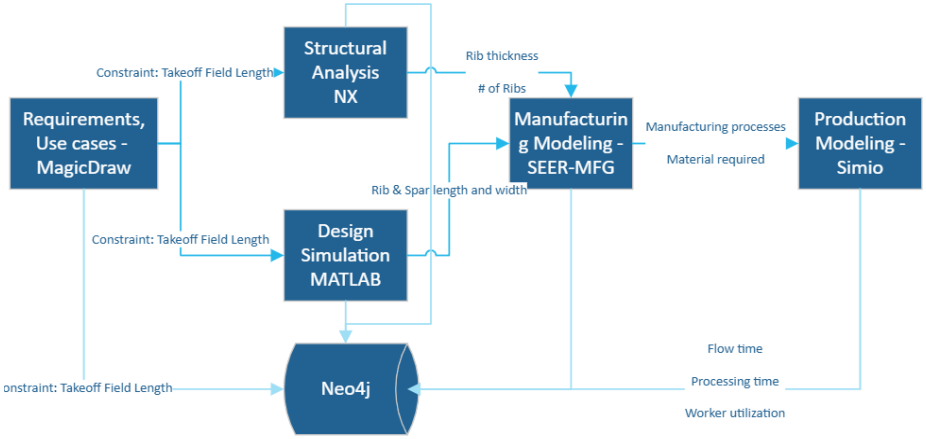


Figure 53. Parameter Flow Diagram Between Environments

Digital Enterprise – As Designed

The *As-Designed* phase consists of the design model in MBSE and the CAD model (Figure 54).

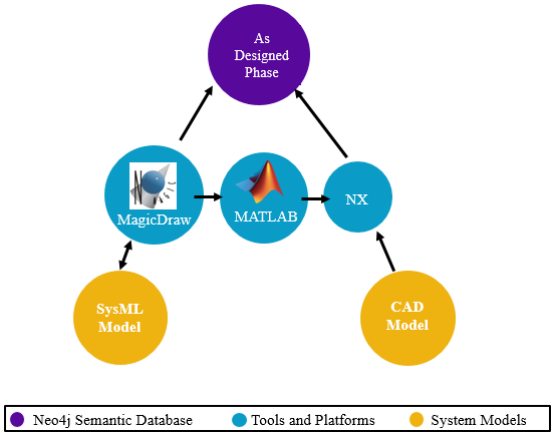


Figure 54. As Designed Data Infrastructure

Based on the figure above, the latest version of the design model is uploaded in Neo4j. The CAD model of the wing is input into the NX tool. Then, MagicDraw completes the structural analysis in NX through the use of a MATLAB wrapper function. This data and results are uploaded to Neo4j to store the latest version.

Neo4j also supports the functionality of containing information and data on different wing configurations (Figure 55 and Figure 56). To access these different configurations, similar queries are required to access the gold layer. The two different configurations are linked to the same requirements model and from there, each model relation is traceable through the Neo4j browser.

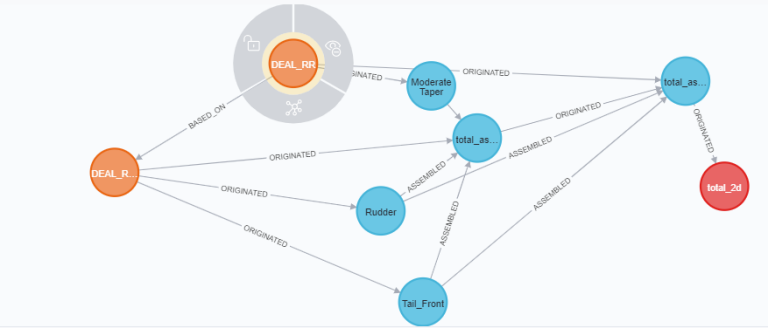


Figure 55. First Wing Configuration (Past Effort)

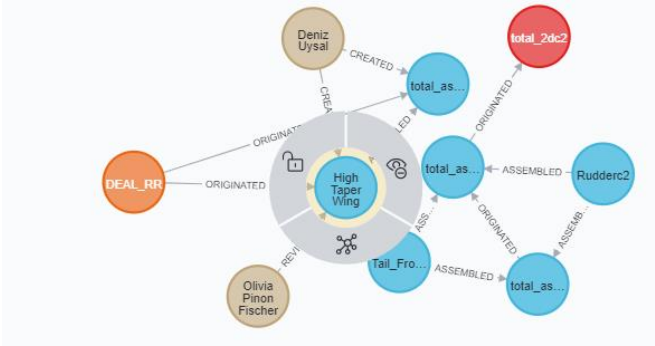


Figure 56. Second Wing Configuration

Digital Enterprise – As Built

The *As Built* phase consists of the manufacturing and production models. In this phase, the results from the structural analysis are fed into SEER-MFG to run the manufacturing analysis. Finally, the data from manufacturing is input to SimPy and Simio where data from the production analysis is generated. All the results from the analyses are then uploaded and stored in Neo4j. It is also important to note that all of these analyses can be done via the MBSE environment in MagicDraw.

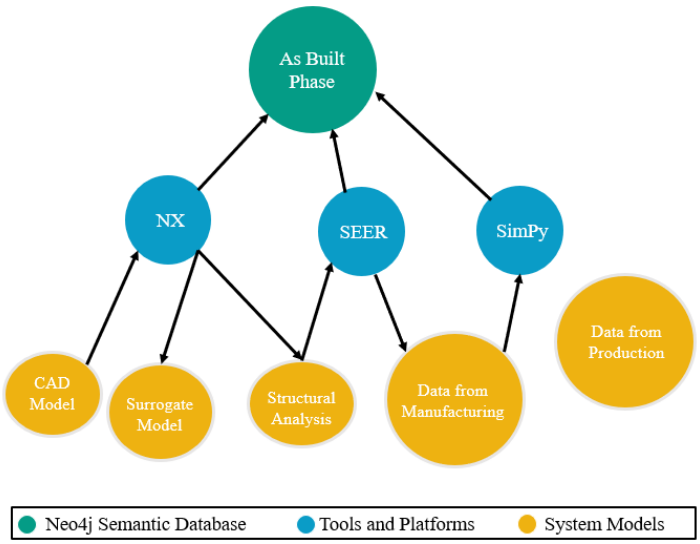


Figure 57. As Built Data Infrastructure

Digital Enterprise Across the Lifecycle

Putting together the data infrastructure for the *As Designed*, *As Built*, and *As Used* phases (developed in 2019-2020) enables a digital enterprise across the lifecycle, both for visualization purposes and in Neo4j, as shown in Figure 58 below.

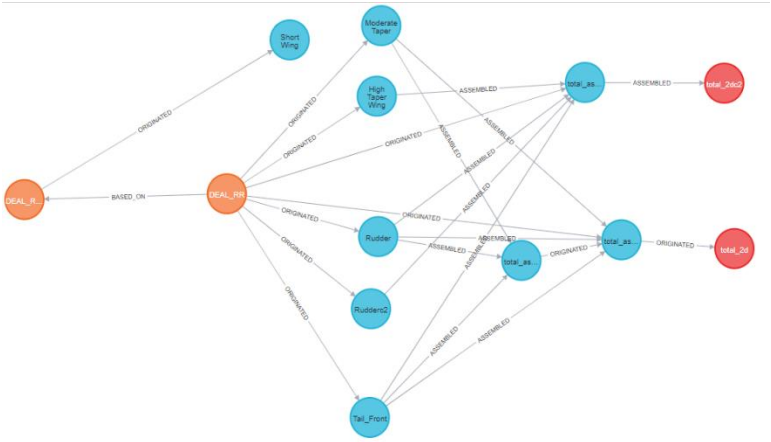


Figure 58. Digital Enterprise Across Lifecycle in Neo4j (*As Designed*)

In Neo4j, under the *As Designed*, the orange nodes are the requirement models, such as the MagicDraw files. The blue nodes are the design models, which represent the NX CAD model or a MATLAB file specific to that version. The red nodes are the manufacturing models, where the SEER and SimPy outputs can be classified.

The following section discusses the results obtained and capabilities developed following the implementation of the methodology and integration of the various tools and models developed as part of this effort.

VI. RESULTS & DISCUSSION

The following sections detail the results from the structural analysis, the manufacturing model, the production model.

Following the completion of the structural, manufacturing, and production analyses, trade studies were completed through a Design of Experiments (DoE) for each domain. Additionally, the requirements were traced and addressed concurrently through the MBSE environment to ensure that requirements are satisfied. Finally, insight from the ability to concurrently trade against design, structural, manufacturing and production considerations were obtained and are explicitly discussed at the end of this section.

Structural Analysis DoE

Three wing models were modeled and tested:

- All-Aluminum Wing: Parts made in Al 6061 (yield point of 35 ksi)
- Al-Steel Wing: Steel 1005 ribs (yield of 84 ksi), all other parts in Al 6061
- All-Composite Wing: Parts made from Carbon Fiber Reinforced Polymer (yield point of 20ksi due to accounting for possible manufacturing defects)

A spacing filling design with 100 cases was created in SAS' JMP[1] in order to determine structural tradeoffs for varying inputs. The different cases were selected such that the main effects and 2nd and 3rd level interactions would be captured. The metallic cases took about 4 hours each, with none of the 100 cases failing. The composite cases also took about 4 hours, with none of the individual cases failing as well. All of the input design parameters, which include the half-span, number of ribs (and by proxy the rib spacing), and rib thickness, were varied. All of the spars were fully constrained on one end and a vertical load of 10 lbf was geometrically distributed over the bottom face of the ribs.

The metallic and composite wings, due to an unknown reason, have extremely high stress at some stress riser locations in many of the cases. These very high stresses are deemed un-physical, being several orders of magnitude higher than all other stress values in the structure, and are likely singularities caused by minute computational errors. As a result of this, the 98th percentile stress is reported as the max Von Mises stress instead (with the 100th percentile being the actual, and un-physical, max stress).

Overall, due to the vertical load only being 10 lbf, all of the cases for all of the materials are overbuilt and none come close to their respective materials' yield stresses.

To ease the process of obtaining stresses without having to spend large amounts of time rerunning the structural models in the future, surrogate models were developed using the DOE results. Artificial neural networks (ANN) were used to create surrogate models for this and aid in the quick probing of parameters and results. JMP offers three activation functions for nodes: TanH, Linear, and Gaussian. Selecting the number of nodes and their associated activation functions is a balancing act between speed and accuracy. The best surrogate model was obtained by using 5 nodes for each of the activation functions.

[1] https://www.jmp.com/en_us/home.html

After the DoE was completed, a measure of goodness was done to see how well the data fit the model. The “residual by predicted” plot depicts the error associated with the RSE for each predicted value. A good “residual by predicted” plot will result in a random scattering of the data points about zero with no distinguishable pattern and a small magnitude relative to the predicted value.

The R^2 for the validation set was generally very good, being on the order of 0.95 on average, which demonstrates a relatively good fit for the model. The residual error is close to zero, few patterns were observed, and the error is generally small, averaging about 3% and with a max of about 8%.

Results from the Manufacturing Models

The manufacturing models output the individual manufacturing processing times for each manufacturing process based on the structural model information. Additionally, the models output the required raw material for each process and the related monetary costs. This information is output to a .csv file, which is then passed to MagicDraw. Because there are four different processes to be simulated, a total of four separate models was needed, one for each process. An example of these outputs can be seen in Table 15, which shows all of the types of outputs for some of the individual tasks in each process. The particular process depicted is that of the All-Aluminum one.

Table 15. Example output file from the manufacturing model for All-Aluminum process

Name	Total Cost/Unit	Material Cost/Unit	Tooling Cost/Unit	Total Labor Minutes/Unit
UpperSkinForm	121650.78	4.28	121457.7	113.28
UpperSkinMachining	609.42	0	159.86	269.74
LowerSkinForm	121650.78	4.28	121457.7	113.28
LowerSkinMachining	592.12	0	159.86	259.36
RibMachining	20127.07	1.11	20099	16.18
RibFinishing	546.31	3.02	442.12	60.71

Results from the Production Model

The production model outputs the factory throughput over a set time, process time, and workstation utilization rate. For SimPy, the information is printed out in the command line (although this is not shown when running the production model through MagicDraw) and saved to an output Excel file. Additionally, the production model outputs the machine utilization rates as figures (Figure 59) within the Python script’s path.

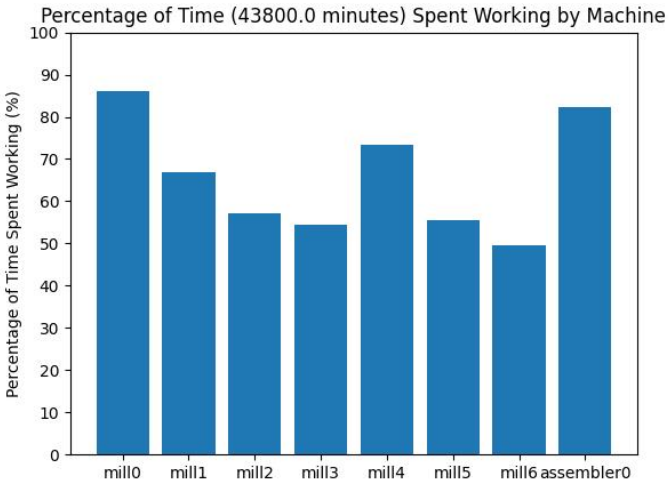


Figure 59. Example machine utilization figure output by the production model.

For Simio, the output information is stored in two separate Excel files, one with the throughput information and the other with the utilization information. A snapshot of the throughput and machine utilization rate results for a few tasks is shown in Figure 60 and Figure 61. Mean, median, max, and other related statistical results are outputted because the production model is stochastic program and consequently does not yield the exact same results from run to run. Therefore, multiple runs are done for each individual case in the DOE such that the results are able to converge on a single value. The number of runs required for such a convergence for each case is about 20. Median values for both types of outputs are used. A bar chart similar to the one produced by SimPy is not outputted due to the sheer unreadability of trying to display the bars for each of the many stations in the Simio models. Because each task/station has a production rate, the actual throughput is obtained by determining the task/station with the lowest throughput, also called the bottleneck. The throughput of the bottleneck station is the throughput for that particular DOE case.

Scenario	Response	Mean	Median	Minimum	Maximum	HalfWidth	Mean Confidence Interval Start	Mean Con	Upper Per	Upper Per	Upper Per	Lower Per	Lower Per	Lower Per
RibHLU2_80	SRUSkin_Fasten_FinalAsser	76.04745224	77.1591362	72.03126	78.75480935	1.720559965	74.32689228	77.76801	77.63686	NaN	NaN	73.84895	72.03126	77.15914
RibHLU2_80	SRUSkin_Drill	76.85131821	77.9656253	72.64244	78.66724399	1.596393439	75.25492477	78.44771	78.36685	NaN	NaN	74.9703	72.64244	77.96563
RibHLU2_80	SRS_Fasten	78.20521662	79.5805148	73.95275	80.3192984	1.653143005	76.55207362	79.85836	79.67576	NaN	NaN	76.46863	73.95275	79.58051
RibHLU2_80	SR_Drill	26.85917023	27.0394588	26.0153	27.33837265	0.287563705	26.57160652	27.14673	27.09078	NaN	NaN	26.70386	26.0153	27.03946
RibHLU2_80	ASpar_Trim	9.118419906	8.96303674	7.806927	10.52392353	0.703868701	8.414551205	9.822289	10.2504	NaN	NaN	8.470601	7.806927	8.963037
RibHLU2_80	ASpar_Debag	11.4890095	11.3410611	9.978797	13.22007966	0.857559096	10.6314504	12.34657	12.88393	NaN	NaN	10.59044	9.978797	11.34106
RibHLU2_80	ASpar_Bag	4.303430132	4.26700113	3.914549	4.720791965	0.208880114	4.094550018	4.51231	4.628398	NaN	NaN	4.087065	3.914549	4.267001

Figure 60. Machine Utilization Rates for Select Tasks

Scenario	Response	Mean	Median	Minimum	Maximum	HalfWidth	Mean Con	Mean Con	Upper Per	Upper Per	Upper Per	Lower Per	Lower Per	Lower Per
Scenario3	SRS_Fasten	40	40	40	40	0	40	40	40	NaN	NaN	40	40	40
Scenario3	SR_Drill	41	41	41	41	0	41	41	41	NaN	NaN	41	41	41
Scenario3	ASpar_Trim	95.8	98	86	104	5.122763	90.67724	100.9228	102	NaN	NaN	88	86	98
Scenario3	ASpar_Debag	95.8	98	86	104	5.122763	90.67724	100.9228	102	NaN	NaN	88	86	98
Scenario3	ASpar_Bag	145.8	148	136	154	5.122763	140.6772	150.9228	152	NaN	NaN	138	136	148
Scenario3	ASpar_HLU	145.8	148	136	154	5.122763	140.6772	150.9228	152	NaN	NaN	138	136	148
Scenario3	Final_Assembly	39	39	39	39	0	39	39	39	NaN	NaN	39	39	39

Figure 61. Throughput for Select Tasks

Results from the Requirement Verification

The purpose of verifying requirements during the design process is to determine whether or not a system is able to meet its system-level requirements. There are four fundamental methods of verification that can be used -- Inspection, Demonstration, Test, and Analysis. These methods are somewhat hierarchical in nature, as each verifies requirements of a product or system with increasing rigor. Since this project focuses on delivering a minimum viable product, the following table shows preliminary results from verification and validation of design, manufacturing, and production requirements using the data from analysis in MagicDraw.

Table 16. Verification of Requirements

	Requirement ID	Requirement	Met (Y/N)
Design	DR01	The aircraft's structure shall remain integrous under a load factor of up to 5 g.	N
	DR06	The Main Spar thickness shall not exceed 0.13 inches	Y
	DR07	The wing total weight shall not exceed 4lb	Y
	DR08	The maximum von Mises shall not exceed 35,000 psi	Y
Production	PS01	The Production System shall have a takt time of less than 3 days.	Y
	PS02	The Production System shall have a square footage of less than 800,000 square feet.	N
	PS05	The Production System shall have an on-time delivery percentage of at least 90 %.	N
	PS06	The Production System shall have the capability to produce at least 40 wings per month.	Y
Manufacturing	MS03	The Manufacturing System shall produce at least 40 wings every month.	Y
	MS04	The Manufacturing System shall manufacture a wingspan that does not exceed 36 inches.	Y
	MS05	The aircraft take-off gross weight with payload shall be less than [TBD] kgs.	N
	MS07	The Manufacturing System shall minimize defects to less than 10%.	N
	MS14	The Manufacturing System shall have tooling costs no greater than $100,000.	Y
	MS16	The Manufacturing System shall be able to maximize machine capacity / utilization	Y
	MS17	The Manufacturing System shall manufacture an aircraft wing no greater than $45,500	Y

Of the 15 requirements that had to be verified and validated, 10 requirements were satisfied overall. The other requirements are related to the overall structure of the program, which cannot be satisfied until operations were to occur for the design considered as part of this study.

Conducting Trade Studies

Trade studies were conducted in a parametric tradeoff environment developed in JMP to allow the user to see how changes in design inputs affect key manufacturing and production outputs.

For each alternative (the aluminum wing, the aluminum + steel wing, and the VARTM and autoclave wings), 100 cases were created with varying wing geometry and factory settings to demonstrate the impact of these inputs on the manufacturing processes, design configuration, throughput, etc. Specifically, the tradeoffs are conducted between design (geometry), material,

factory layout vs throughput, wing cost, material cost and tool cost. The results are presented in the following sections.

Structural and Manufacturing Trades

For the structural and manufacturing trades, three types of wings were considered:

- All-Aluminum Wing, where parts are made with Al 6061 (yield point of 35 ksi)
- Aluminum-Steel Wing, where the ribs are Steel 1005 (yield of 84 ksi), and all other parts are Al 6061
- All-Composite Wing, where parts are made from Carbon Fiber Reinforced Polymer (yield point of 20ksi due to accounting for possible manufacturing defects)

The manufacturing inputs include the wingspan, rib spacing, rib thickness, and quantity of ribs. The ranges for these values are: 26 – 36'' for the wingspan, 1.6 – 3.2" for the rib spacing, 0.16 – 0.3" for the rib thickness, and 6 – 8 for the quantity of ribs. Additionally, the minimum weight range is 0.7 lbs. for the composite wings, 1.93 lbs. for the all-aluminum wing, and 3.25 lbs. for the aluminum-steel wing.

The results are plotted in JMP to analyze the trends between the weight, cost, and maximum Von Mises against the design variables for different trades. The results from this trade study are shown in Figure 62 to Figure 66. The blue crosses represent aluminum-steel wing designs, the black dots represent all-aluminum wing designs, the red triangles represent composite VARTM wing designs, and the green squares represent composite autoclave wing designs.

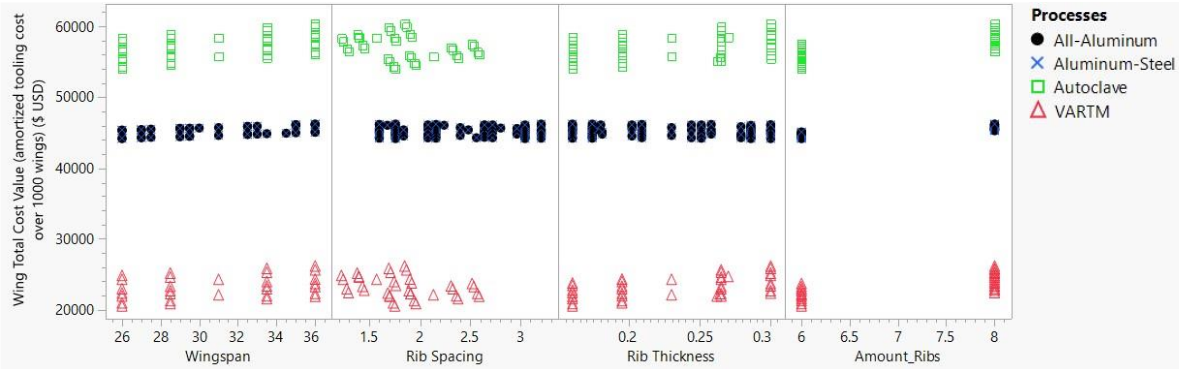


Figure 62. Total Cost Value (tooling cost amortized by 1000 wings)

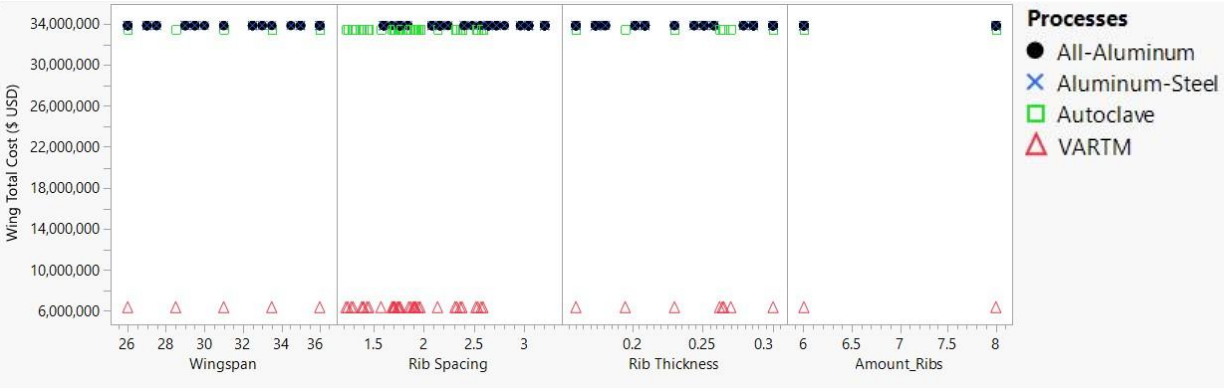


Figure 63. Total Cost per Wing

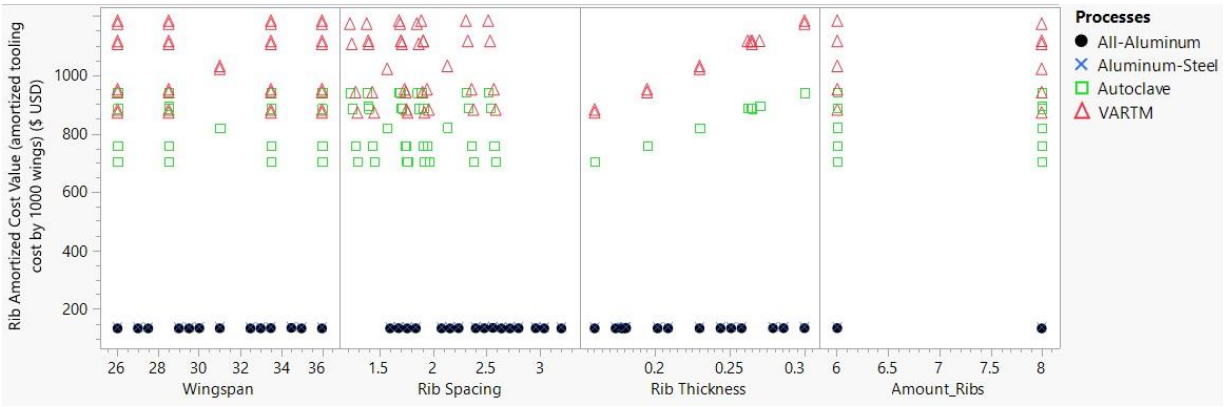


Figure 64. Amortized Cost of a Rib Per Wing

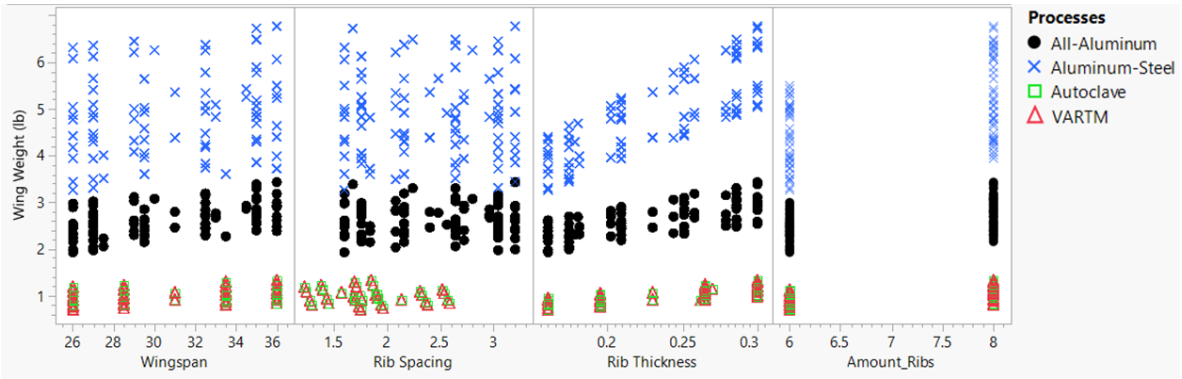


Figure 65. Wing Weight

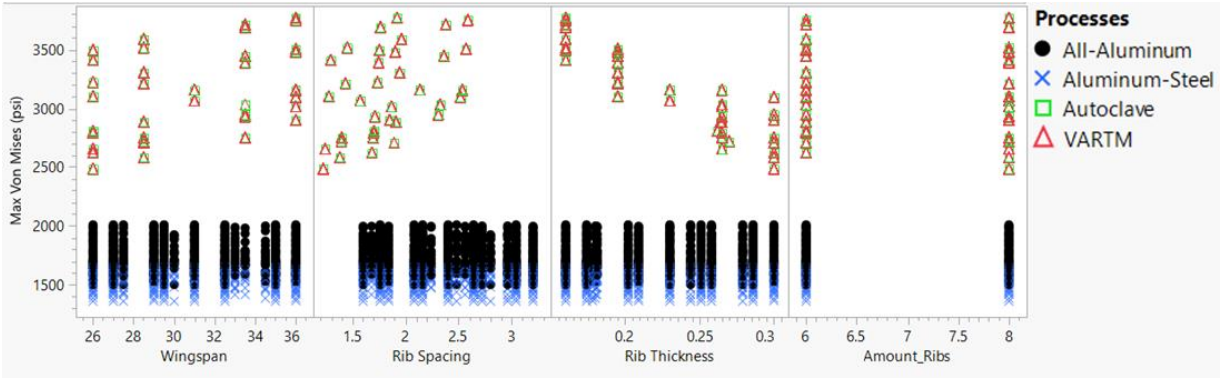


Figure 66. Wing Structural Analysis

The aluminum-steel wing total cost will always be around 0.01% more than an all-aluminum wing. As expected, this means that the difference arises from using steel to manufacture the ribs instead of aluminum. It is also assumed that the same set of tooling will be used for the entire production run. Therefore, it costs slightly more to produce an aluminum-steel wing than all aluminum wing (cost is around $45,000) since set-up time drives labor time and tooling cost overshadows other costs due to the small and simple wing geometry while using industrial equipment.

The composite wings do not always necessarily cost less than the metallic ones. In fact, the autoclave composite wings can cost just as much if not more than the metallic ones. This is because the assembly portions of the metallic and autoclave composite processes involve the same activities, just performed on different materials. Comparing the costs in Figure 62 and Figure 63, it can be seen that the tooling of the metallic processes is slightly more expensive than the autoclave composite processes, hence the metallic ones having a lower cost when tooling is amortized. Similarly, the actual labor cost of the composite process is higher, causing the tooling-amortized cost to be higher. Overall, the autoclave composite process has a lower total cost due to the overwhelming expense of metallic tooling, with the VARTM composite process having the lowest cost of all processes due to its assembly activities not involving drilling and fastening.

With regards to the structural analysis of the wing, the wingspan affects the wings' structural response in different ways due to the different materials. For instance, larger wingspans directly lead to larger weights, but only lead to higher stresses for the composite wings. It is possible that the connection between the parts may be producing the differences. Therefore, alternative connection types should be investigated and used to create new trades. Additionally, it is expected that steel ribs will produce smaller deformations than the all-aluminum wing. Figure 65 and Figure 66 demonstrate that, although an all-aluminum wing is significantly lighter, it deforms more than an aluminum-steel wing and has higher regions of stress. On the other hand, an aluminum-steel wing is significantly heavier, which lowers the performance of the wing. The composite wings are by far lighter than the metallic ones, up to three times lighter than the aluminum wing and almost a whole order of magnitude lighter than the aluminum-steel wing, indicating that composite wings far exceed both metallic wings structurally if all the parts share similar dimensions. It does this while sustaining just a little more stress that is still nowhere close to its yield stress.

Overall, the manufacturing trades show the aluminum-steel wing to be slightly inferior to the aluminum ones. The fact that it experiences lower stresses on average is of questionable benefit since the overall stresses are already incredibly low. This is done at the cost of being more

expensive and heavy because some of its components are made of steel. Pure aluminum is therefore the superior of the two metallic materials. Both sets of metallic wings, however, are significantly heavier and generally more costly than composite wings, in particular the VARTM ones. While there can be some debate with regards to whether a metallic or autoclave wing is superior, depending on what objectives the designer or customer values most, the composite wing manufactured using VARTM almost unequivocally represents the best option: it weighs the least, costs the least, and is very structurally sound while doing so.

To test the sensitivity of the tooling cost on the amortized total cost value of the wing, three different number of wings to be produced are used: 500, 1000, and 2000 wings.

Production Trades

For the production trades, 6 different station configurations (Table 17) and the 400 cases from the manufacturing analysis were used. The configurations involve changing the number of duplicate stations. Duplicating stations increases a station's throughput by a factor equal to the number of its duplicates, which improves total throughput if a bottleneck station is duplicated. "Main Part Stations" is used to indicate the number of duplicate stations for each of the manufacturing processes. A more specific station-naming convention, e.g., number of rib machining stations, was not used because the equipment and machinery used by each manufacturing process is different. This was also done because it desired to use only one set of inputs for all the different processes to minimize the otherwise gargantuan number of variables to be inputted. This means that "Main Rib Station" refers to a different station type for a metallic process versus a composite process. The specific station types chosen to be duplicated for each manufacturing process were selected based on which types were most likely to be bottleneck stations. As a common input is being used for four very different manufacturing processes, the station types were also chosen based on their conceptual similarity with one another, e.g., the act of machining out a metallic rib involves laying down and forming material, which is conceptually equivalent to the hand layup process for composite ribs. Rib machining and rib hand layup station types are thus favored for their respective processes as opposed to, say, rib finishing and rib hand layup. With these factors in mind, the main rib, spar, and skin stations refer to machining stations for metallic processes and hand layup stations for composite processes. Fitup stations are functionally identical between all the processes and so are the assembly station type of choice to duplicate.

Table 17. Machine Configuration

Configuration #	# Main Rib Stations	# Main Spar Stations	# Main Skin Stations	# Fitup Stations
1	1	1	1	1
2	1	1	1	2
3	1	2	2	2
4	2	2	2	2
5	2	2	2	3
6	3	3	3	3

As mentioned before, the production model runs 20 iterations per configuration and scenario to determine the number of components made per machine. Furthermore, the median factory throughput and standard deviation is calculated for the number of wings made. This process is then automatically repeated for each output SEER file (a scenario) that has been provided to the production model.

The full results of the factory throughput based on the factory configurations are shown in Figure 67.

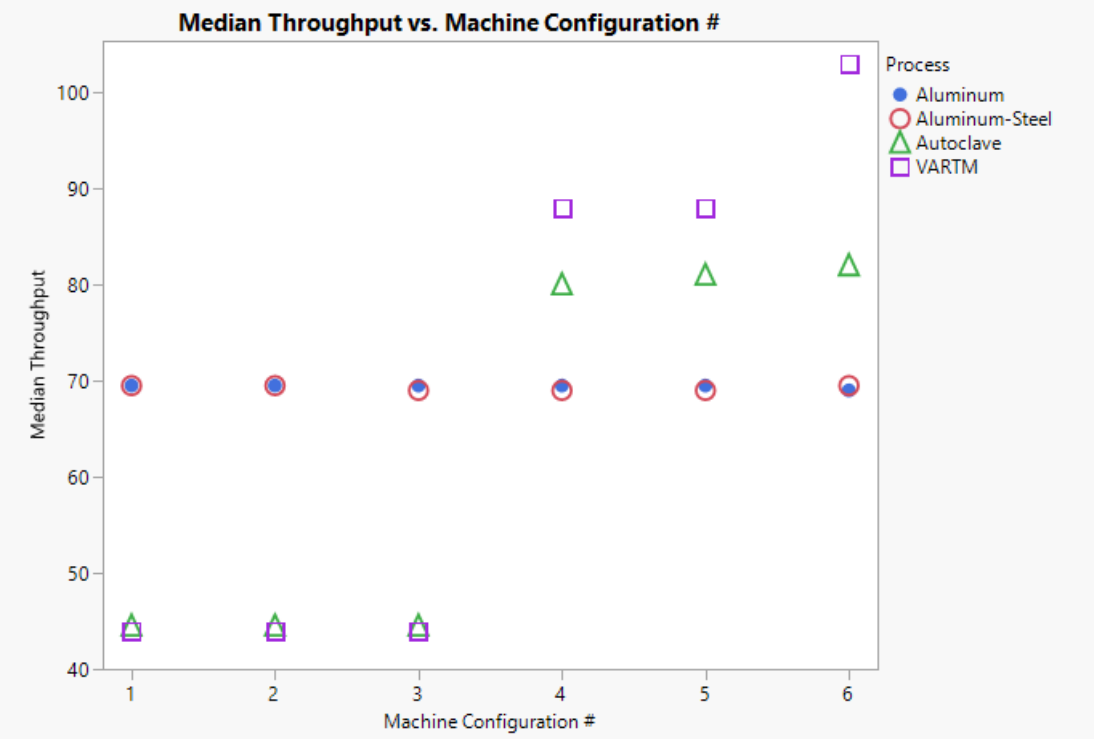


Figure 67. Median factory throughput over one month of production

The metallic processes have a higher throughput than the composite ones in the first several machine configurations because the latter have a bottleneck in the rib layup stations for those configurations. This is due to rib layups taking significantly more time to complete than rib machining tasks. However, as more rib layup stations are added, this bottleneck is alleviated and the composite throughput rises above that of the metallic ones. The autoclave process experiences a non-hand layup and non-fitup related bottleneck (which are the only stations affected by changes in machine configuration in this iteration) after configuration 5, which is why its throughput stagnates while VARTM throughput continues to increase. The metallic processes did not see an increase in throughput at all because their bottleneck stations are the drilling and fastening related ones, which were not changed when changing machine configurations, due to the aforementioned desire to affect only stations that were present in all the processes.

Full Results

With the results from the structural, manufacturing, and production trades, a comprehensive representation of the effect of different factors on throughput, wing weight, wing cost, and max Von Mises are created. The metallic wing-only representation is depicted first, as shown below.

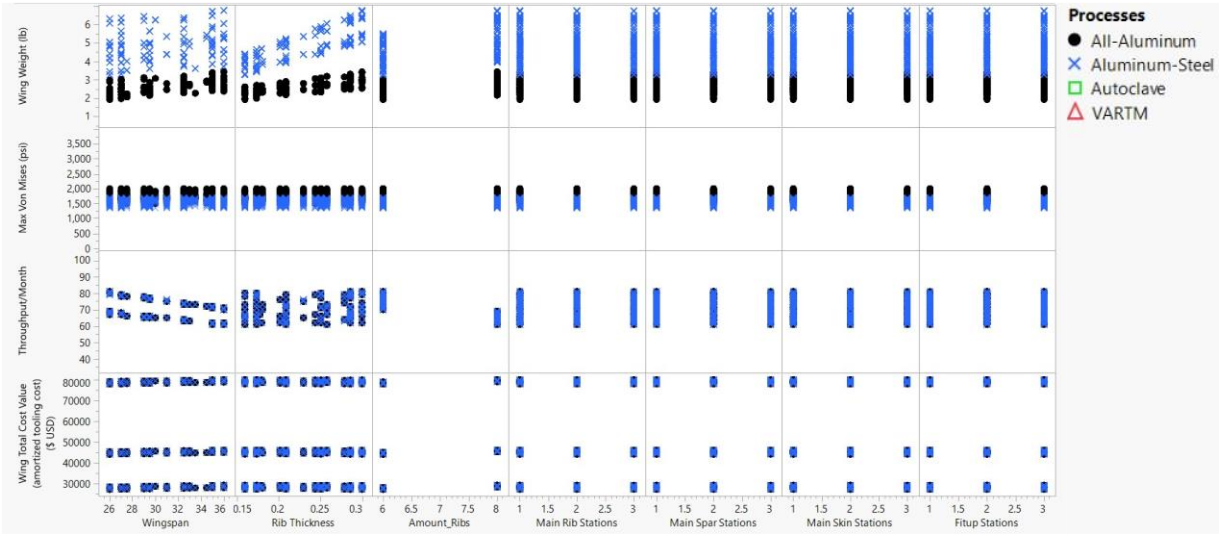


Figure 68. Comprehensive Trades, Metallic Processes

Larger wingspans incur larger weights, lower throughput, and slightly higher costs for both metallic processes, as is expected, due to the larger amount of material required. Thicker ribs exhibit a similar behavior for wing weight, though the effect of thicker ribs on the other outputs is less concrete. In a similar manner, larger numbers of ribs cause the wing weight to go up, the throughput to go down, and the amortized cost to increase a little. Neither wingspan, rib thickness, nor number of ribs had any significant effects on max stress. This is likely because the metallic wings are so overbuilt, given the wing dimensions used, that the parts are structurally saturated and will not experience lower stresses without much larger changes in the dimensions. The overall trends thus indicate that smaller wings with fewer, thinner ribs are more likely to meet all the requirements imposed. None of the machine configurations yielded any effects because none of the configurations examined targeted the bottleneck stations for the metallic processes.

The comprehensive trades for the composite processes are depicted next and presented below.

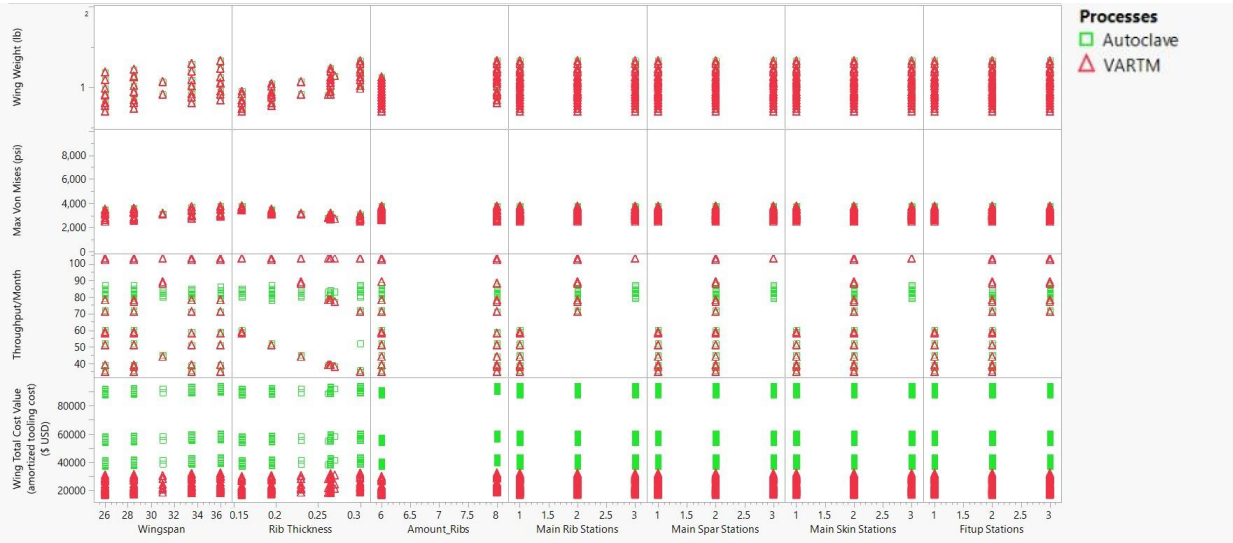


Figure 69. Comprehensive Trades, Composite Processes

The trends for the composite processes essentially match those of the metallic ones, though to a slightly different degree. There is a more notable decrease in throughput with an increase in rib thickness. The rib thickness also has a more noticeable effect on max stress, with thicker ribs able to bear larger loads or, for the same load, experience lower stresses. In addition, the machine configurations do have a more marked impact on throughput, primarily the number of rib stations. This is because the rib hand layup task is the bottleneck station for both autoclave and VARTM processes, allowing an increase in the appropriate number of stations to dramatically increase the throughput.

The figure below depicts the comprehensive trades with all the processes included.

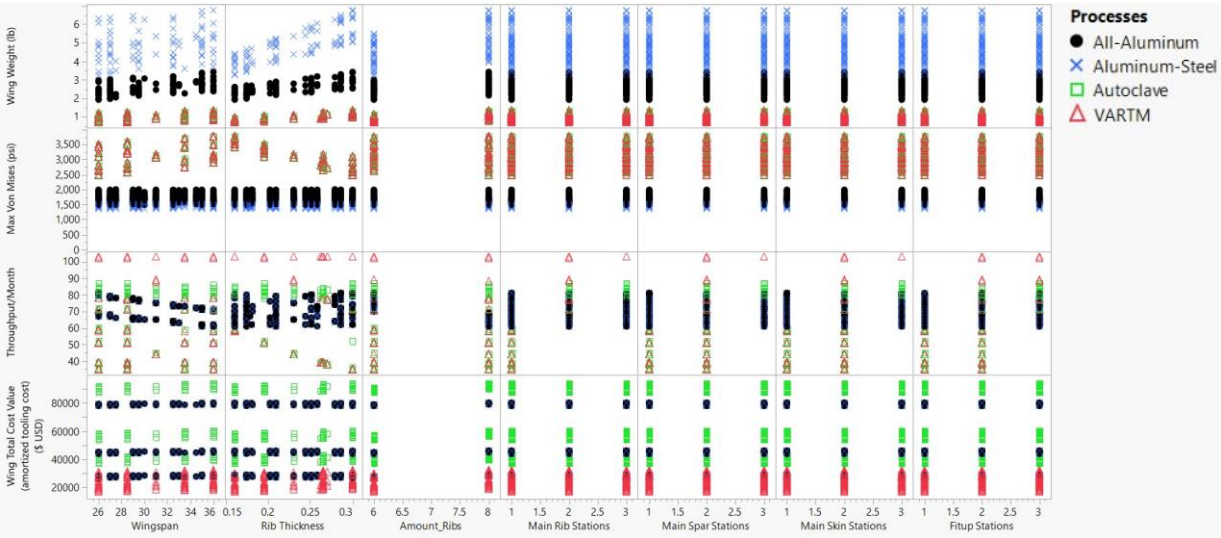


Figure 70. Comprehensive Trades, All Processes

One notable observation from Figure 70 is that the wing cost is not affected at all by the machine configuration. This is because in the context of this research, equipment costs are not accounted for due to there being no cost data for any of the equipment being used. Another is that the autoclave process costs about as much, if not more, than the metallic ones despite being a composite process. This is due to the fact that the assembly tasks for the autoclave process are the same as for the metallic processes and include drilling and fastening. The tooling for these tasks is extremely expensive, causing those processes to be similarly expensive. The VARTM process, on the other hand, has very low tooling costs due to the paste bond assembly task not being as dependent on tooling. This thus causes it to be less expensive as a whole.

With the application of requirements, the scatterplot matrix was filtered (Figure 71) to determine the feasible design space. The filters are that the wing must weigh under 4 lbs., have an amortized cost of less than $45,000 per wing, have a throughput of greater than 40 wings per month, and have a max (98th percentile) Von Mises stress of less than 20 ksi for composites, 35 ksi for aluminum, and 84 ksi for steel.

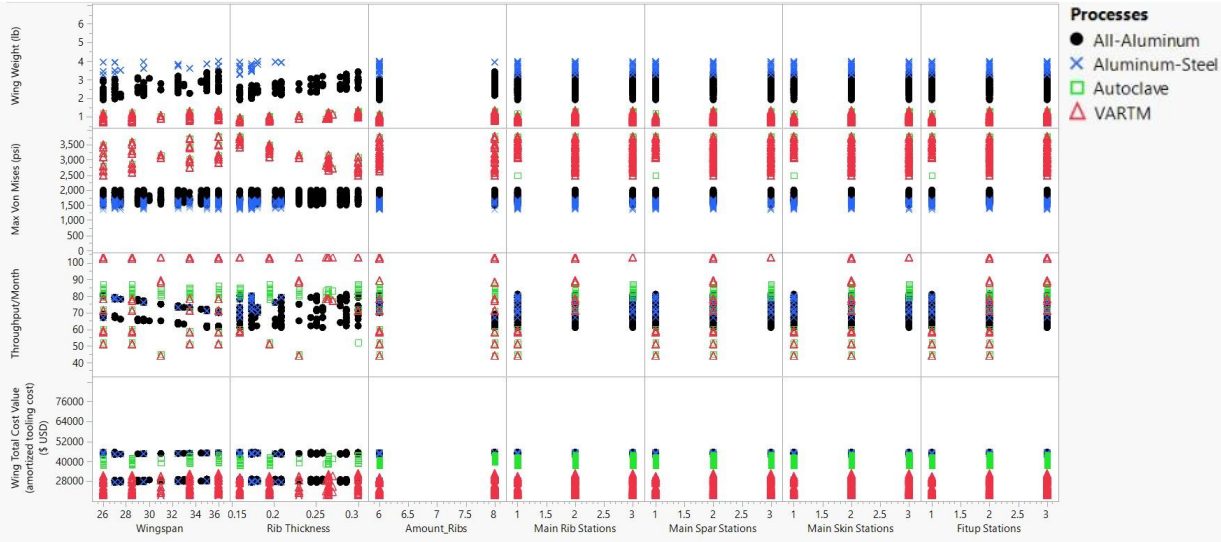


Figure 71. Filtered Trades

From the filtered trades, about 43% of the original 7200 possible configurations were feasible. When applying the requirements, the design space reduced by 20% when applying the wing weight requirement, primarily eliminating the metallic wings. The design space is unaffected by a applying a maximum Von Mises requirement since none of the wings are close to their yield stresses: a larger wing load will need to be applied in the next iteration of this work for it to make a difference. The design space is reduced further by 14% when applied the production throughput (wing/month) requirement. The feasible design space is reduced an additional 38%, to only about two-fifths of its original size of 7200 possible configurations, when the cost requirement is applied. The metallic designs constitute about 40% of these feasible designs, while roughly the same amount of the remaining designs are with the VARTM manufacturing process. The remaining use the autoclave process. Thus, respectively, 32%, 8%, 15%, and 45% of the remaining designs use the pure aluminum, aluminum-steel, autoclave, and VARTM processes.

In the context of the requirements considered, the results suggest that composite wings manufactured using VARTM are preferred. They weigh significantly less, do not experience significantly larger stresses, and, at least in the VARTM's case, also incur lower tooling costs. For the metallic wings, the rib material played an important role in which designs were feasible because, while the Al-Steel wings experience lower stresses, they also weigh significantly more than the full Al wings and so are overall less optimal; there are about 4 times more feasible full Al wings than Al-Steel ones. The machine configuration did not have a significant impact since the general throughputs were already so high and so met the requirements already.

Discussion

The overarching expected outcome of this research was to build upon the existing digital enterprise proof-of-concept to demonstrate the seamless integration of design, manufacturing, and production, in the *As Designed* and *As Built* phases of a product's lifecycle. Some key objectives were to trace and verify requirements, develop architectures and frameworks for digital manufacturing, illustrating the integration of system models with other analytical tools, and developing a link between manufacturing capabilities and design.

The team, building upon the existing infrastructure, created models for tracking, verifying, and validating requirements, activity diagrams, and use case diagrams with the MBSE environment MagicDraw, and connected MagicDraw with external tools such as NX, SEER-MFG, Simio, and SimPy to conduct analyses. In the design phase, the system structure was translated into a 3D design in NX, where specified inputs, like the geometry, were used for structural analysis. The output was then fed to the manufacturing model. In the manufacturing phase, manufacturing models for metallics and composite wings were developed and their outputs fed to inform production models developed in SimPy and Simio. The results from all those analyses were aggregated into a parametric tradeoff environment, hence allowing the users to 1) instantaneously assess the impact of varying design inputs on key manufacturing and production outputs, 2) quickly visualize high level trends, and 3) identify design configurations and factory settings that lead to a desired throughput or satisfy a range of constraints.

Table 18 describes the models developed and available as a result of this research.

Table 18. Technology Deliverables

#	DELIVERABLE NAME	DESCRIPTION	FORMAT OF DELIVERY
1	Final Technical Report	A comprehensive, cumulative, and substantive summary of all technical advancements and significant accomplishments achieved during the project. Includes a detailed documentation of the methodology developed and results obtained as well as user resources to run each model developed	Electronic (MS Word and pdf documents)
2	Final Technical Presentation/Demonstration	The final presentation includes a comprehensive, cumulative, and substantive summary of all technical advancements and significant accomplishments achieved during the project, along with a demonstration of the methodology developed.	Electronic (PowerPoint presentation)
3	Modeling Environment	Includes relevant models, data and script developed as part of the effort. In addition, the folder includes ReadMe files for each model (system model in MagicDraw, structural models in NX, manufacturing models in SEER-MFG, production models in Python and Simio)	Electronic
4	Transition Plan	Written plan for successful transition of project outcomes after period of performance including distribution and follow-on efforts for phase(s) 2 & 3. Desired future industry partners are clearly identified with a plan of action for future participation, if/when relevant	Electronic (Word document
5	Educational Impact	Documentation on course/lab module presented to students demonstrating the use of technology to increase awareness in topic area will be provided. Made available to MxD Learn for future educational content.	Electronic (PowerPoint presentation and Word document)

This research creates a foundation for future accomplishments to further the complexity of the digital enterprise, connect the phases of the lifecycle to other products, and conduct trade studies. In the current context, this research tested and evaluated various integration paths, and applied them to a digital manufacturing vision for a fully interconnected digital enterprise.

The satisfaction of the Key Performance Indicators (KPI)/metrics presented in Table 19 are indicative of the success of the methodology developed.

Table 19. KPI/Metrics used to Validate Successful Technology Development and Deployment

#	KPI/Metric	PAST STATE	CURRENT STATE
1	Increased traceability: requirements are linked to truth data and models	An authoritative source of truth (ASOT) existed that integrate requirements, models and data as they relate to the “As Designed”, “As Built” and “As Used” phases	This ASOT was expanded to include the manufacturing/production requirements, models and data generated as part of this research. The outcome is the full digital traceability of requirements at each phase considered
2	Increased capacity for reuse		The team leveraged and expanded on the models, data, and infrastructure developed in (Kargin et al., 2021)
3	Knowledge transfer		The team demonstrated the ability to share and jointly access models, data, and information; ability to learn around common tools and representations

Because the accomplishments of this research were a foundation to show basic functionality of a digital enterprise, there is room for further development in complexity and future work. This is addressed in the following section.

VII. CONCLUSIONS & FUTURE WORK

This research successfully developed and implemented a methodology aimed at capturing manufacturing and production system considerations in a model-based environment, hence allowing for the connection and integration of product design, manufacturing and production models and data through an authoritative source of truth (ASoT).

In the As Designed phase, the team accomplished requirement decomposition, tracking, and verification in MagicDraw. The various use cases pertaining to design, manufacturing, and production as well as activity diagrams were modeled. Structural analysis models for both metallic and composite wings were developed and integrated with the system model. In the As Built phase, manufacturing and production models were developed and integrated with the system model.
The results of these analyses were integrated within a parametric environment in JMP as a means to demonstrate the ability to 1) instantaneously assess the impact of varying design inputs on key manufacturing and production outputs, 2) quickly visualize high level trends, and 3) identify design configurations and factory settings that lead to a desired throughput or satisfy a range of constraints. Such capabilities are critical to the ability to influence the design early in the design phase of a program’s lifecycle.

In addition, all models and data generated were stored in a graph-database based on Neo4j that serves as a digital thread. This enables version tracking and identification of gold layer data, as well as data ownership. Overall, this research demonstrated a working digital enterprise infrastructure, for digital manufacturing, as a minimum viable product for the design, manufacturing and production of metallic and composite aircraft wings.

Next Steps

Parametric Structural Model

The parametric structural model can be improved to reduce stress risers, such as 1-dimensional connectors between edges. Additional capability needs to be added to allow for modifications to the shape of the rib to be done. This will simplify the process of making changes to the main spar's thickness**.**

Production Model

It is currently difficult to determine bottleneck stations a priori and therefore difficult to determine which stations should be duplicated for meaningful throughput improvement when changing the machine configurations. Although the initial bottlenecking stations can be relatively easily determined a priori based on the manufacturing times, subsequent ones after the initial bottleneck is cleared are not as simple to discern due to the dynamic nature of the model. One of the main focuses in the future is thus to implement more formal line balancing that can be done in the middle of the Simio runs, while stations are duplicated, to know which ones should be duplicated on the fly. This would also need to be done in such a way as to avoid requiring a large number of additional variables in Simio, i.e., done in a way as to avoid being forced to have a variable for every possible task or station, since potentially any of them could eventually become a bottleneck.

Model validation using Flight Data

The airplane could be instrumented and sensing data (e.g. structural health data) collected during a flight test campaign to help validate the results of the structural analysis model, for example.

Challenges

The main challenges faced throughout this research are summarized below:

- **Tool Integration with System Model:** the realization of the MBSE model as a single source of truth has proven challenging and may not be realistic given the sheer diversity in analysis tools used by the industry. A logical next step would involve investigating third-party integration platforms/solutions to pass information from one tool to another.
- **Communication:** communication across team members working on each aspect of the project is key. The documentation of the hypotheses made at each level of the analysis is critical to ensure consistency in the results and the conclusions.
- **Consistency Management:** consistency management is a central paradigm for Digital Engineering (Schindel, 2020) but one that is difficult to achieve without the proper definition and architecting of the Digital Thread.

Transition Plan

The transition plan should contain a concise statement explaining the transition activities for each deliverable. Deliverables that will be transitioned through multiple routes must have a transition statement for each of those activities.

The transition plan for each deliverable should be summarized in a table as shown below. Deliverables can transition through deployment at an industry member's site, as an educational reference or through a commercialization effort.

The table below provides a catalog of all of the project deliverables and their respective transition routes. Deliverables can transition through deployment at an industry member's site, as an educational reference or through a commercialization effort. Each of these transition routes are detailed below.

Table 20. *Deliverable Deployment Summary*

#	DELIVERABLE FILE NAME	TECHNOLOGY INTEGRATION	EDUCATION	COMMERCIALIZE
1	Final Technical Report	X	X	
2	Final Technical Presentation/Demonstration	X	X	
3	Modeling Environment	X	X	
4	Transition Plan		X	
5	Educational Impact		X	
6	Quarterly Technical Review		X	

Deliverable 1 – Final Technical Report

- Technology Integration: Rolls-Royce will disseminate the Final Technical Report through its Engineering Organization to illustrate the project Proof-of-Concept for Model Based Systems Engineering in Digital Manufacturing before applying similar concepts where possible on Defense programs.
- Education: The Final Technical Report will be used to educate Rolls-Royce Digital Engineering stakeholders so that they can better understand the "art of the possible" of MBSE & Digital Manufacturing through this Proof-of-Concept demonstration.

Deliverable 2 – Final Technical Presentation/Demonstration

- Technology Integration: Rolls-Royce will disseminate the Final Technical Presentation/Demonstration through its Engineering Organization to illustrate the project Proof-of-Concept for Model Based Systems Engineering in Digital Manufacturing before applying similar concepts where possible on Defense programs.
- Education: The Final Technical Report will be used to educate Rolls-Royce Digital Engineering stakeholders so that they can better understand the "art of the possible" of MBSE & Digital Manufacturing through this Proof-of-Concept demonstration.

Deliverable 3 – Modeling Environment

- Technology Integration: Rolls-Royce will deploy the Georgia Tech Models to its MBSE team as an example proof-of-concept for on-going learning and development using off-the-shelf software tools such as MagicDraw.
- Education: The proof-of-concept Modeling Environment will be used to educate Rolls-Royce MBSE users so that they can develop deeper understanding of the art-of-the-possible for MBSE integration across the lifecycle.

Deliverable 4 – Transition Plan

- Education: The Transition Plan will be used to educate Rolls-Royce, Georgia Tech, and MxD MBSE practitioners so that they can understand potential uses for MBSE within the context of Digital Manufacturing and the overall Digital Enterprise.

Deliverable 5 – Educational Impact

- Education: The Educational Impact deliverable will be used to educate Rolls-Royce, Georgia Tech, and MxD MBSE practitioners so that they can understand the lessons learned through this project.

Deliverable 6 – Educational Impact

- Education: The Quarterly Technical review, similar to the Final Technical Presentation (Deliverable #2), will be used to educate stakeholders so that they can better understand the "art of the possible" of MBSE & Digital Manufacturing through this Proof-of-Concept demonstration.

Education Plan

Elements of this research were presented as part of AE 6372 – Aerospace Systems Engineering with the purpose to expose students to the benefits and challenges associated with the digitalization of the systems engineering process. In particular, the lecture presented the following:

- the challenges and need to bring "ilities" (e.g. manufacturability, producibility, etc.) earlier in the design process
- the tools, processes and methods aligned with model-centric approaches and the digitalization of the systems engineering process in general, e.g. Model-Based Systems Engineering, as a means to support the digital continuum and traceability between requirements and results of modeling & simulation analyses
- best practices in model-based product design

Eventually, this material and the concepts highlighted herein are envisioned to be integrated into a broader Digital Engineering class and curriculum aimed at ensuring that our undergraduate and graduate students are versed in "Digital".

VIII. ACKNOWLEDGEMENTS

This material is based on research sponsored by Office of the Under Secretary of Defense for Research and Engineering, Strategic Technology Protection and Exploitation, Defense Manufacturing Science and Technology Program under agreement number W15QKN-19-3-0003 between MxD and the Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S Government.

IX. REFERENCES

A. F. Donaldson, M. French, J. Matlik, J. Myer, K. Pond, & R. Randjelovic. (2019). *A View of the "Digital Twin*. AIAA Digital Engineering Integration Committee (DEIC) Digital Twin Subcommittee Briefing.

Agile Alliance. Minimum Viable Product (MVP). Retrieved from https://www.agilealliance.org/glossary/mvp/.

AIAA Digital Engineering Integration Committee. (To be published in 2022). Digital Thread: Definition and Value - An AIAA and AIA Position Paper. In: American Institute of Aeronautics and Astronautics (AIAA).

Akundi, A., & Lopez, V. (2021). A Review on Application of Model Based Systems Engineering to Manufacturing and Production Engineering Systems. *Procedia Computer Science, 185*, 101-108.

Ben Redwood. Types of 3D Printing. Retrieved from https://www.3dhubs.com/knowledge-base/additive-manufacturing-technologies-overview/.

Bretz, L., Tschirner, C., & Dumitrescu, R. (2016). *A concept for managing information in early stages of product engineering by integrating MBSE and workflow management systems.* Paper presented at the 2016 IEEE International Symposium on Systems Engineering (ISSE).

Gottschall, M., Binder, B., Reglitz, S., Saada, H., Couto, L. D., Cremona, F., & Burgio, G. (2018). *Novel Framework Approach for Model-Based Process Integration from Requirements to Verification Demonstrated on a Complex, Cyber-Physical Aircraft System*. Retrieved from https://www.sae.org/publications/technical-papers/content/2018-01-1947/:

Hatakeyama, J., Seal, D., Farr, D., & Haase, S. (2018). *Systems Engineering "V" in a Model-Based Engineering Environment: Is it still relevant?* Paper presented at the 2018 AIAA SPACE and Astronautics Forum and Exposition.

Huang, J., Gheorghe, A., Handley, H., Pazos, P., Pinto, A., Kovacic, S., . . . Rabadi, G. (2020). Towards digital engineering: the advent of digital systems engineering. *International Journal of System of Systems Engineering, 10*(3), 234-261.

INCOSE, A. (2014). A world in motion: systems engineering vision 2025. *International Council on Systems Engineering*.

Kargin, Y. O., Barnes, A. A., Uysal, O. D., Pinon-Fischer, O. J., Balchanos, M. G., Mavris, D. N., . . . Matlik, J. F. (2021). *Digital Enterprise Across The Lifecycle.* Paper presented at the AIAA Scitech 2021 Forum.

Menshenin, Y., Knoll, D., Brovar, Y., & Fortin, C. (2020). *Analysis of MBSE/PLM Integration: From Conceptual Design to Detailed Design.* Paper presented at the IFIP International Conference on Product Lifecycle Management.

Mordecai, Y., de Weck, O. L., & Crawley, E. F. (2020). *Towards an Enterprise Architecture for a Digital Systems Engineering Ecosystem.* Paper presented at the Proceedings of the Conference on Systems Engineering Research (CSER), Virtual, Redondo Beach, CA, USA.

Mourtzis, D., Maropoulos, P., & Chryssolouris, G. (2015). Digital enterprise technology: systems and methods for the digital modelling and analysis of the global product development and realisation process. In (Vol. 28, pp. 1-2): Taylor & Francis.

Myung, S. (2003). *Implementation of the digital manufacturing in automotive industry.* Paper presented at the Dassault Systems Korea Annual Conference.

NikPakvasa (2009). PLM Connection Europe: PLM Journey at Rolls-Royce. Retrieved from https://blogs.sw.siemens.com/news/plm-connection-europe-plm-journey-at-rolls-royce/

Office of the Deputy Assistant Secretary of Defense for Systems Engineering. (2018). *Departmentof Defense Digital Engineering Strategy*. Retrieved from https://man.fas.org/eprint/digeng-2018.pdf:

Paritala, P. K., Manchikatla, S., & Yarlagadda, P. K. (2017). Digital manufacturing-applications past, current, and future trends. *Procedia engineering, 174*, 982-991.

Promyoo, R., Alai, S., & El-Mounayri, H. (2019). Innovative digital manufacturing curriculum for industry 4.0. *Procedia manufacturing, 34*, 1043-1050.

Radtac. *Rolls-Royce: market-leading innovation by introducing Agile Product Lifecycle Management*. Retrieved from https://4bgggq445y9zafll01zckv31-wpengine.netdna-ssl.com/wp-content/uploads/2018/01/Rolls-Royce-Case-study-.pdf:

Reinhard Geissbauer, stefan Schrauf, Philipp Berttram, & Farboud Cheraghi. (2020). *Digital Factories 2020 - Shaping the Future of Manufacturing*. Retrieved from https://www.pwc.de/de/digitale-transformation/digital-factories-2020-shaping-the-future-of-manufacturing.pdf:

Schindel, W. (2020). Consistency Management as an Integrating Paradigm for Digtal Life Cycle Management with Learning. In I. A. S. E. L. C. M. A. Pattern (Ed.), *INCOSE/OMG MBSE Patterns Working Group*. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:aselcm_pattern_--_consistency_management_as_a_digital_life_cycle_management_paradigm_v1.2.2.pdf: INCOSE.

Siedlak, D. J., Pinon, O. J., Schlais, P. R., Schmidt, T. M., & Mavris, D. N. (2018). A digital thread approach to support manufacturing-influenced conceptual aircraft design. *Research in Engineering Design, 29*(2), 285-308.

Singh, M. (2018). Review Paper On Digital Manufacturing & Design.

The MITRE Corporation. *Architectural Frameworks, Models, and Views*. Retrieved from https://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/system-architecture/architectural-frameworks-models-and-views:

Vertex Software. (Accessed October 2, 2019). The Limitations of PLM Software. Retrieved from https://www.vertexvis.com/resources/blog/limitations-plm-software

X. APPENDIX A: USER RESOURCES

This Appendix provides a comprehensive guide for navigating through the Model-Based Systems Engineering, structural, manufacturing, and production models, as well as conducting analysis

Introduction

This user guide is designed to provide documentation on how to use the integrated models and run analysis. There are four sections – Model Based Systems Engineering Environment, Structural Analysis, Manufacturing Modeling and Production Modeling. Each section describes the tool(s) used and how to navigate through the environment.

Model Based Systems Engineering Environment

Purpose

The Model Based Systems Engineering (MBSE) environment is used for architecting a system. An architecture framework is an encapsulation of a minimum set of practices and requirements for artifacts that describe a system's architecture. Models are representations of how objects in a system fit structurally in and behave as part of the system.

General Description

The MBSE environment is developed in MagicDraw, a business process, architecture, software and system modeling tool with teamwork support. The models follow the MagicGrid framework (Figure 72), which includes views regarding requirements, the system behavior, activity diagrams, structure, and parametrics.

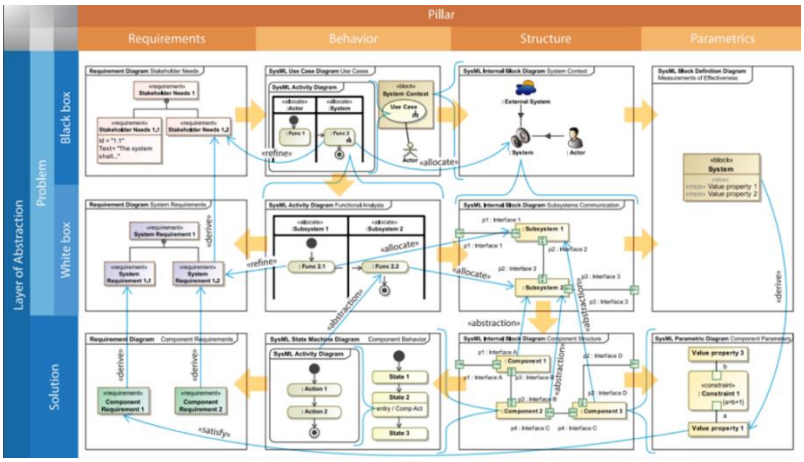


Figure 72. MagicGrid Framework

Navigating to the Project

To open the project, find the tab at the top of MagicDraw that says “collaborate” (Figure 73) and select the option to login. To login, type in your username and password, change the server name to “128.41.183.112:1119”, and the server type to “MagicDraw Teamwork Server” (Figure 74).

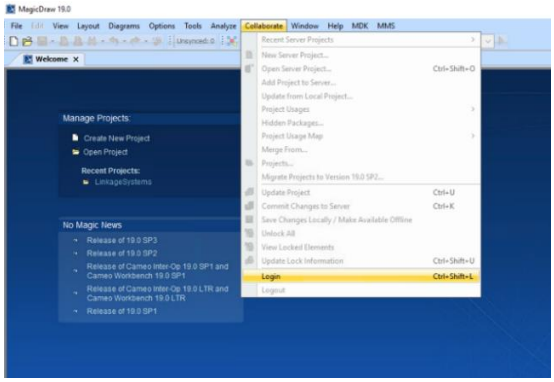


Figure 73. MagicDraw Start-up Window

Figure 74. Logging in to Server

After logging in to the teamwork server, the user can go back to the “collaborate” tab and select the option to “open server project.” The server project where this located is under the “Grand Challenges 2020-21” tab and called “DEAL-RR” under the “RR-2021” branch (Figure 75).

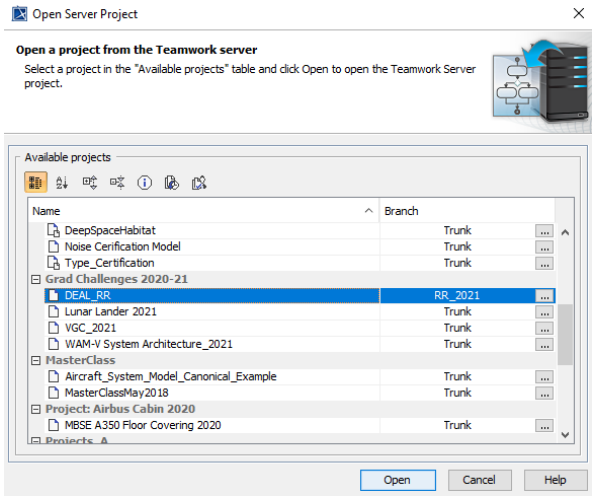


Figure 75. Server Project

After the project is loaded, the user should see two distinct areas – the containment tree, where all of the models are housed, is located on the left and the working space is located on the right (Figure 76).

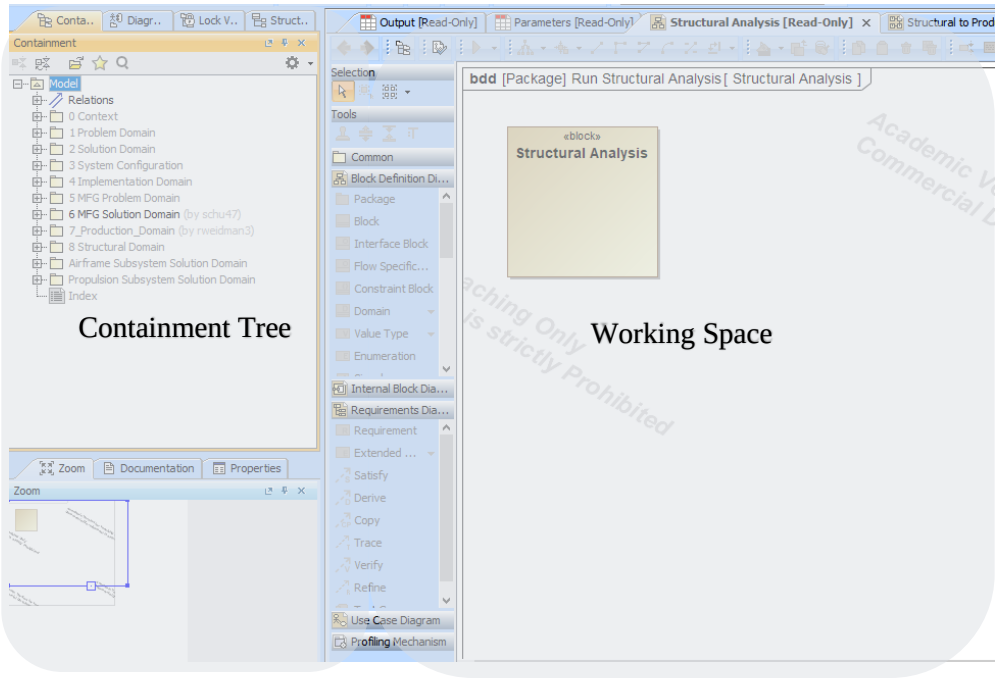


Figure 76. Project Window

The containment tree is organized into 11 high-level packages:

- 0 Context
- 1 Problem Domain
- 2 Solution Domain
- 3 System Configuration
- 4 Implementation Domain
- 5 MFG Problem Domain
- 6 MFG Solution Domain
- 7 Production Domain
- 8 Structural Domain
- Airframe Subsystem Solution Domain
- Propulsion Subsystem Solution Domain

For the purpose of this research, the packages of interest are 0, 5, 6, 7, and 8.

The “Context” package contains information regarding activities, actors, the program structure, and use cases for the design, manufacturing and production domains. The “5 MFG Problem Domain” package contains manufacturing requirements, diagrams for requirements verification, the breakdown structure of the aircraft’s wing and manufacturing process, as well as outputs from manufacturing. The “6 MFG Solution Domain” contains activity diagrams that describe the process of manufacturing a metal and composite wing, and the parametric diagram used to run SEER-MFG (the manufacturing tool). The “7 Production Domain” package contains stakeholder needs, requirements verification and validation, and an opaque behavior that is used to run an

external SimPy script and outputs the factory throughput over a set time, process time, and workstation utilization rate. The “8 Structural Domain” package includes stakeholder needs, requirements verification and validation, instances tables for inputs and outputs of structural analysis, and an opaque behavior to run a MATLAB script. It also includes a parametric diagram that shows the flow of inputs/outputs from the structural model to the manufacturing and production models.

Each package also includes a content diagram, which shows a high-level representation of the project structure. For the purpose of this project, the content diagram is used to organize all of the diagrams in each package into one location. An example of a content diagram for the “8 Structural Domain” package is shown in the figure below.

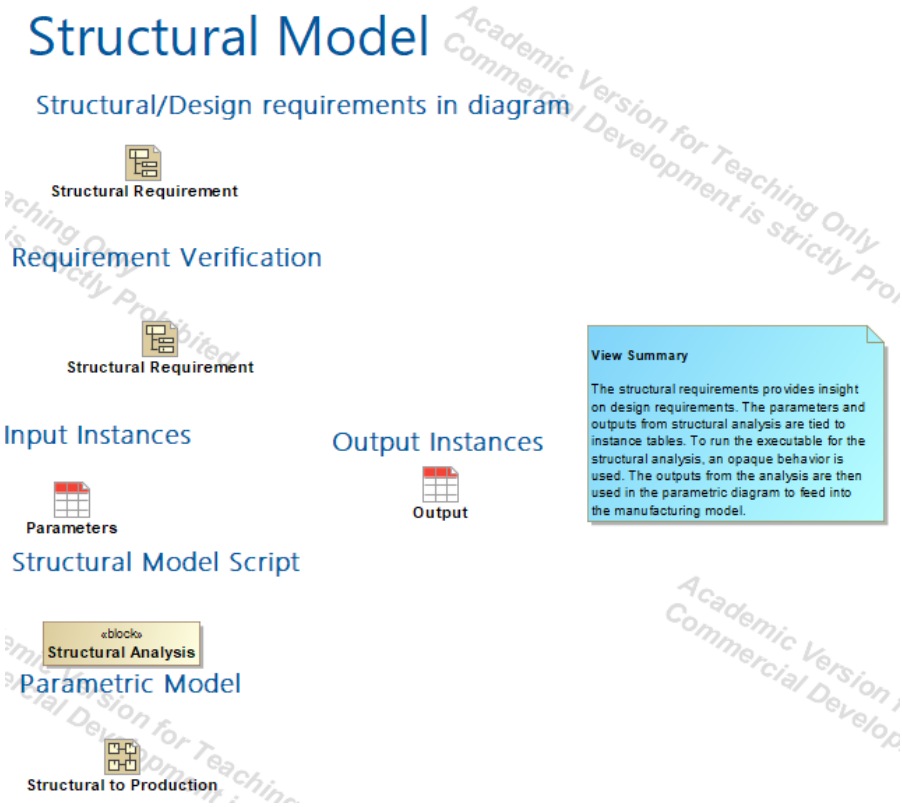


Figure 77. “Structural Domain” Content Diagram

To view a certain diagram of interest, simply double click on the icon from the content diagram or the containment tree. If the diagram says it is “read only,” the diagram needs to be “locked” in order for the user to make any changes. To do so, the user will need to right click on the diagram, and select “lock” (Figure 78). Upon completion, the diagram will now be ready for editing.

*Note: only one person may lock out a diagram at a time

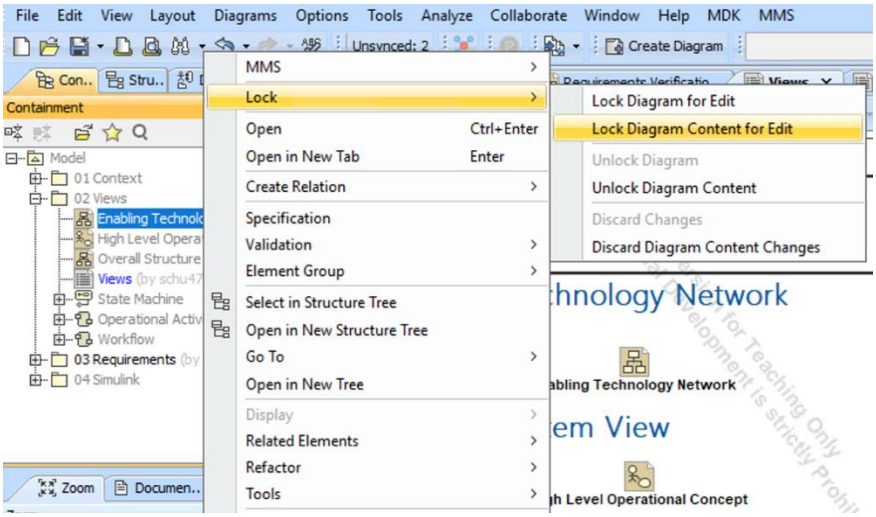


Figure 78. Locking Diagram for Editing

Then, the user can add new elements or create a new diagram by right-clicking on the package in the containment tree and selecting what is to be added.

Use Case and Activity Diagrams

A use case diagram is used to refine stakeholder needs with use case scenarios. Use cases relating to the design, production, and manufacturing model are shown in the figure below.

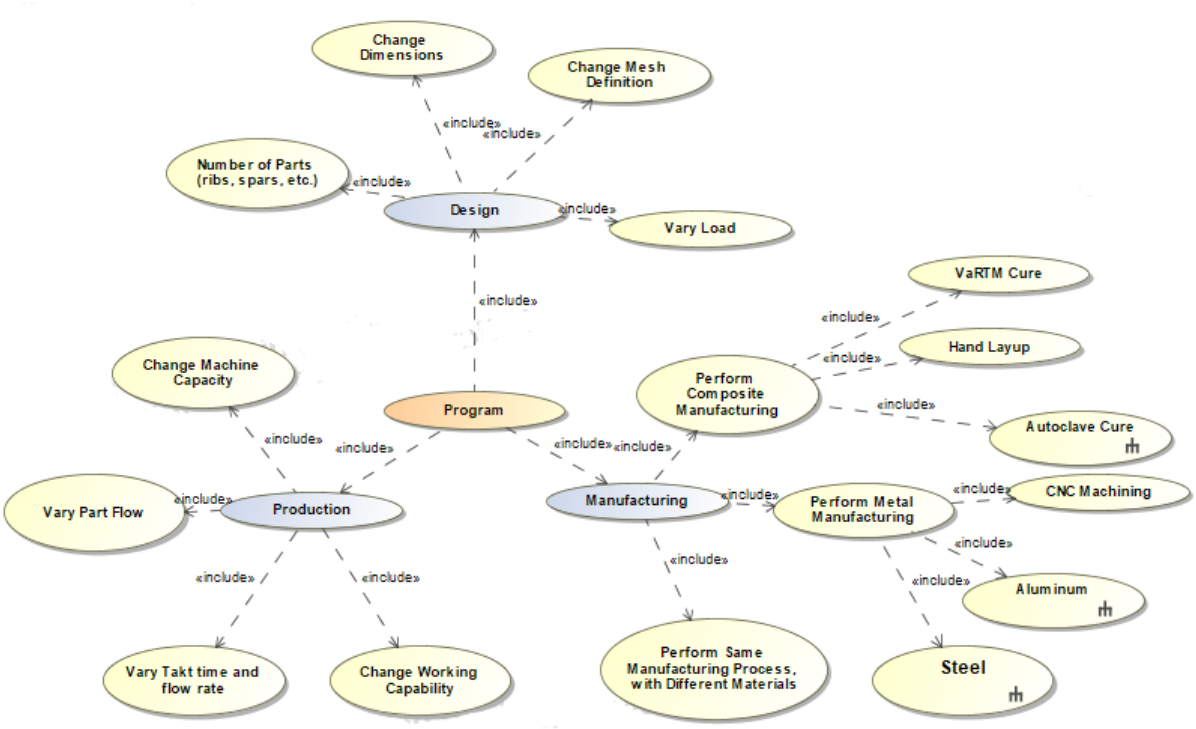


Figure 79. Design, Production, and Manufacturing Use Cases

Use cases can then be further decomposed into activity diagrams by right clicking on the oval and selecting “create diagram”. An activity diagram describes the sequence of “actions” that describe the behavior of a block or structural element. Use cases that have a “rake” symbol on it mean that there is an activity diagram tied to it. In this case, there is an activity diagram for manufacturing an aircraft wing with sheet metal, or aluminum, and manufacturing a composite wing with autoclave cure. An example of an activity diagram is shown below.

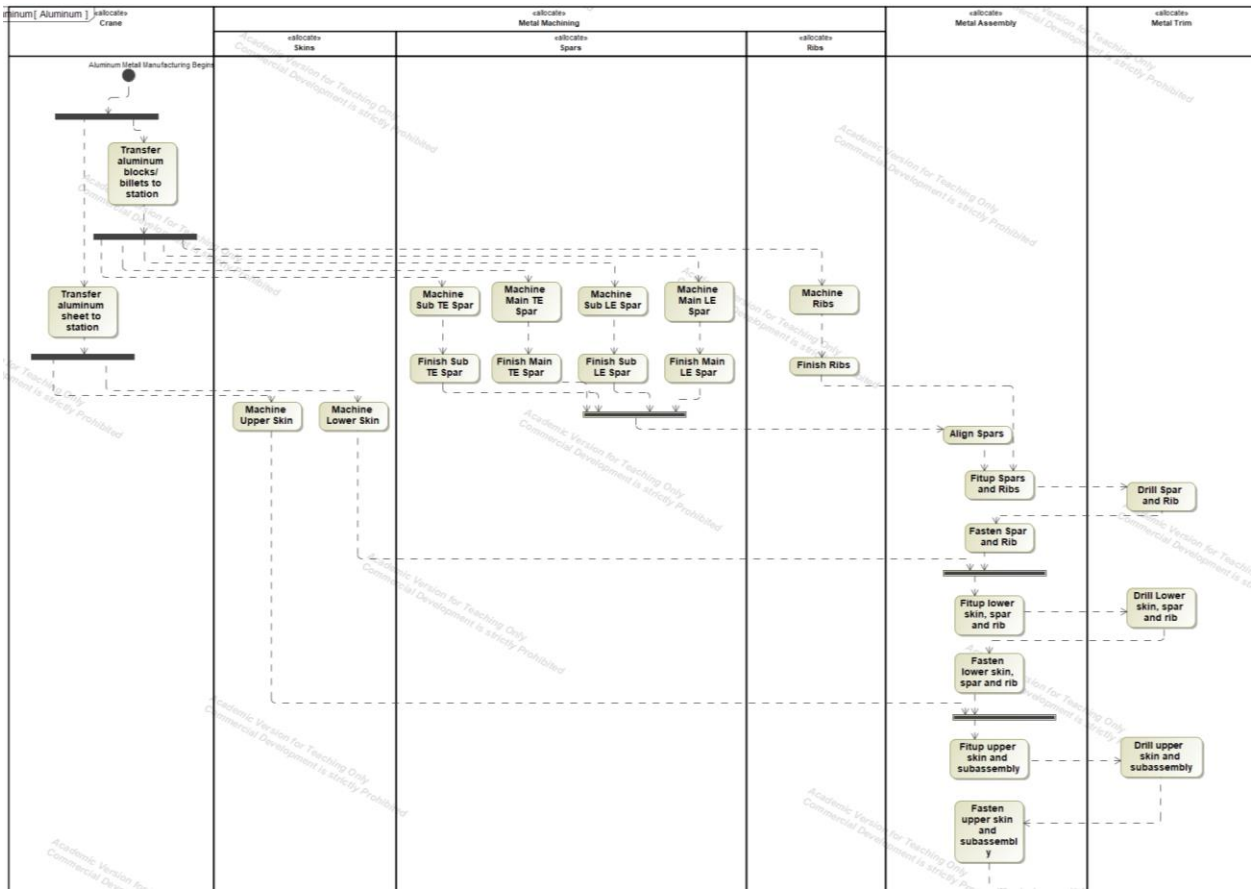


Figure 80. Aluminum Manufacturing Activity Diagram

To add a new “activity,” the user can select the “activity” option in the SysML Activity Diagram tab (Figure 81) just before the containment tree. Then the user can select which swimlane the activity needs to go to, rename it, and attach it to the previous/next activity with a flow.

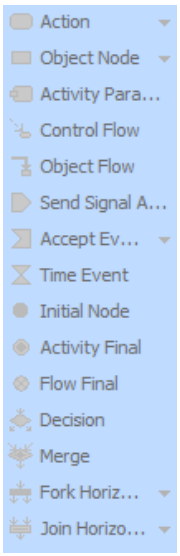


Figure 81. SysML Activity Diagram Elements

In an activity diagram, the two most commonly used flows to connect between activities and signals are control flows (dotted line) and object flows (bold line). Control flows show the flow of control from one action and object flows show the path along which an object or data can pass. When creating an activity diagram, elements that are needed include a “start node” (black circle) which represents the starting point for the activity, a “stop node” (white circle with black circle inside), which is the stopping point for the activity, and “activities” that are allocated to swimlanes (the columns). Additional elements could include send and accept signals to pass signals from one activity to another.

Requirements Verification and Validation

Each package includes a content diagram, requirements table, requirements tree, and diagrams for requirements verification and validation.

Requirements are read from an external Excel file into the requirements table. To make changes or additions to the requirements, the user will need to edit the requirements on the Excel sheet and “read” the new changes. The requirements will auto-populate in the containment tree.

To verify and validate requirements, the user will need to navigate to the “Requirements Verification” diagram. There, the user will see a plethora of different requirements (Figure 82) that are tied to a “block” and a “constraint block.” An example of a constraint diagram and its corresponding parametric diagram are shown in Figure 83 and Figure 84, where the requirement is based on the main spar thickness of the wing.

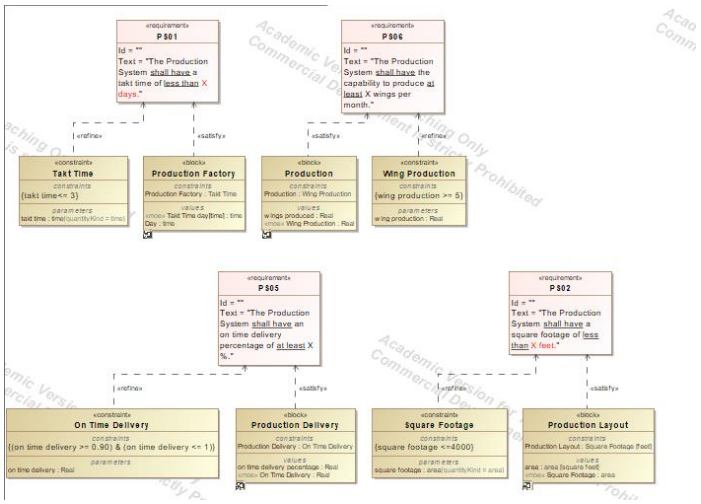


Figure 82. Requirements Verification Diagram

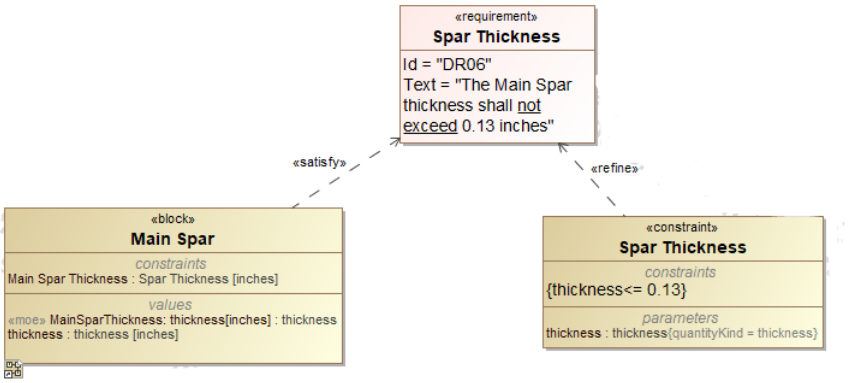


Figure 83. Requirements Constraint Diagram

Additionally, each “block” is tied to a parametric diagram that includes a measure of effectiveness (MOE), the value, and the constraint block that has an input port (green outlined square) tied to it as shown in the Figure below.

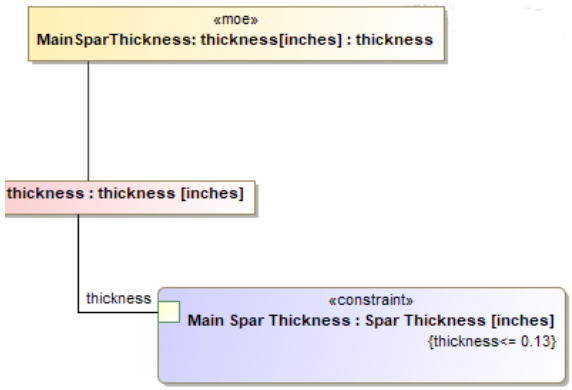


Figure 84. Requirements Parametric Diagram

In this case, the input port is the thickness. This tells the diagram that the parameter that is being passed for verification is the thickness of the main spar. After the input port is identified, the requirement is ready for verification.

To do so, the user needs to select the “block” of the requirement that is to be verified and press the simulation button in the right-hand corner of the working space (Figure 85).

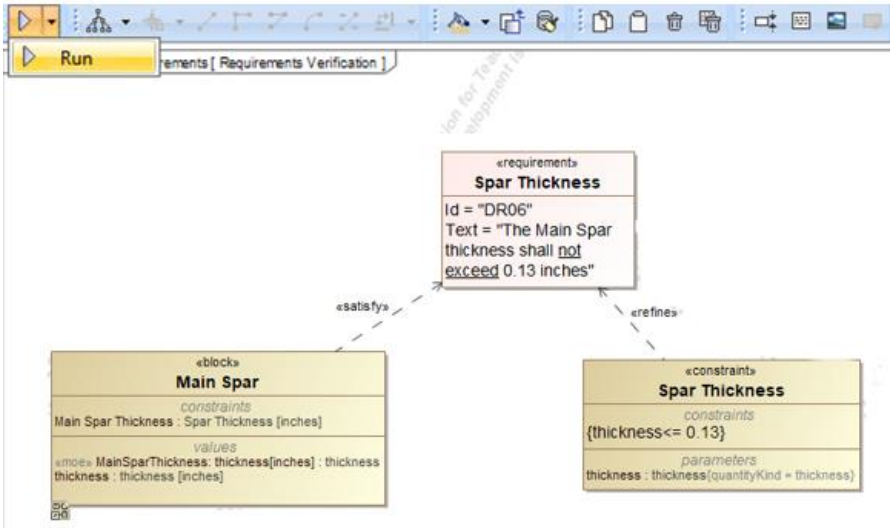


Figure 85. Running a Simulation

Then, a value can be given to the constraint property and this determines whether the requirement is satisfied (highlighted in green, as shown in Figure 86) or not satisfied (highlighted in red as shown in Figure 87).

Figure 86. Requirements Simulation Window- Pass

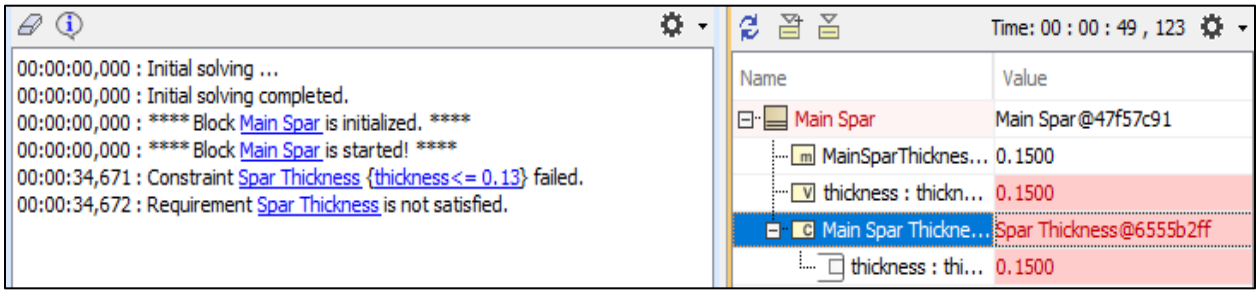


Figure 87. Requirements Simulation Window -Fail

If new requirements that are created need to be verified, the user will need to do the following:

- Drag the new requirement from the containment into the requirements diagram
- Create a “constraint block”
    - Specify the block’s name
    - Specify the constraint using mathematical expressions such as >=, <= or = = and the parameter that is being passed
        - i.e. thickness, takt time, operation time
    - Select the constraint block and create a “refine” relationship to the requirement
- Create a “block”
    - Specify the block’s name
    - Create a constraint property that ties the “block” to the “constraint block”
        - It will be displayed as “block”: “constraint block”
    - Create a value property
        - Assign it a descriptive name that relates to what the parameter will be
            - i.e. the day, the time, etc.
        - Assign the value property a “type”
            - i.e. Real, time, length, watts, etc.
    - Create a measure of effectiveness (moe) property
        - Assign it a name and a “type”
            - Note: The type will be the same as what was assigned for the value property
- Select the “block” and create a parametric diagram and press ok
- Connect the moe to the value property
- Create a port on the constraint property
    - Assign the parameter name to it
    - Assign the type
- Connect the value property to the constraint property

After following these steps, the new requirement will be ready for verification.

Opaque Behaviors

One of the elements that are used in this model is an “opaque behavior,” which is a behavior with implementation-specific semantics. For example, a behavior with Java or C++ code is an opaque behavior.

The table below shows which behavior each model is tied to.

Table 21. Opaque Behavior linkage to Model

Model	Opaque Behavior
Structural	MATLAB
Production	Groovy
Manufacturing	MATLAB

The following figures demonstrate how to set up, attach and run an opaque behavior to run external tools for analysis. The Figure below shows an example of the specification window of the

opaque behavior. Here, the user can set the “language” and “body”, which is the script that is to be run.

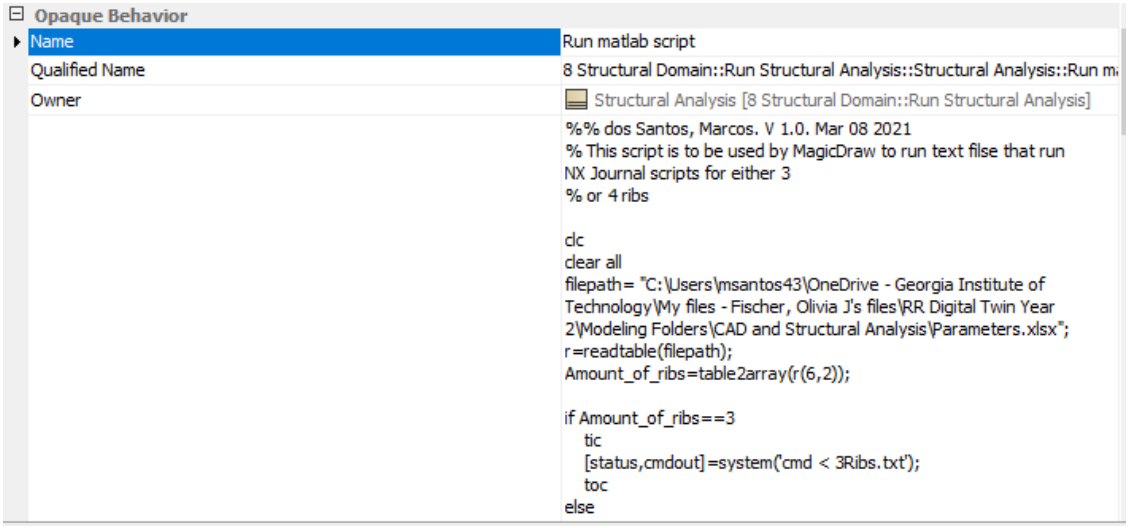


Figure 88. Opaque Behavior Setup

To test the code before assigning the opaque behavior to a block or diagram, the evaluation window is used to execute the code. An example is shown in Figure 89.

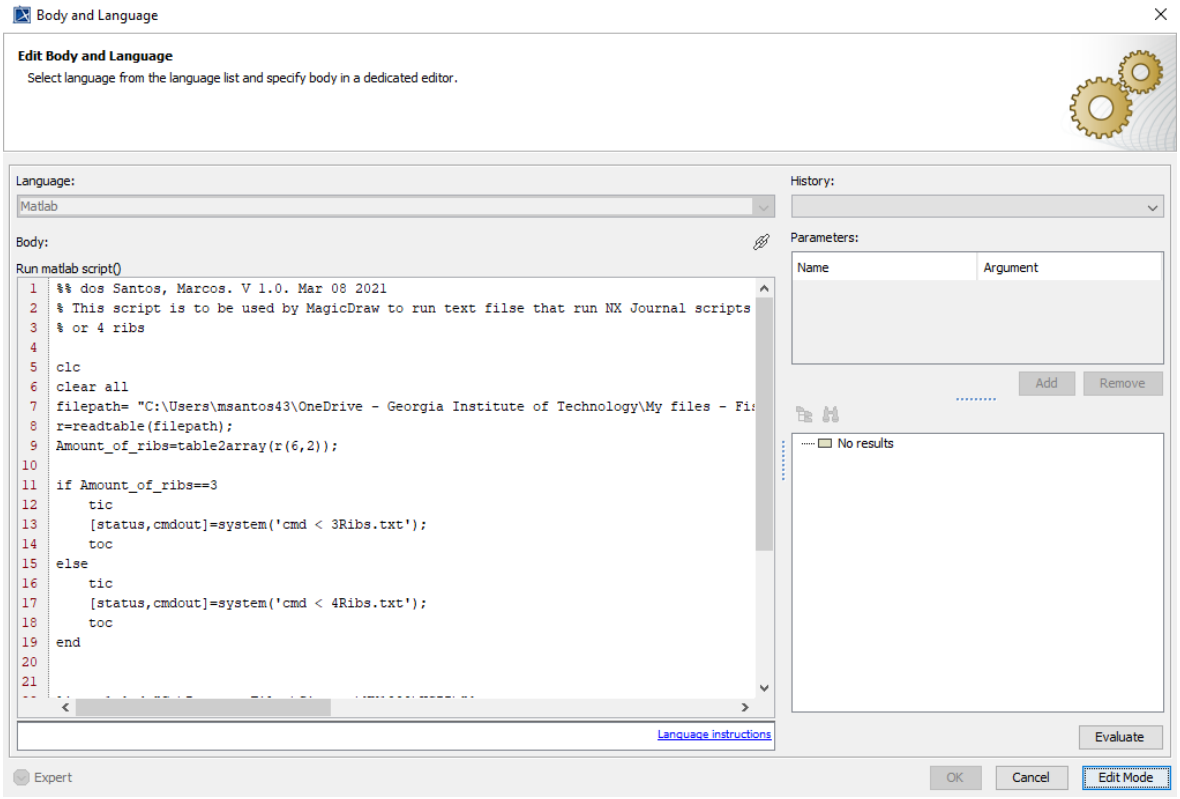


Figure 89. Body and Language Window of an Opaque Behavior

After the opaque behavior is specified, it can be assigned to a block as its “classifier behavior” and “owned behavior.” Now, the block is ready for simulation. The simulation is run and a window (Figure 90) appears that will run the opaque behavior, and therefore the MATLAB script (and structural analysis).

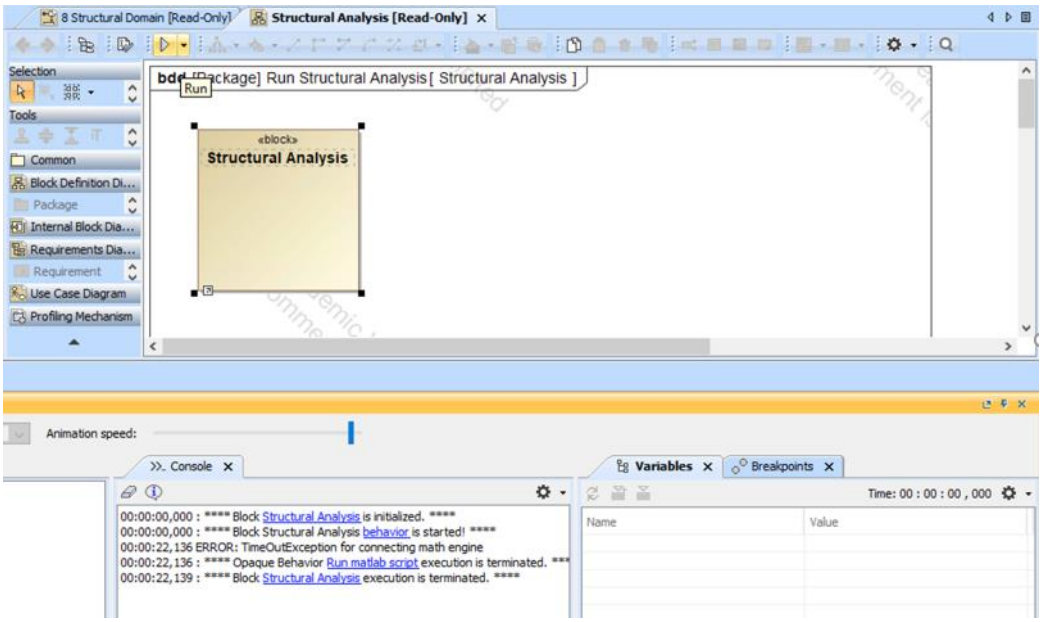


Figure 90. Running an Opaque Behavior

The analysis is complete after the opaque behavior execution is terminated.

Parametric Diagram

A Parametric diagram is a specialization of an Internal Block Diagram (IBD) that enforces mathematical rules (Constraints) defined by “Constraint Blocks” across the internal “Part Value Properties”. In the MBSE environment, the parametric diagram (Figure 91) is used to tie the outputs from structural analysis to the manufacturing and production models.

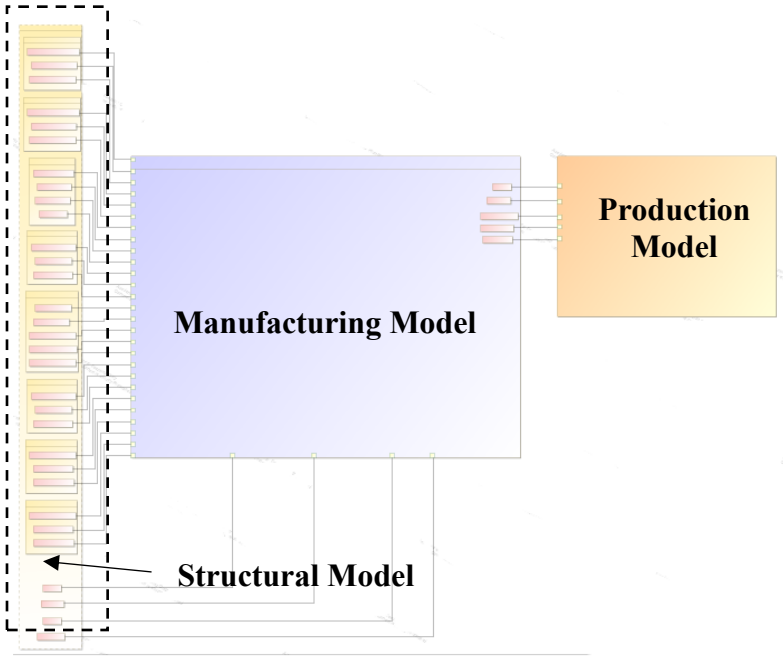


Figure 91. Parametric Diagram

The yellow box represents the structural model, where the pink boxes within the structural model represent the outputs that are generated from structural analysis. These outputs are directly tied to an instance table, under the package “Structural Domain, Black Box, from NX.” The last four value properties in the structural model are the outputs from the manufacturing model. The blue box represents the manufacturing model, and the part values there represent the outputs of the production model, which is in orange.

If there are more outputs that need to be added to the parametric model, the user can create a new element called “value property” and then drag the value of the new instance into it. To check if this is successfully, the user will need to open the specification of the value property (Figure 92) and ensure that the “default value” is set to the value of the instance.

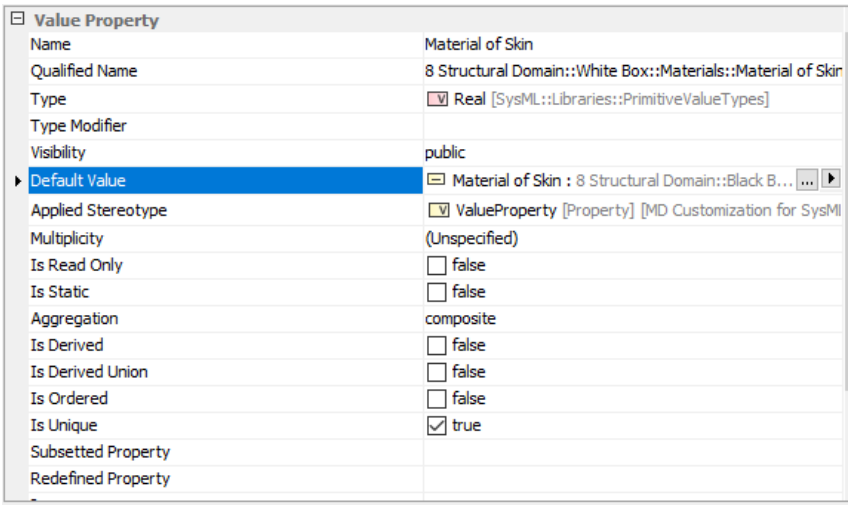


Figure 92. Specification of Value Property

Block Definition Diagrams

A decomposition of the manufacturing process was created using block definition diagrams and are located in the “white box” package of “5 MFG Problem Domain.” There, the user will find a breakdown of an aircraft wing (Figure 93), the manufacturing process (Figure 94), and manufacturing specialization (Figure 95).

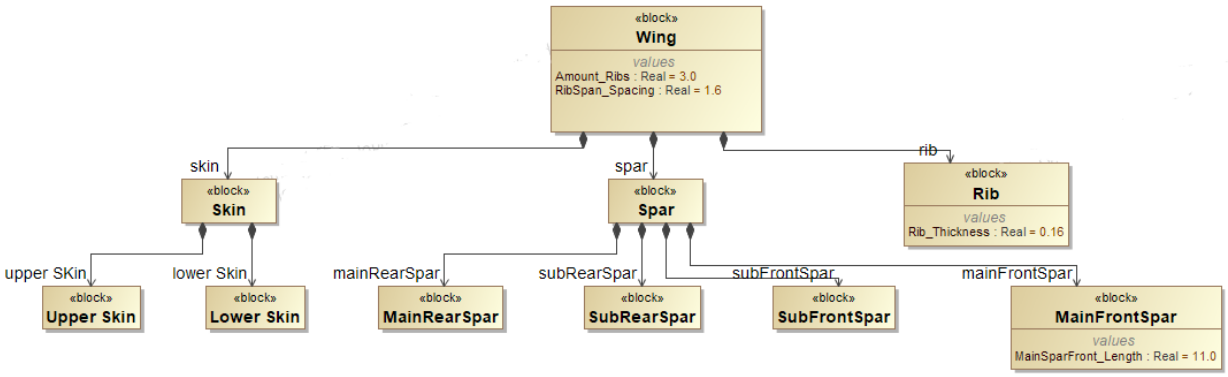


Figure 93. Wing Structure Breakdown

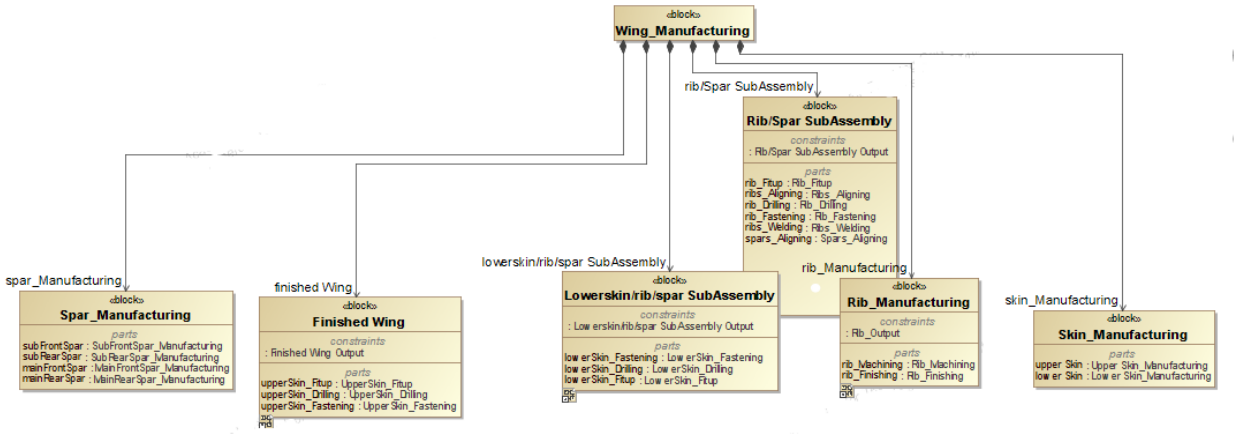


Figure 94. Wing Manufacturing Breakdown

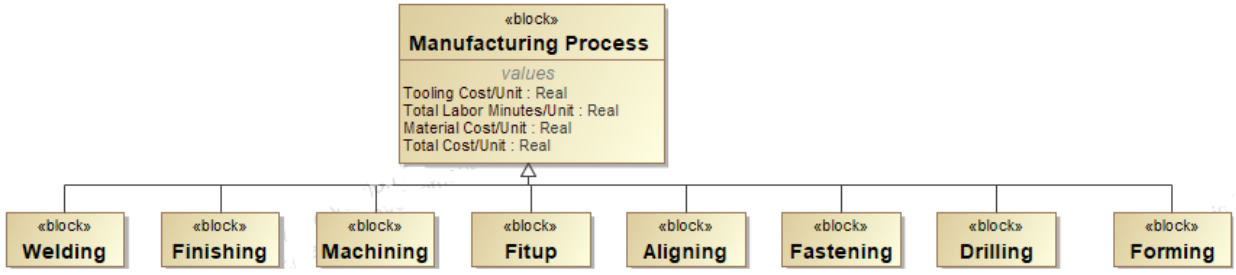


Figure 95. Manufacturing Specialization

Notice that for some of the blocks in Figure 94, a parametric diagram exists at a different level. The user can double click on the block and see one of the following diagrams (Figure 96). The example below is the processes (machining and finishing) used to manufacture the rib.

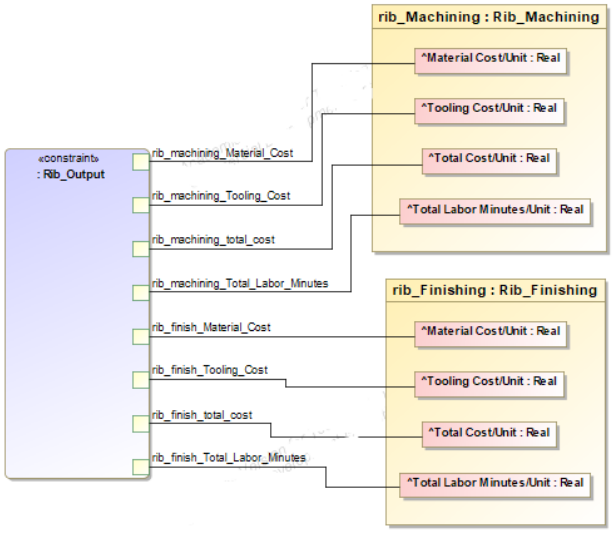


Figure 96. Rib Manufacturing Parametric Diagram

The user can “simulate” or “run” the parametric diagram to see the output values that correspond with rib machining and rib finishing.

Currently, the script that is used to read the files is user specific, so for new users, they will need to edit the path to accurately read the outputs from SEER. TO do so, navigate to the constraint block within the containment tree and expand its properties, like the Figure below.

Figure 97. Rib Output Constraint Property

Then, double click on the double brackets to open the specification of the window (Figure 98).

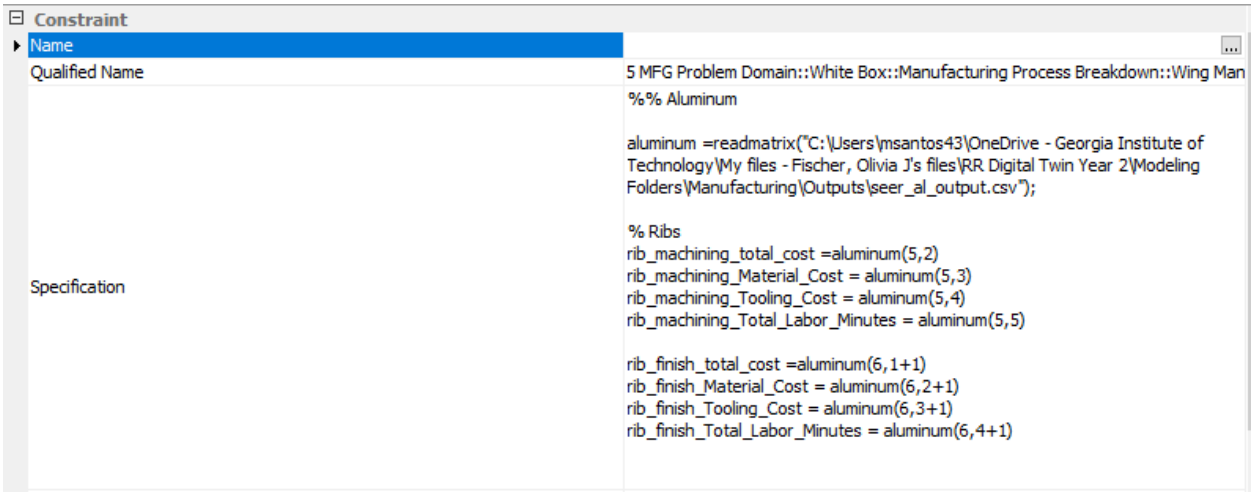


Figure 98. Specification of Rib Constraint

Then, the user will be able to edit the file path to their own to read the correct .CSV output file.

Parametric Structural Analysis

This user guide is designed to provide documentation on how to create and use the environment. In addition to creating the codes in NX, the user will use MATLAB and Excel for completing and automating the capability. This is a relatively simple process if the user already has knowledge of how to use NX for creating Finite Element Models (FEM). However, if the user does not have knowledge on FEM creation using NX but does have in other software for FEM, the process to be followed should be similar. If the user is not familiar with NX, he/she is encouraged to create a simple 2D FEM of a cantilever beam case following a tutorial online (a good representation of the process to be followed: https://youtu.be/5B6n1ZzDF6Q).

The main idea in creating this capability is that similar tools can be used for many design exploration problems. Therefore, although the user manual refers to a systematic structural analysis environment created for an Unmanned Aerial Vehicle (UAV) wing, the user can use the general process to apply to any other relevant design case.

Purpose

Structural analysis allows engineers to investigate what loads and boundary conditions can be applied to the actual part for a given margin of safety before structural failure. Structural failure for metallic materials is usually said to have occurred when the material reaches its yield strength. For this purpose, the Von Mises stress can be used as an indication of the stress level for ductile materials. However, the process described below constructs the FEM of a composite wing. Determining structural failure in this case usually requires stress, strain, and other types of data. To demonstrate the capability of the environment, only stress and strain results for each element of the parts are extracted. The user is certainly encouraged to explore other ways of analyzing the structural behavior by extracting other data. After following the manual, the user should be able to explore additional capabilities.

The requirements of the parametric structural analysis environment are to change the value of certain design variables (e.g., the wingspan), update the mesh, create and apply material and

mesh, apply connection between elements, apply boundary conditions and loads, solve, and extract stress, strain, geometry, and mass from the model. The environment is automated to run through use case using a design of experiments containing 100 cases. Because each analysis can take a significant amount of time (1 to 2 mins), to conduct rapid trades, visualize trends, and rapidly verify that structural requirements have been met, surrogate models of stresses can be built 100 cases. Although not described here, the user can use JMP's Artificial Neural Networks to create a design of experiments and the surrogate models. Ideally, the R2 values of the training and test sets should be close to 1 and should be similar (to decrease overfitting) and the residual error by predicted should center around zero and no clear pattern should be observed.

The sequence of steps below is provided to guide the user in making the user's own environment since, to this date, the environment's ability to work properly is path dependent. The final journal code produced for this process on Siemens NX is used in a loop within MATLAB for changing an input Excel file that modifies the CAD geometry before the FEA process begins. The stress outputs can be collected during the loop and outputted to a CSV file, where subsequent statistical approaches can be used. From this CSV file that has all the maximum Von Mises for the cases ran, Artificial Neural Networks (ANN) can be used in JMP to build surrogate models (i.e., equations) that can be copied and pasted into a MATLAB file for reading the input design variables and rapidly giving out the value of the maximum Von Mises.

Automating the Structural Analysis

Although the process below is targeted for FEA, it does not limit itself to the automation of FEA. Indeed, similar steps, especially the ones prior to the FEA section, can be used for other processes as the geometry is updated and the process follows.

The description below will assume that the user already knows how to conduct FEA for either a single part or an assembly of parts within Siemens NX. Certain codes are needed because some necessary functions, such as geometry update, are not made readily available for the user who wishes to customize and automate their FEA process (these will be referred to as Add-on Codes). Demonstration videos of how the environment works and how to create it should have been provided as well.

NX is user-friendly and adaptable in the sense that it allows the user to write codes in Python, VB.net, and other languages to modify the color of parts, for instance, and automate certain time-consuming tasks. This is done through the journal function, under the developer tab (if it is not visible, the user can make it visible by right clicking on grey ribbon, as shown in Figure 99, and selecting Developer). The team has learned that because the NX journal library in Python is not easily accessible and because most forum questions about NX journaling refer to VB.net programming, rather than Python, VB.net is the preferred language for coding.

Figure 99. Grey Ribbon in Siemens NX

One does not necessarily need to know extensively how to code in VB.net or Microsoft's Visual Basic language in other to automate their processes. The reason is two-fold: many users have gone through similar problems and had their questions answered in the forums and NX allows the user to record certain tasks into a journal. The latter is useful in debugging and making most

of the processes automated, although not all tasks can be added to the journal through the record function. That is when the forums come in handy. The main forum for NX journaling questions is: https://nxjournaling.com/

Process to Automate FEA on Siemens NX

During the steps, to play files with ".vb" extension, refer to the files in the "Add-on Journals" folder under the Main Codes folder. It is important to follow these steps, even if they seem redundant. It was found that the NX journal can fail at times, and making the process redundant can help in ensuring that the process does not fail half-way.

The following steps have worked well for the purposes of designing a small UAV wing. A video of the process has also been made available to make it easier to follow the process ("Environment Creation"). If the user's analysis is parametric, make sure to reference the geometry dimensions to an Excel file with the desired dimensions. This "Master Excel file" will be updated for the Design of Experiments (DoE) run conducted from Matlab.

1. Open the assembly of parts.
    a. Make sure that all constraints are set appropriately when creating the assembly of parts so that upon changing dimensions, the parts remain aligned as the user desires.
2. To change the journal language (if desired) from VB.net to another, go to Menu -> Preferences -> User Interface Preferences -> Tools -> Journal -> Journal Language (languages available are C#, C++, Java, Python, and VB.net).
3. Reference the geometry dimensions to an Excel file with the desired dimensions by adding "ug_excel_read ('path\parameters.xlsx', 'B6')" to each variable dimension, where "path\parameters" is the path to the Excel file and "B6" is the cell whose dimensional value will be used by the CAD.
4. To record a new journal, go to the Developer tab and record a new a journal by pressing Record and entering a journal name.
5. Play "UpdateforExternalChange.vb" located under Add-on Journals in the Main Codes folder. Errors may occur - it is normal. Press Ok and continue the process.
6. Make all components work parts by double clicking the individual parts.
7. Play "UpdateforExternalChange.vb" again for redundancy. Errors may occur - it is normal. Press Ok and continue the process.
8. Double-click on all assembly components again to make sure they are updated after the Excel files are read for updating the geometry values.
9. Play "UpdateforExternalChange.vb" again for redundancy.
10. Run FEA under Pre/Post as would be normally done.
    a. Under the Solution window (as shown below), settings, such as 5000 steps, running on foreground, and using element iterative solver, can be modified. This is done under the General side option.
    b. Since not given by default, make sure to request outputs for strain if that is desired (this is done under the Case Control side option and navigating through the Output Requests).

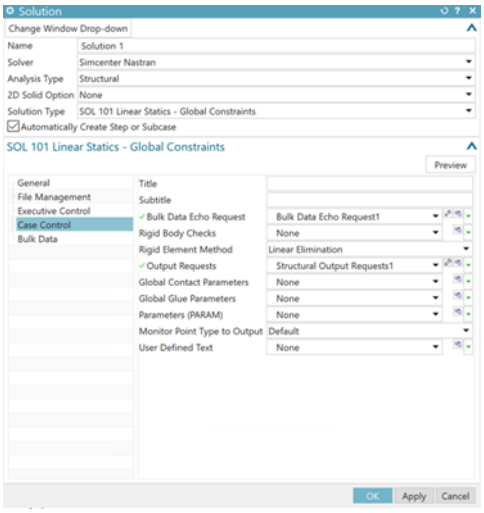


Figure 100. Solution Window

11. Create material (under Manage Materials, if needed), create mesh collectors, assign materials under the collectors, and apply connection between elements
    a. For 2D mesh collectors, to assign variable thicknesses, make sure to create an inter-part expression that references the parameters Excel file. Under Developer, access Expressions, and under Actions press the side icon Create Multiple Interpart Expressions. Choose the part whose expression the thickness needs to reference and subsequently choose the dimension.
        i. Note that alternatively, the user could create the dimension under Expressions as was described in Step 3.
    b. For the connection between elements, 3D elements can use Mesh Mating function in the FEM definitions to ensure the nodes are common between the different meshes. For 2D elements, the user can use 1D elements (RBE2/RBE3) or contact/gluing definitions (this will be in the Simulation definitions.
12. To extract stress/displacement/strain or any other type of data for elements, follow steps 13-18.
13. After the solution is complete, close all the pop-up windows and double-click on Solution 1 and then Results in the Simulation Navigator. This will load the FEA results file.
14. Press Return to Home in the top part of the screen. Go to Post Processing Navigator.
15. Under Menu, Insert, select Selection Recipe.
16. Use the following configurations for all the mesh collectors you wish to extract from:
    c. Strategy: Bounding Volume
    d. Name: type the name of the part to have its FEA data extracted
    e. Under Entity Types, Input Filter should be Mesh
    f. Object: Hover over and click on the part for which a mesh collector was created
    g. Tick Element if the user wishes to output results for the mesh elements
    h. Give a box dimension and position that encompasses the bounds of the part for all simulations
    i. Under Containment, select Inside

17. Under Results, go to Manipulations and open the Results Manipulation Drop-Down and Select Extraction.
18. Under Type, select Selection Recipe from the drop-down menu. Choose the Selection Recipe that the user previously created and the type of data to be extracted. Make sure to add the desired result, as shown below boxed in red. Export the CSV file, specifying the format, file path, and name. Repeat the process for each mesh collector the user desires to extract data from.

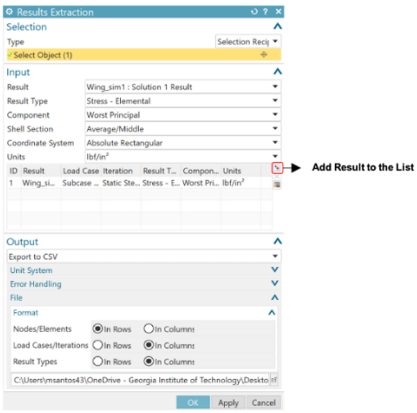


Figure 101. Results Extraction Window

19. Stop the recording of the journal. The code is now ready.
20. To extract geometry information, adapt the code below to fit the user's needs. The code below extracts the number of ribs and the length of the ribs from the CAD model. Insert this part before the beginning of FEA section. On the other hand, if the specific dimension does not need to be extracted from the CAD model, a simple hand-back calculation may be used to calculate the weight of the component.

```
' ========================= Define Expressions to be Probed ============================


        Dim expNameToFind As String = "AmountofRibs" 'Name of dimension as defined in NX
        Dim myExp As Expression
        myExp = workPart.Expressions.FindObject(expNameToFind)

       Dim expNameToFind1 As String = "RibLength"
        Dim myExp1 As Expression
        myExp1 = workPart.Expressions.FindObject(expNameToFind1)

' ============= End of Defining Expressions to be Probed ==============================

' ======================== Writing out Expressions Values to a CSV File ===============

       Dim csvFilePath As String = "Path\SEER_Inputs.csv" 'Path to create or existing
       file
        Dim outFile As IO.StreamWriter =
       My.Computer.FileSystem.OpenTextFileWriter(csvFilePath, False)
        Dim Union As String = "Rib_M_Qty" & "," & 2 * myExp.Value

       Dim Union2 As String = "Rib_M_Rlength" & "," & 2 * myExp1.Value

       outFile.WriteLine(Union)
        outFile.WriteLine(Union2)
```

```
        outFile.WriteLine()

         outFile.Close()
         Console.WriteLine(My.Computer.FileSystem.ReadAllText(csvFilePath))

' ================== End of Writing out Expressions Values to a CSV File ===============
```

Running NX in Batch Mode

To run the journals in batch mode so that NX does not need to be opened for the FEA process to take place and for results to be extracted, follow the steps below to modify the journals.

1. To allow the code to be run from the command line, add the following (the locations are indicated in the video and below).

*Towards the beginning of the code, right after **Module NXJournal,** add the code below*

```
' ===== Add this section for running it on the command line

Dim s As Session = Session.GetSession()
    Dim lw As listingwindow = s.listingwindow
    Dim dp As part
    Dim wp As part

    Sub Main(args() As String)

         lw.open
         OpenPart(args(0))
         Dim theSession As NXOpen.Session = NXOpen.Session.GetSession()
         Dim workPart As NXOpen.Part = theSession.Parts.Work

         Dim displayPart As NXOpen.Part = theSession.Parts.Display

        ------------------------------------------------------------------------------------------------------------------------

        … MAIN CODE

        ------------------------------------------------------------------------------------------------------------------------
```

*Towards the end of the code, between **End Sub** and **End Module,** add the code below:*

```
' ================== Add this Section for running it on the command line =========

        Sub OpenPart(ByVal thePart As String)
         Dim basePart1 As BasePart
         Dim partLoadStatus1 As PartLoadStatus
         basePart1 = s.Parts.OpenBaseDisplay(thePart, partLoadStatus1)

         dp = s.Parts.Display
         wp = s.parts.work

         partLoadStatus1.Dispose()
    End Sub
```

```
Public Function GetUnloadOption(ByVal dummy As String) As Integer
    GetUnloadOption = NXOpen.Session.LibraryUnloadOption.Immediately
End Function
```

To run the code in batch mode (i.e., on the command window, with no interfacing with NX), it is necessary to access NX's command prompt. The goal is to run the created journal on a particular NX file. To do that, follow this process:

1. Open the Windows command prompt, and type and enter *cd "C:\Program Files\Siemens\NX1899\UGII\".* This is where the NX's command prompt is located.
2. Enter *"C:\Program Files\Siemens\NX1899\UGII\nxcommand.bat"*
3. Enter *run_journal "[location of journal file]" -args "[location of NX file on which journal is to be run]"*

If the user's goal is to run the FEA case for many files in batch mode, the team recommends writing a Matlab code (as shown in the *Matlab file for Running FEA in Batch Mode* folder provided) that reads a .txt file with above commands to be sent to the Windows command prompt. In creating several cases, the user must access and modify the values in the Excel file on which the part geometry is referenced.

Running the Models

To run the models, the user needs to first locate the model folders, of which there are two, one for the composites and one for the metallics. There is only one model file for the metallics because a slight change in one of the input files, as will be noted below, allows the user to simulate either the all-aluminum or aluminum-steel wing. The file to run the models is located under the Main Codes>Matlab Codes directory and is named Run_Structural_Analysis.m. To run this file, the user needs to first have Siemens NX and Visual Basic installed and follow the following steps:

1. Locate the files listed in the comments in the Run_Structural_Analysis.m script. Open those files and update all the file paths. Similarly, update all the file paths in the main script itself. There is a tremendous number of file paths to be changed, so they are the most likely source of error in trying to run the models.
2. In all the BETA_3Ribs.vb and BETA_4Ribs.vb, within the first 50 lines, are some commands that refer to the NX-specific user ID. Locate these commands, there should are many per file, change them all. A screenshot of this is shown below for the BETA_3Ribs.vb file:

```
' ------------------------------------------------
'   Menu: Tools->Journal->Play...
' ------------------------------------------------
Dim markId1 As NXOpen.Session.UndoMarkId = Nothing
markId1 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible, "make up to date")

theSession.ListingWindow.Open()

' User Function call - uc4577

Dim part1 As NXOpen.Part = Nothing
part1 = theSession.Parts.Work

workPart.RuleManager.CreateDynamicRule("root:", "msan43DC9CA9hgb1", "Any", "%ug_updateForExternalChange(false)", "")

Dim part2 As NXOpen.Part = Nothing
part2 = theSession.Parts.Work

Dim value1 As Object = Nothing
value1 = workPart.RuleManager.Evaluate("msan43DC9CA9hgb1:")

Dim part3 As NXOpen.Part = Nothing
part3 = theSession.Parts.Work

workPart.RuleManager.DeleteDynamicRule("root:", "msan43DC9CA9hgb1")

theSession.ListingWindow.Close()
```

Figure 102. NX-specific user ID in a BETA_3Ribs.vb file, highlighted in green

3. The .vb files only work for the user with that specific NX user ID, so the user will have to find out what theirs is due to not being able to use the model developer's, Marcos Nascimento, user ID. To find out what the user's ID is, open up NX and record any commands and save the resulting .vb file. Afterwards, open the file, it should contain the user's ID within the first 50 lines, similar to the above screenshot. Replace all user IDs in all the BETA_3Ribs.vb and BETA_4Ribs.vb with the correct ones for the user.
4. To change materials between the all-aluminum and aluminum-steel wings, go to line 1791 in the BETA_3Ribs.vb file and line 1570 in the BETA_4Ribs.vb file. To perform analysis on the all-aluminum wing, uncomment the suppressed line. To perform analysis on the aluminum-steel wing, leave the suppressed line suppressed. A screenshot of the line in question for the BETA_3Ribs.vb file is shown below.

```
1779    Dim markId81 As NXOpen.Session.UndoMarkId = Nothing
1780    markId81 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible, "PSHELL")
1781
1782    theSession.DeleteUndoMark(markId81, Nothing)
1783
1784    Dim markId82 As NXOpen.Session.UndoMarkId = Nothing
1785    markId82 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible, "PSHELL")
1786
1787    Dim markId83 As NXOpen.Session.UndoMarkId = Nothing
1788    markId83 = theSession.SetUndoMark(NXOpen.Session.MarkVisibility.Invisible, Nothing)
1789
1790    propertyTable7.SetMaterialPropertyValue("material", False, physicalMaterial2)
1791    'propertyTable6.SetMaterialPropertyValue("material", False, physicalMaterial2)
1792
1793    propertyTable7.SetTablePropertyWithoutValue("bending material")
1794
1795    propertyTable7.SetTablePropertyWithoutValue("transverse shear material")
1796
1797    propertyTable7.SetTablePropertyWithoutValue("membrane-bending coupling material")
1798
1799    propertyTable7.SetBaseScalarWithDataPropertyValue("element thickness", "MainSpars_Thickness", unit3)
1800
```

Figure 103. Screenshot of BETA_3Ribs.vb file, line that should be suppressed/unsuppressed to change manufacturing process is in green

Also included in the software deliverables are BETA_3Ribs_Demonstration.vb and BETA_4Ribs_Demonstration.vb files to allow the user to see what the files are actually doing.

Manufacturing Modeling

Purpose

Manufacturing modeling is used to estimate important quantities related to the creation of the product such as how long a product will take to manufacture and how much it will cost to do so. This is a significant aspect to capture early on during a product's design because, while the design itself may have exceptional performance, it is possible that its manufacturability is so low as to cause it to be infeasible. Avoiding these types of pitfalls early on thus leads to more informed design decisions. For this work, this involves providing the manufacturing model with nominal geometry and manufacturing data to yield outputs required for the production model to return important results (e.g., time for manufacture and number of wings can be produced given the production constraints) as well as outputs related to the manufacturing quality of the product itself (e.g., cost).

General Description

There are many different types of analyses that can be done for manufacturing modeling. In this work, it will primarily consist of estimating production costs and times for a product given its manufacturing process. Several prominent ways exist to capture this estimation, including statistical modeling, analogous modeling, and generative-analytical modeling. Statistical modeling relies on historic data and uses techniques such as regression analysis to correlate past products' characteristics to its cost so that the cost of new products with similar characteristics can be captured. Analogous modeling estimates cost by comparing a product with a known cost to one with an unknown cost and analyzing their functional, geometric, and cost structure commonalities; if these commonalities exist, then the cost of the unknown-cost product is similar to that of known one. Generative-analytical models use information on "relevant processes of product creation" to determine a product's cost drivers and how much it should cost. It thus requires a large amount of input data from the user. In the aerospace field, statistical modeling is also often known as parametric estimation while generative-analytical modeling is commonly known as bottom-up estimation. The tool to be used for the manufacturing modeling in this work, SEER-MFG, uses a combination of parametric and bottom-up estimation and is leveraged to estimate the cost of a manufacturing and assembly process for a part.

There are three main parts to building a model and running a cost estimate:

1. Define the work elements
2. Choose knowledge bases
3. Enter parameters

Work breakdown structures are composed of work elements, which contain the processes and parameters that the user fills in to compose an estimate. There are thirteen standard work element types. The work elements used in these project files are machining, fabrication, and assembly. Then, knowledge bases are used to define parameter data for a work element at a high level. Finally, parameters are fine-tuned to define the work element (i.e. product, processes, manufacturing description). For instance, the aluminum manufacturing process flow follows the process shown in Figure 80. Subsequently, SEER-MFG will estimate the factors such as cost, labor, materials, and tooling per unit for each process. An example of the outputs for the aluminum process are shown in Figure 104.

Name	Total Cost/Unit	Material Cost/Unit	Tooling Cost/Unit	Total Labor Minutes/Unit
UpperSkinForm	121650.78	4.28	121457.7	113.28
UpperSkinMachining	609.42	0	159.86	269.74
LowerSkinForm	121650.78	4.28	121457.7	113.28
LowerSkinMachining	592.12	0	159.86	259.36
RibMachining	20131.41	1.09	20099	18.79
RibFinishing	546.31	3.02	442.12	60.71
MainFrontSparMachining	20333	1.28	20099	139.63
MainFrontSparFinish	677.21	20.03	442.12	129.03
MainRearSparMachining	20333.91	2	20099	139.75
MainRearSparFinish	677.21	20.03	442.12	129.03
SubFrontSparMachining	20264.64	0.1	20099	99.33
SubFrontSparFinish	673.17	20.01	442.12	126.63
SubRearSparMachining	20264.62	0.11	20099	99.3
SubRearSparFinish	673.1	20.01	442.12	126.59
SparAlignment	816691.31	0	816518.38	79.66
FitupRibs	1360341.05	0.01	1359723	254.09
DrillRibstoSpars	4774269.18	0	4773972	128.95
FastenRibstoSpars	4774558.03	72.04	4773972	211.37
FitupLowerSkin	1360254.39	1.06	1359723	214.76
DrillLowerSkin	4775520.12	0	4773972	623.16
FastenLowerSkin	4775362.56	279.17	4773972	446.69
FitupUpperSkin	1360254.39	1.06	1359723	214.76
DrillUpperSkin	4775520.12	0	4773972	623.16
FastenUpperSkin	4775362.56	279.17	4773972	446.69

Figure 104. SEER-MFG Aluminum Process Outputs

Running SEER-MFG

In this work the SEER models are automatically built, run, and their results extracted using automated Excel tools, which are provided as part of the software deliverables. As long as SEER-MFG is installed and the flex files located in the correct folders building and running a model or even a DOE is as easy as just pressing the “Run DOE” button in the Excel tool.

Flex files are used to obtain the correct outputs from SEER, with the outputs themselves also being stored in the Excel tools’ NewDOEOut tab. To start, place the MFG_OUTPUTS.flx file in *C:\Program Files\SEER\SEER-MFG 8.2\Documents\Export Templates* (if no admin access to the folder is given, copy *C:\Program Files\SEER\SEER-MFG 8.2* to a directory to which the user has access and conduct process placing the necessary files there).

The Flexport file’s name can be changed but must also be changed appropriately in the DOE Inputs tab. The Flexport file itself denotes what types of outputs SEER will provide and can be changed via the *Export & Import > Flexible Output options*. Loading up the current *MFG_OUTPUTS.flx* template from the Flexible Output menu in SEER will better detail some of the defaults that should be followed for which outputs should be selected and how to do so as well. The layout, what all the buttons do, and the meaning of the tabs are all further explained below in the SEER-MFG Model Development section. To simply use the tool, the user must decide the number of cases they want to run, which is changed in the DOE Inputs tab in the “Number of Runs” box. This specifies the cases, which are stored along rows in the DOE tab, that will be fed in to the SEER model, which is generated by reading the commands in the Script tab. After the user has specified the Number of Runs, they can test the model by pressing the “Open SEER and Insert Model (test)” button, which only opens up a test model with the current inputs in the tab. Alternatively, the user can also press the “Run DOE” button, which will run all the cases

they specified. In both situations, the outputs are stored along the columns of the NewDOEOut tab. These results can then be easily exported to .csv files or any other needed file types afterwards.

Running SEER and the Structural Analysis Surrogate Models in MagicDraw

To run SEER in MagicDraw, the user will need to navigate to the “Manufacturing Model Context Diagram” and select the parametric diagrams under the composite and metallic manufacturing heading. There, the user should see a diagram similar to that of Figure 105. For each of the blue constraint blocks, the functions in the Matlab file have been copied and pasted in their definitions. Every time the run button is pressed, output values from the .csv seer output files are read in the console.

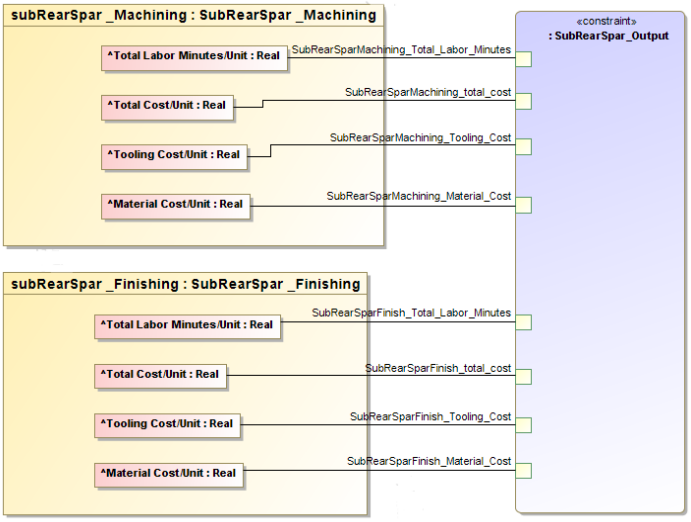


Figure 105. Sub Rear Spar Parametric Diagram

In the case where the user wants to change the MATLAB code that is tied to the constraint block, the user will need to open the “specification” of that constraint and edit the code there.

The user can see the full table of outputs (Figure 106**Figure** is an example for one DoE case) through the packages “5 MFG Problem Domain, White Box, and the Manufacturing Output Instances for each manufacturing process” or from the manufacturing content diagram.

	Name	Total Cost/Unit : Real	Material Cost/Unit : Real	Tooling Cost/Unit : Real	Total Labor Minutes/Unit : Real	Cure Part : Real	Tool Prep : Real	Clean Tools : Real	Set-Up Mins/Unit : Real
1	aileron_Spar_Manufacturing.aileron_Spar_HLU	119355.82	130.78	119120.4		0	1.24	0.08	18.8
2	aileron_Spar_Manufacturing.aileron_Spar_Trim	73.43	0	0	44.06	0	0.52	0.08	16.34
3	aileron_Spar_Manufacturing.aileronSpar_Cure	721.36	457.72	0	158.18	90.33	1.24	0.08	13.34
4	finished Wing.upperSkin_Driling	4778141.68	0	4773972	471.82	0	0.02	0.17	11.84
5	finished Wing.upperSkin_Fastening	4775661.31	299.44	4773972	474.11	0	0.02	0.16	6.48
6	finished Wing.upperSkin_Fitup	1360386.06	1.2	1359723	271.88	0	0.02	0.2	21.41

#	Name	Total Cost/Unit : Real	Material Cost/Unit : Real	Tooling Cost/Unit : Real	Total Labor Minutes/Unit : Real
1	lower Skin_Manufacturing.lowerSkin_Drilling	4776054.2	0	4773972	871.01
2	lower Skin_Manufacturing.lowerSkin_Fastening	4775670.01	180.11	4773972	627.61
3	lower Skin_Manufacturing.lowerSkin_Fitup	1360512.45	0.65	1359723	340.74
4	lower Skin_Manufacturing.lowerSkin_Forming	122447.36	3.08	121457.7	591.95
5	lower Skin_Manufacturing.lowerSkin_Machining	2437.49	0	143.96	1376.12
6	mainFrontSpar_Manufacturing.mainFrontSpar _Finishing	1289.18	20.69	442.12	495.82
7	mainFrontSpar_Manufacturing.mainFrontSpar _Machining	22236.53	0.92	20099	1281.97
8	mainRearSpar_Manufacturing.mainRearSpar _Finishing	1289.18	20.69	495.82	442.12
9	mainRearSpar_Manufacturing.mainRearSpar _Machining	22237.59	1.44	20099	1282.29

Figure 106. Manufacturing Process Output

SEER-MFG Model Development

This section focuses on model development for SEER-MFG, specifically on how to automate the running of a previously existing model.

First, a disclaimer on creating a model or checking the accuracy of a previously existing one. SEER-MFG requires an often-significant amount of background knowledge in a manufacturing process in order to be able to properly model said process. This generally requires the user to have been physically involved and had physical experience with said processes, rely on a subject matter expert who has done the same, or a combination of both by building off of a previously existing model or process. Without this knowledge, it becomes incredibly difficult to tell if the SEER-MFG model being developed is outputting realistic results or not. Even so, considerable care must be taken to make sure that all the inputs are correct and realistic because SEER-MFG itself does not know what is realistic and what isn't. To add on to this, many work elements in SEER-MFG will have an input that has considerable effects on the outputted production time and cost values but whose SEER-default value is completely unrealistic. An untrained eye that does not notice this and does not change it will obtain completely unrealistic results. An example is the default tow width when modeling an automated fiber placement process: the default is 0.5'', which is actually an incredibly low tow width for the AFP process. A more realistic input is around 4'', and so if the user does not notice this and keeps the default value, they will end up with production times and costs 8 times higher than what the true value is. Caution must thus be used and the user must fully explore each work element to identify all such inputs and determine the best values for them.

The remaining of this section and its subsections are focused on the automated running of the SEER-MFG model. It assumes the user has already performed his/her due diligence and ensured that their model is accurate.

The primary way to develop the script to automatically run SEER-MFG is by using one of the many Excel-based DoE tools. The tool was originally developed to allow SEER-MFG to run a very large number of cases as part of a DoE and use those results to generate surrogate models. However, the scripting used to automatically run those cases has also been found to be extremely useful to help run SEER-MFG from the command line as well, though in that case the Excel tool primarily serves as a script developer and debugger; it itself does not actually run SEER-MFG

from the command line. Since the script is important to both of these tasks, its development and debugging will be reviewed first.

Many different versions of the Excel tool exist, though they all function almost identically. The rest of this documentation refers to the version used for this research. The Excel tool is composed of many different tabs, the majority of which are shown Figure 107.

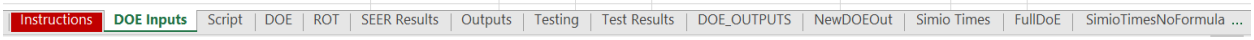


Figure 107. SEER-MFG Excel Tool Tabs

The most important tabs are DOE Inputs, Script, DOE, and NewDOEOut. The Introductions tab has some instructions on how to use the tool, but they are mostly outdated and rely on tabs that are no longer used. The DOE Inputs tab is where the types of inputs are stored as well as the tab used to test the script and run the DOEs. The Script tab is where the script for the SEER-MFG model is stored. The DOE tab is where the actual DOE for all the desired inputs are stored. The NewDOEOut tab is where the outputs of the script testing and the DOE runs are stored. If any of these tabs are missing or misnamed, then the tool is not usable.

The DOE Inputs tab takes the look shown in Figure 108.

	B	C	D	E	F	G	H	I	J
1						type 1 in G5 if you want			
2	**DoE Inputs**	**Current Value**				SEER to keep the labels		**DoE Parameters**	**Value**
3	Model.ParametricWing.USkin_SF_Length	2.25				with the numbers		Number of Design Variables	22
4	Model.ParametricWing.USkin_SF_Width	1.2				**Keep Names?**		Number of Outputs	38
5	Model.ParametricWing.USkin_SF_Thickness	0.004				0		Number of Runs	3
6	Model.ParametricWing.LSkin_SF_Length	2.25						Pause Time	0
7	Model.ParametricWing.LSkin_SF_Width	1.2			**SEER Output File Name**			SEER template file name	test_alum2.flx
8	Model.ParametricWing.LSkin_SF_Thickness	0.004			TestTextFileWorks2.txt			Close SEER how often (in # of runs)?	20
9	Model.ParametricWing.Rib_M_Qty	6			(Make a text file in the folder				
10	Model.ParametricWing.Rib_M_Rlength	1.2			of this excel sheet to use for the			**Open SEER and Insert Model (test)**	
11	Model.ParametricWing.Rib_M_Rwidth	0.1			SEER export)			**Close SEER (Do this after testing)**	
12	Model.ParametricWing.Rib_M_Rthickness	0.013						**Run DoE**	
13	Model.ParametricWing.LESpar_M_Rlength	2.25							
14	Model.ParametricWing.LESpar_M_Rwidth	0.0915							
15	Model.ParametricWing.LESpar_M_Rthickness	0.013							
16	Model.ParametricWing.TESpar_M_Rlength	2.25							
17	Model.ParametricWing.TESpar_M_Rwidth	0.0915							
18	Model.ParametricWing.TESpar_M_Rthickness	0.021							
19	Sub_spar_forward_length	2.25							
20	Sub_spar_forward_width	0.022							
21	Sub_spar_forward_thickness	0.004							
22	Sub_spar_rear_length	2.25							
23	Sub_spar_rear_width	0.021							
24	Sub_spar_rear_thickness	0.005							
25									
26									
27									

Instructions | DOE Inputs | Script | DOE | ROT | SEER Results | Outputs | Testing | Test Results | DOE_OUTPUTS | NewDOEOut | Simio Times | FullDoE | SimioTimesNoFormula ...

Figure 108. SEER-MFG DOE Tool DOE Inputs Tab General Look

The types of desired inputs are stored underneath the column labeled “Doe Inputs.” The values for these inputs for the currently stored case is stored in the adjacent column under “Current Value.” The value in columns F and G do not ever need to be changed and so are ignored. Columns I and J are primarily related to using the DOE tool to run a DOE, however, still bear some importance for regular debugging and script testing runs as well.

The Number of Design Variables currently has little effect on how the tool is being used; however, it is good practice to make sure that the value for it matches the number of inputs under DOE Inputs. The Number of Outputs dictates how many output values will be stored by the DOE Tool in the *NewDOEOut* tab whenever a case is run. Specifically, it allocates a number of cells in the *NewDOEOut* tab equal to the Number of Outputs value. If this number is larger than the actual number of outputs provided by SEER (dependent on the model size and the output types requested in the Flex file), the extra unused cells are filled with zeroes. If the number is smaller

than the actual number of outputs provided by SEER, the extra outputs are not stored. It is therefore prudent to make sure that the value for Number of Outputs is at least equal to the number of outputs actually outputted by SEER. At the least, it should be a large number, though doing so results in extraneous data being stored that can bloat the Excel file size.

The Number of Runs dictates how many DOE cases are parsed and run in the *DOE* tab. The Pause Time should be left at 0 for the DOE cases to be run through as quickly as possible, though it can be non-zero if the user decides to want to briefly take a look at the models as they are being run to make sure there are no glitches. The SEER template file name is the name of the Flex file that will be used to define what outputs are stored in the Excel tool. It must be stored in the *C:\Program Files\SEER\SEER-MFG 8.2\Documents\Export* Templates directory. “Close SEER how often” is used to determine how many consecutive DOE cases are run before the DOE tool closes and reopens SEER-MFG. Higher numbers result in faster overall run time. However, SEER-MFG is an inefficient program and sometimes does not clear its memory completely between cases, leaving behind leftover data that can accumulate and cause run errors if it is not closed often enough. The default value of 20 is adequate for most scenarios.

The *Open SEER* and *Insert Model* button is used to run SEER with whatever script is in the Script tab with the inputs currently stored in the Inputs tab. Doing so also outputs the results specified by the Flex file into the *NewDOEOUT* tab. It is the primary way to test the SEER Model being developed in the script tab. Only one such SEER model can be made at a time. If a change is then made in the Script tab and the model desired to be re-run, the *Close SEER* button needs to be pressed to close the model. The *RunDOE* button is used to run the script for as many cases as specified in the Number of Runs cell and uses the inputs specified in the *DOE* tab.

The Script tab takes the general appearance shown in Figure 109.

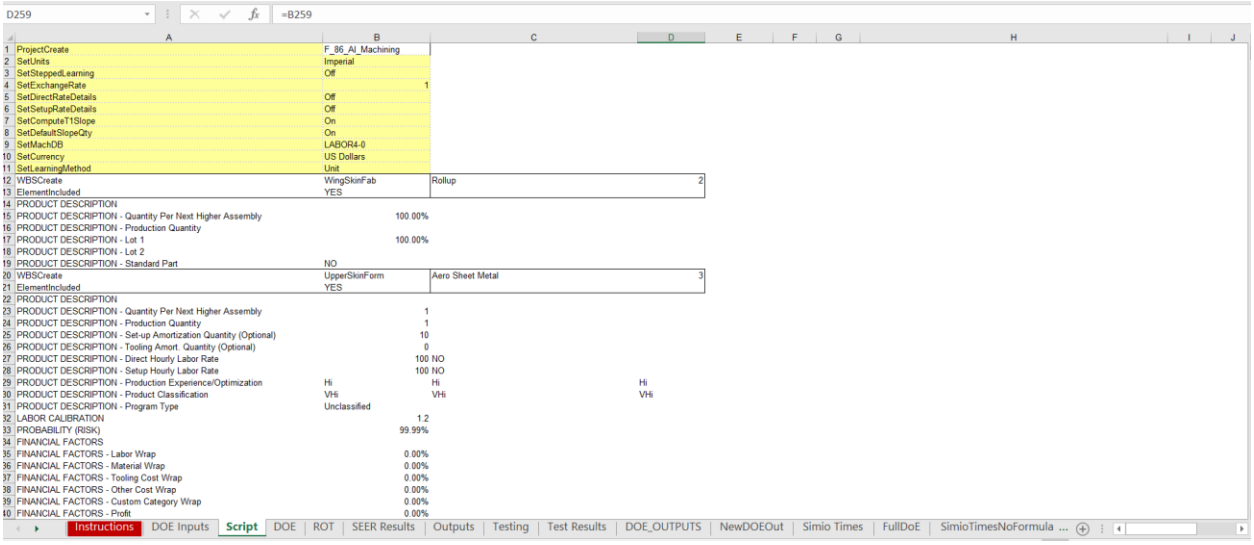


Figure 109. DOE Tool Script Tab General Appearance

Each line is a command used to tell SEER-MFG how to create a model. The first two dozen or so lines define the overall parameters of the model and tend to not change or be extremely similar between different models. They should only be touched if there is a specific need, such as wanting to include learning curves and changing the title of the model. Almost every line afterwards correlates to a line of inputs in each SEER work element. If a new model is being made and

commands for it do not exist yet in the Excel tool, then the command can be obtained by simply going into the SEER model, copying the specific line of inputs desired there, and pasting it into the Excel tool. Care must be taken when using old Excel tools to develop current-year SEER models. This is because the formatting of the inputs and how they are spaced can change with different versions of SEER, causing commands that work perfectly with SEER in one year to not work at all for another. This is sometimes obvious because SEER will simply yield an error when trying to run the script, but at other times can be very dangerous by not throwing an error at all. In such cases, it simply uses default values for all the inputs instead of whatever the user specified. Thus, a crucial part of developing and testing a SEER model is to look through the model in SEER, line by line, and compare it with the script used to generate it in the Excel tool and see if the two are consistent.

The commands in the Script tab are linked to the inputs to be varied in each DOE by linking the formulas for key inputs in the Script tab to variable values in the DOE Inputs tab. This allows the script to be automatically updated when the Run DOE button in the DOE Inputs tab is pressed: pressing the button causes the input values in the DOE Inputs tab to cycle through all the inputs for each case in the DOE tab, and each time this happens, the script's values are updated.

The DOE tab generally looks like what is presented in Figure 110.

A1 | Model.ParametricWing.USkin_SF_Length

	A	B	C	D	E
1	Model.ParametricWing.USkin_SF_Length	Model.ParametricWing.USkin_SF_Width	Model.ParametricWing.USkin_SF_Thickness	Model.ParametricWing.LSkin_SF_Length	Model.ParametricWing.LSkin_SF_Width
2	17.7672156	2.988332134	0.010183847	17.7672156	2.988332134
3	20.9869549	4.886460364	0.008889156	20.9869549	4.886460364
4	13.3	6.17	0.104	13.33	6.17
5	23.72534908	3.846303884	0.007688816	23.72534908	3.846303884
6	24.95097979	4.046485904	0.0108528	24.95097979	4.046485904
7	18.06607437	4.515901595	0.00824391	18.06607437	4.515901595
8	19.25532462	3.924473982	0.011065522	19.25532462	3.924473982
9	19.77134088	3.711718937	0.010522019	19.77134088	3.711718937
10	20.35618964	3.164611904	0.009857962	20.35618964	3.164611904
11	17.4185057	3.384551299	0.00794207	17.4185057	3.384551299
12	21.46344086	3.974895815	0.009089776	21.46344086	3.974895815
13	15.96291799	3.372027016	0.012364185	15.96291799	3.372027016
14	20.75505528	3.217076249	0.012946015	20.75505528	3.217076249
15	22.23753489	3.909010553	0.012000698	22.23753489	3.909010553
16	16.70972073	3.537609286	0.011256393	16.70972073	3.537609286
17	15.93310466	4.478378868	0.010612096	15.93310466	4.478378868
18	20.59728401	4.875046833	0.010059061	20.59728401	4.875046833
19	16.38235772	4.083397344	0.007157189	16.38235772	4.083397344
20	21.02562452	3.705683775	0.009048374	21.02562452	3.705683775
21	16.91986246	4.67714691	0.007858767	16.91986246	4.67714691
22	17.89289498	4.298877188	0.010840468	17.89289498	4.298877188
23	18.6545545	2.755241551	0.007593993	18.6545545	2.755241551
24	19.66198807	4.97577733	0.009264754	19.66198807	4.97577733
25	21.4249238	4.13280288	0.008324981	21.4249238	4.13280288
26	20.53611196	4.366357728	0.009131886	20.53611196	4.366357728
27	23.43310038	4.130466494	0.011401387	23.43310038	4.130466494
28	18.63000744	3.584454355	0.009456563	18.63000744	3.584454355
29	24.46849164	3.614500973	0.010503926	24.46849164	3.614500973
30	20.90744396	4.416406489	0.007848239	20.90744396	4.416406489

Instructions | DOE Inputs | Script | DOE | ROT | SEER Results | Outputs | Testing | Test Results | DOE_OUTPUTS | NewDOEOut | Simio Times | FullDoE | SimioTimesNoFormula ...

Figure 110. DOE Tool DOE Tab General Appearance

The first row is the list of variables names for each respective column. Every row after the first row is a DOE case, with each case having its own input values. When the tool is used to run a DOE, only as many input values are grabbed and sent over to the *DOE Inputs* tab (and subsequently used to update the script) as the value for the Number of Design Variables cell in the *DOE Inputs* tab.

A screenshot of the *NewDOEOUT* tab is not provided because its format is completely dependent on what settings are used in the Flex file. The Flex file allows outputs to be exported as a series of values in a column or in a row. It is up to the user to decide which is more useful. However, of note is that the number of outputs obtained from the DOE tool is equal to the number of rows with

values/names in them in column B of the *NewDOEOut* tab. Thus, if only 40 rows in column B are filled out, then the DOE tool will only yield 40 outputs, regardless of how many were specified in the *DOE Inputs* tab. Additionally, when using the *Run DOE* functionality of the DOE tool, only numeric outputs will be produced in the *NewDOEOut* tab. Any string outputs that are supposed to be produced will instead return a value of 0 since the DOE tool is expecting a numeric output.

With the most important tabs now covered, how to use the DOE tool will now be covered. To use the DOE tool's DOE functionality, the user will first need to determine what inputs will be varied and enter them in the *DOE Inputs* tab, as well as the preliminary values for each. Then, they need to enter the correct values in the *DOE Parameters* box on the right. Afterwards, the user needs to set up the script to generate their model in the SCRIPT tab. The desired SEER parameters to be changed will need to have their formulas in the Script tab be linked to the input values in the *DOE Inputs* tab. The last step, assuming the user has already properly set up their Flex file and determined the correct format of the outputs, is to enter in their DOE in the *DOE* tab and then hit the *Run DOE* button in the *DOE Inputs* tab. Depending on how large the file is, SEER can run through between 120-300 cases per hour, with a possible maximum of about 1000 cases per hour. The outputs are stored in the *NewDOEOut* tab according to how the user configured their Flex file and can afterwards be imported in to JMP or another program to create surrogate models.

The DOE tool's other use is to develop and test the script so that it can be exported to a .txt file and used to run SEER from the command line. This functionality was not used in this iteration and so will not be covered. Debugging models will be focused on instead in case the user wants to develop their own models. Debugging is done by making changes to the script, pressing the *Open SEER* and *Insert Model* button on the *DOE Inputs* tab, observing and analyzing the generated model, pressing the *Close SEER* button, and making the appropriate changes to the script in the Script tab before starting the process over.

One of the primary things to watch out for is SEER throwing an error as the model is being generated. When the *Open SEER* button is pressed, SEER will be opened and the model generated in real time. SEER has a tendency to not throw any explicit errors if the user does not touch the SEER window as the model is being generated, even if there are obvious errors in the script. Therefore, to ensure there are no errors in the script and the model, the user must continuously click on the SEER window that is opened when the *Open SEER* button is pressed until the SEER window is maximized (it will attempt to minimize itself several times) and they can see the model being generated, work element by work element, in real time. Any overt errors will make themselves evident at this point via pop-up window.

The other primary thing to watch out for when debugging is to make sure that all the commands in the script are properly carried over into the actual model. Sometimes, some inputs are improperly spaced out in the script, causing the resulting operation or process in the SEER model to have incorrect values or be missing the process entirely. These types of errors do not cause pop-up windows to appear and so can must be diligently searched for to be found. Some such errors can cause drastic changes in the outputs, making these one of the worst types of errors to have.

Several other things need to be watched out for when debugging. It is typically good practice to have the script be as short as possible but still able to capture all the details needed in the SEER model. SEER will give default values for some process options if they are not explicitly provided. Some operations and process types will have a large number of details that the user does not

actually need to change from the default. It is therefore more efficient and enhances the readability of the script if only the minimum number of script lines is provided to have SEER generate the exact model required. Care must be taken when doing this, however, to make sure that no important lines of commands are left out. Additionally, extreme caution must be made to make sure that the datatype of the input's cells is correct, i.e., a percentage input value does not have a dollar amount, and a measurement or scalar input is not transformed into a percentage. This can cause the outputs to be wildly inaccurate. A prime example is, during development of the model, the datatype for the Number of Tools input was accidentally left as a percentage instead of as a general number. Only 6 tools were needed, but because the datatype was as a percentage, the number in the cell was 600 (600%). SEER expects a number and completely ignores the datatype, causing SEER to mistakenly believe that 600 tools were needed instead of the original 6 and outputting an outrageously high tooling cost as a result.

Another error to watch out for is in regards to exporting the script from the Excel tool to a .txt file. This is typically done by selecting all the rows of inputs and as many columns as have values in them. It is simple to verify that the correct number of rows have been highlighted to be carried over. It is less simple to make sure that the correct number of columns are selected, however, and care must be taken to ensure that all columns with relevant inputs/values are selected, as some lines of commands take up to 20 columns or more to be fully defined. For example, for the aluminum and steel models, columns A through N needed to be transferred over, with columns M and N only being used for a few lines and causing them to be easily missed. Transferring an insufficient number of columns will cause SEER to assume default values for the missing inputs, which can result in undesired outputs. This covers roughly all of the debugging hints and tips that can be obtained without gaining more hands-on experience from the debugging itself.

SEER-MFG is often used to connect the manufacturing discipline with the structural analysis one. This is done mainly through the inputs SEER uses, which are often the outputs of the structural analysis. SEER-MFG, being a bottom-up cost-estimating tool, requires part dimensions in order to obtain its estimates. It so happens that one of the primary outputs of structural analysis and optimization are also a part's dimensions. This is, therefore, how the two are linked: structural optimization is done to converge on a part size to satisfy the design requirements, and these part sizes are used as inputs into SEER-MFG to obtain cost and time estimates. SEER-MFG itself plays no part in the structural optimization and does not care what is done in that step so long as the requisite part dimensions are provided.

The types of inputs that SEER-MFG most commonly require are a part's length, width, and thickness. For parts that have multiple instances within an assembly, such as ribs in a wing assembly, said number of instances per assembly is also required. Thus, for the metallic models used for this research, these input types were required for the ribs, upper and lower skins, and main and sub forward and rear spars.

For a composite manufacturing model, all the input types remain the same except for the thickness. For composites, the thickness is dependent on the number of plies used for each part, given that each ply also has its own thickness. Composites are therefore more complicated in that the singular thickness value is split up into multiple thickness values dependent on the number of plies used and their orientations. Historically, in ASDL, three such values are needed for each part: number of plies in the 0 direction, number of plies in the $\pm$45 direction (total, so 1 +45 and 1 -45 is 2 plies total), and number of plies in the 90 direction. Obtaining these ply values is what makes composites modeling significantly more difficult than metallic modeling.

With the inputs now covered, some attention is given to the outputs. SEER is able to export a very large variety of them and can do so for the individual work elements or for the entire model/project. Typically, the most useful ones are total cost, material cost, labor cost, tooling cost, and labor time. Total cost is typically the sum of the labor, material, and tooling costs, but sometimes does not exactly follow that formula depending on how tooling costs are accounted for. Labor time is the total amount of man-hours (or man-minutes) needed for whatever activity is being calculated for. It makes no assumptions on how many workers are involved. It is essentially the production time for the activity it is calculated for given that only 1 worker is performing that activity. Great care, however, must be exercised in distinguishing between production time and production rate. Production time is the amount of time it takes to produce something and is equal to the labor time. Production rate (throughput) is how many products can be made/activities done in a set amount of time given a specific number of resources. It needs more information than just the labor time to calculate. These resources range from number of workers to number of machines allocated to a specific activity to number of instances of said activity to the amount of space available. Significantly more effort is required to estimate production rate as opposed to production time, with such problems tending to take the form of line balancing problems and factory layout problems and production scheduling problems. Calculating throughput is, therefore, not a trivial task and often requires many assumptions.

SEER-MFG Flex Files

Flex files are important to consider because they determine how the outputs are obtained. They can be accessed by going to the “Export and Import” tab in SEER-MFG and clicking on the “Flexible Export” button. A menu will pop up with three tabs: Load/Save Template, Options, and Template. The Load/Save Template tab is shown below in Figure 111.

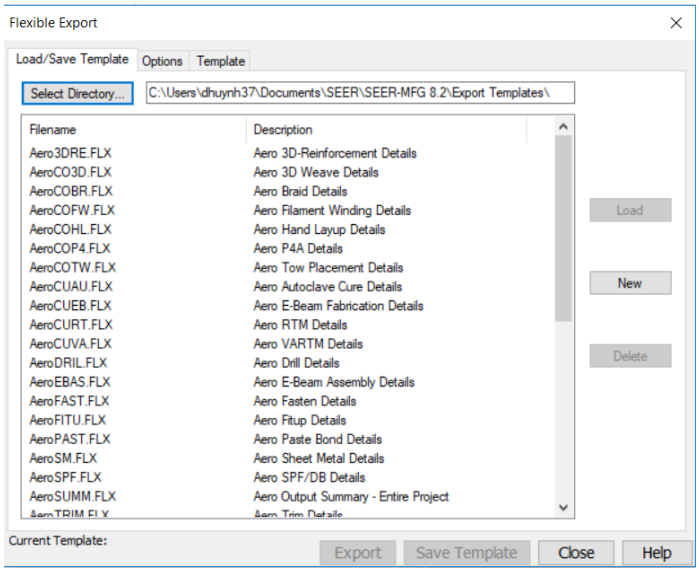


Figure 111. Flexport Load/Save Template tab

As the name suggests, this tab is used to load in or save new flexport templates. Loading one in is simple: simply highlight over the name of the desired template and hit the “Load” button. The “Export” button will export all the outputs specified in the Template tab according to the options listed out in the Options tab. This can be to the clipboard, to a file, or to a report. The “Save Template” button saves the currently entered inputs in the Options and Template tabs to a new

flexport file, which the user will then name and provide a description for. This file is located in the Export Templates folder, which itself is located in whichever folder SEER-MFG was installed in by the user. Similarly, the user can insert and use flexport templates they did not make by putting its template file in the Export Templates folder.

The Options tab is presented in Figure 112.

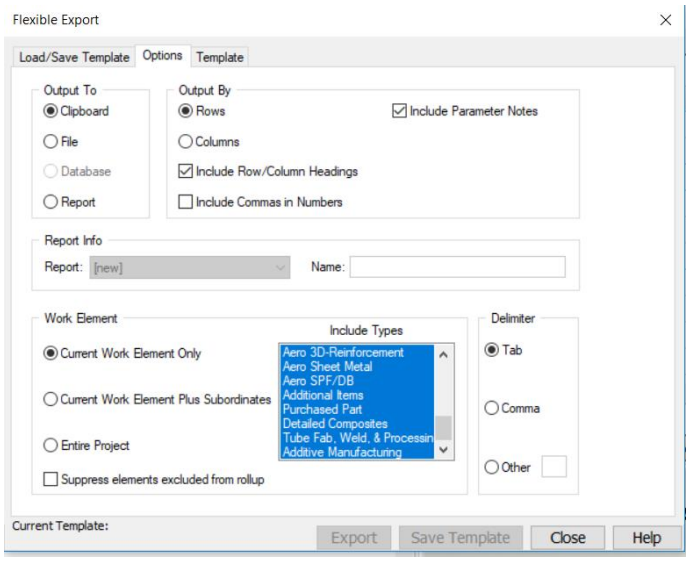


Figure 112. Flexport Options tab

The “Output To” options have already been discussed. The user can also output by rows or columns. For the DOE Excel tool, the user must output by rows. The Parameter Notes option does not produce any notable effects, so it is irrelevant whether or not it is checked. The “Include Row/Column Headings” option allows the user to see what the outputs they are seeing correspond to. As an example, SEER-MFG typically just outputs a long list of numbers with little context. Checking the “Include Row/Column Headings” option allows the user to see which numbers correspond with material cost, tooling cost, total cost, etc. It is a useful option when initially developing a flexport file and seeing what it produces to format it and the *NewDOEOut* tab. However, that option cannot be checked when actually using the Run DOE functionality of the DOE tool because it produces two columns of outputs whereas the DOE tool is only expecting one column.

Below the “Output By” box are the equally important “Work Element” options. They indicate what parts of the model will actually have outputs produced. In most cases, it is desirable to have “Entire Project” selected so that the entire model’s outputs are exported. However, as of SEER version 8.3, this option is bugged and will output nonsensical values. Instead, the user will have to use the “Current Work Element Plus Subordinates” option and select the very first work element in the SEER model before they export any outputs. This can be done automatically when carrying out DOE runs by leaving the command *SelectWBSNumber 1* (the “1” is in a separate column for Excel, one tab over in a text file) at the very end of the script used to generate the model. In the “Include Type” box, it is generally good practice to have everything selected except the “Project” and “Rollup” items, since those two can be obtained by summing up the rest of the model. The Delimiter type is typically set to Tab, though Comma should produce no issues and is easier to

read if running DOEs through external means (Python, Matlab, etc.) instead of the DOE Excel tool.

The Template tab takes the general format shown in Figure 113.

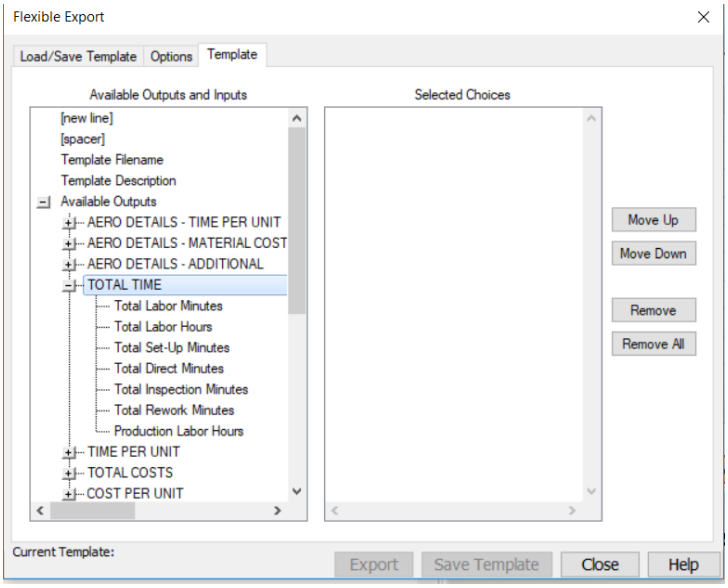


Figure 113. Flexport Template tab

The “Move Up” and “Move Down” buttons are used to order the outputs. Each work element gets a complete set of all the inputs and outputs in the “Selected Choices” box unless otherwise designated in the Options tab. Useful outputs are generally under the Time Per Unit and Cost per Unit lists. Additionally, details for specific operations in a work element (such as the cleaning time, cure time, setup time, ply cutting time, etc.) are generally found under the Aero Details lists. The last notable option is the “Name” output underneath the Work Element Information list. “Name” yields the name of the work element whose outputs are being exported and allows the user to see which elements the outputs correspond to.

Production Modeling in SimPy

Purpose

The production model determines the order of the manufacturing steps, queuing, storage, supply, and assemblies. Because of this, the production model depends on the selected manufacturing model and processes. There are several data sources that are required to create the production model – the CAD Model, the information from SEER-MFG, structural analysis, and facility characteristics.

Accessing Production Files

The production model requires three files to run: the production model Python script titled *deal_production.py,* an input Excel file (of any name), and an input file from the SEER-MFG manufacturing outputs (of any name). NOTE: the input SEER file (or files) *must* be in its own separate folder from any other files. The production model is capable of running any number of SEER input files for one input Excel file. To run additional SEER input files, simply populate the SEER input folder with uniquely named SEER input files.

The structure of these files should be identical to those found in the Github for the production model; in the Github, the input Excel file is labeled as *input_file.xlsx*, and the SEER file is labeled as *test.xlsx*. The production model Github can be found at the following link: rweidman3/DEAL_Production: DEAL Production Model Python Script (gatech.edu).

Running Production Analysis from IDE

The production model can be run from any IDE that is capable of running Python. Running the production model is done by performing the following steps:

1. Open an IDE capable of running Python.
2. Open the *deal_production.py* script.
3. Navigate to the "main" function of the script. This is found at the bottom of the script.
4. As shown in Figure 114, edit *input_file* variable to file path and name of your input Excel file.
5. Edit the *seer_files* variable to the file path of your input SEER files. At the end of the file path, you *must* include an additional backslash and asterisk in order for the code to properly execute.
6. Save the file.
7. Run the file.

The outputs of the production model will be saved in the file path of the *deal_production.py* script to a folder labeled *outputs*. An output Excel file will be created for each input SEER file and given the same name as the input SEER file.

```
2594  # Create a main function, so the program can be executed
2595  if __name__ == '__main__':
2596      # Let's try running it!
2597
2598      # First define your input file
2599      input_file = r"C:\Users\rweidman3\Documents\Python Scripts\v_gc\input_file.xlsx"
2600      # Create the list of all of our SEER files
2601      seer_files = glob.glob(r"C:\Users\rweidman3\Documents\Python Scripts\v_gc\seer_file\*")
2602
2603      # Create an iteration counter
2604      i = 0
2605      # Now iterate through the SEER files and run DEAL_Production for each
2606      for file in seer_files:
2607          # run the automated deal_production script
2608          automated_deal_production(input_file, file)
2609          # Print how many iterations you've run
2610          print(f"Scenarios processed: {i}")
2611          # iterate
2612          i += 1
2613
2614      # now that we're done, print that we've processed all of the SEER files
2615      print("Ground Control to Major Tom...all SEER files successfully processed!")
2616
```

Figure 114. Main function of the production model. The input_file and seer_files variables must be changed to the users system in order for the production model to run.

Running Production Analysis in MagicDraw

To run SimPy in MagicDraw, the user will need to navigate to the "Production Domain" Context Diagram and select the "Production Model Script" block diagram. From there, the user should see the diagram in Figure 115.

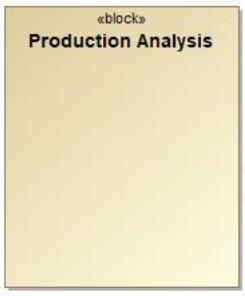


Figure 115. Production Analysis Block

The block is specified by an opaque behavior (as described in Section Opaque Behaviors) with the language “Groovy.” Every time the run button is pressed, the input variable values from design and manufacturing are fed to the Groovy Runtime code. The Groovy script that is used to execute command line programs is as follows:

Runtime.getRuntime().exec "path\\to\\program.exe path\\to\\scripts.py"

Each run in MagicDraw takes only a few seconds to complete. The user will know that production analysis is complete when the opaque behavior execution is terminated. At this point, the output .csv file will reflect the results from production analysis and figures are created into a specified folder on the user’s computer. Examples of the results are shown in the Figures below.

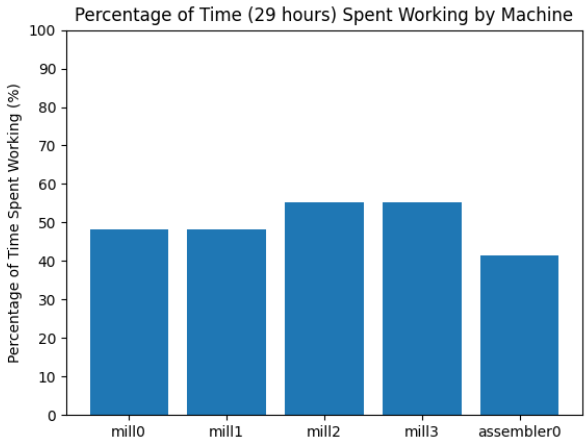


Figure 116. Example machine utilization figure output by the production model.

	wings_made_mean	wings_made_std	mill_rib0_percent_work_mean	mill_rib0_percent_work_std	mill_spar0_percent_work_mean	mill_spar0_percent_work_std
0	14	0	83.02770422	0.03914001	99.90182648	0

Figure 117. Example printout from a production model simulation.

Production Modeling in Simio

Design & Manufacturing Scenario

Four design and manufacturing scenarios were chosen to be modelled in Simio. The scenarios consist of a carbon fiber composite material and metallics being utilized. The first composite scenario uses a hand lay-up and autoclave cure fabrication technique. The fabrication techniques for the second composite scenario include a hand lay-up and VaRTM cure technique. The autoclave scenario requires a low level of integration and joins the parts through fasteners. Since the VaRTM scenario requires a higher level of integration, the assembly utilizes both co-bonding/paste bonding and fasteners for its joining technique. Industry practices provided structural data while the SEER model outputs processing time data for the Simio models.

Building the Simio Models

Facility

Initial assumptions made before modelling include a Poisson arrival process for source entities (interarrival times are exponentially distributed and independent of one another), independent service times, queue capacity is at 50, and queue discipline follows a first-in first-out (FIFO) approach. Starting from the highest-level process flow chart, subassemblies and shared process steps were identified (ex. DEAL – VARTM Process Model).

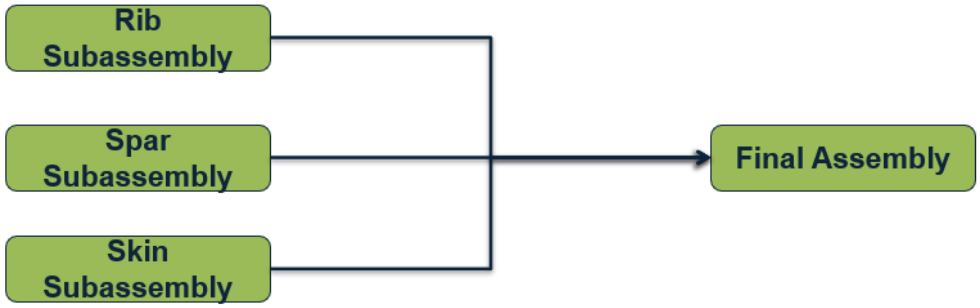


Figure 118. Highest-level process flow chart from the Autoclave and VARTM Process Models.

In order to recreate the subassemblies and final assembly in the model, the “Drawing” tab is selected under the “Facility Tools” ribbon. Selecting “Floor Label” allows for the placement of a rectangle in the model window, shown in **Figure *119*** Fig. 1.2.1.

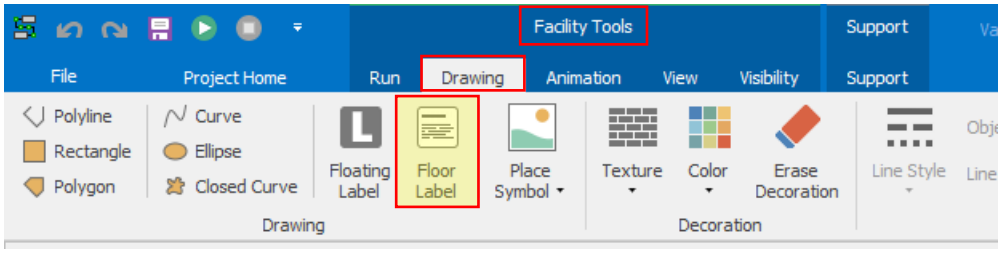


Figure 119. How to create a floor label, with relevant tools boxed in red, and the final “Floor Label” shaded in yellow.

Using the floor labels, the subassemblies and final assembly were physically mapped to areas in the model window. The background colors were adjusted to allow for unique color associations

for each area mapped. The text for each subassembly was adjusted to display the titles of the subassemblies by adjusting the text between label tags. This is shown in Figure 120.

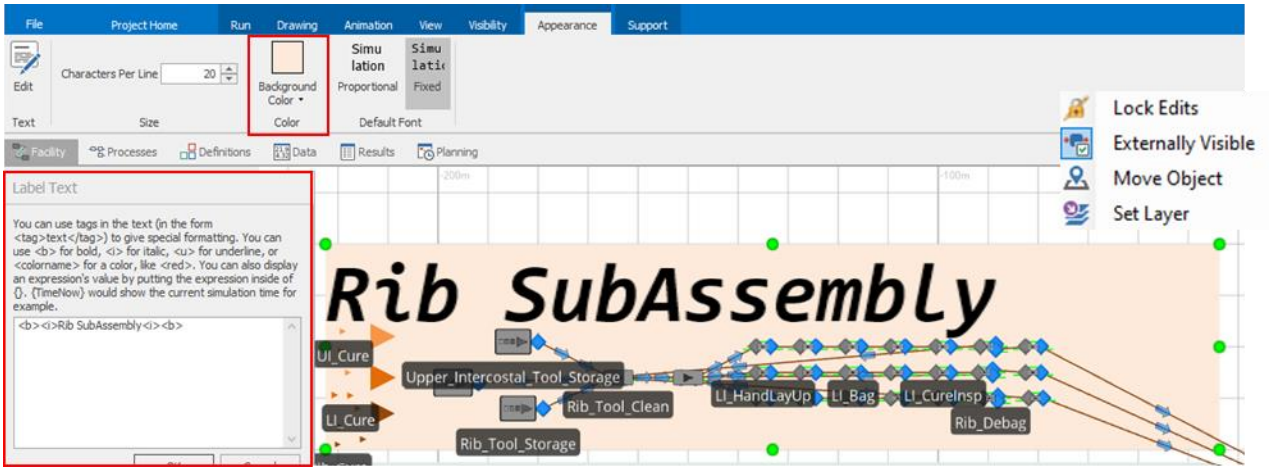


Figure 120. How to change floor label color and title floor labels, and lock edits.

All floor labels edits were locked after placement to keep the labels from being moved. To lock the edits, right click on the floor label and toggle “Lock Edits”, shown in Figure 120. The tile will turn blue when locked. The result of the area mapping is shown below:

Composite Subassemblies: Description

Each composite model has a total of four subassemblies, the Rib, Spar, and Skin, and Final Assembly. The Rib subassembly consists of one line with the assumption the shear ties and joints are also being made as the rib is made in the production process. The Spar subassembly consists of an Aileron Spar line, Main Leading and Trailing edge lines and Sub Leading and Trailing edge lines, making a total of 5 lines. The Skin subassembly consists of an Upper Skin line and a Lower Skin line. The final assembly for the Autoclave process requires a fit-up, drill, and a fasten between each of the sub-assemblies, starting with the final ribs. For the skin sub-assembly, however, the upper and lower skin are attached separately to create the final wing. For the VaRTM model, the same applies, however the steps per attachment differs, instead of a drill and fasten after the fit-up, the components are paste-bonded together. All these descriptions are shown below in Figure 121, Figure 122, Figure 123 and Figure 124.

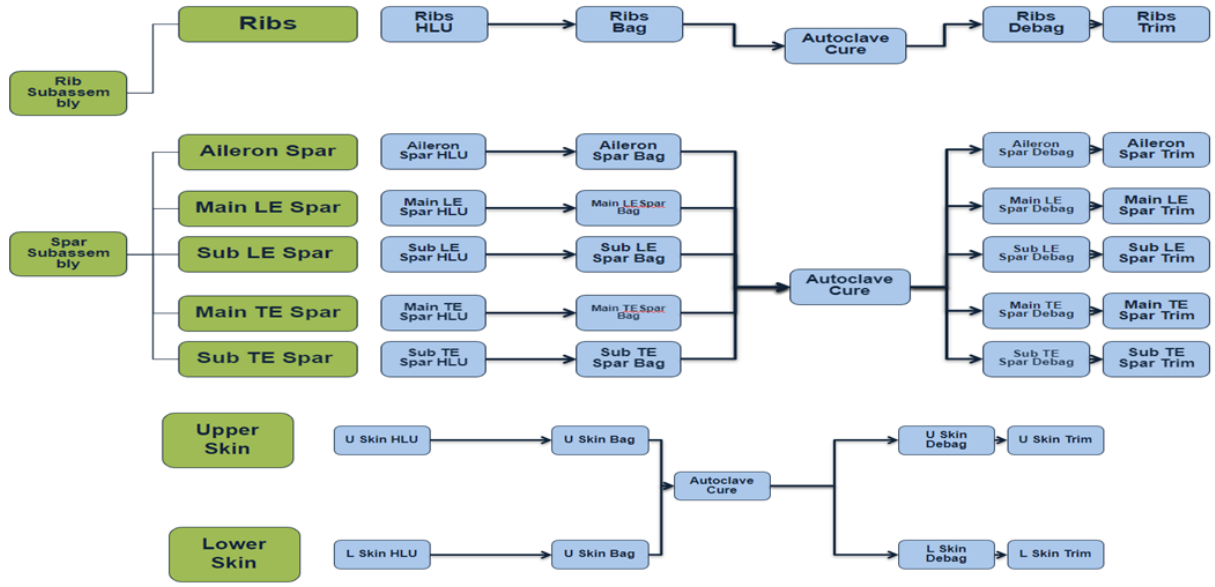


Figure 121. Process flow chart for each subassembly line for the process using the Autoclave cure.

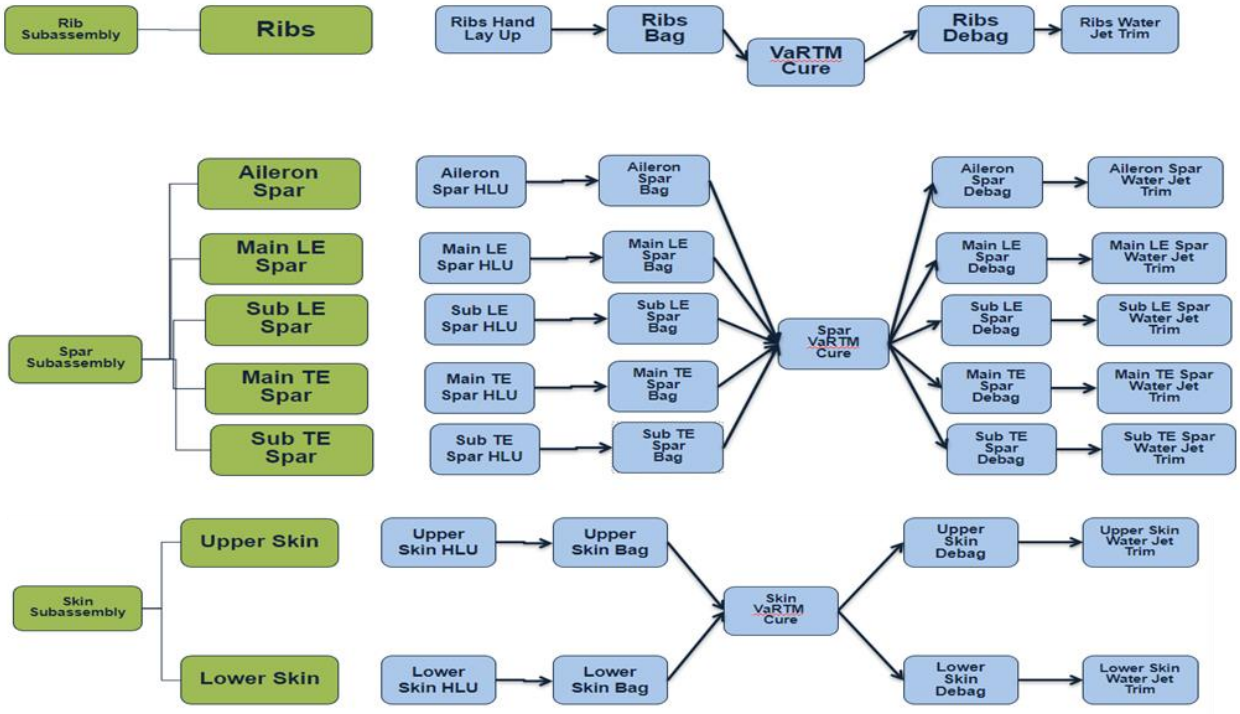


Figure 122. Process flow chart for each subassembly line for the process using the VaRTM cure.

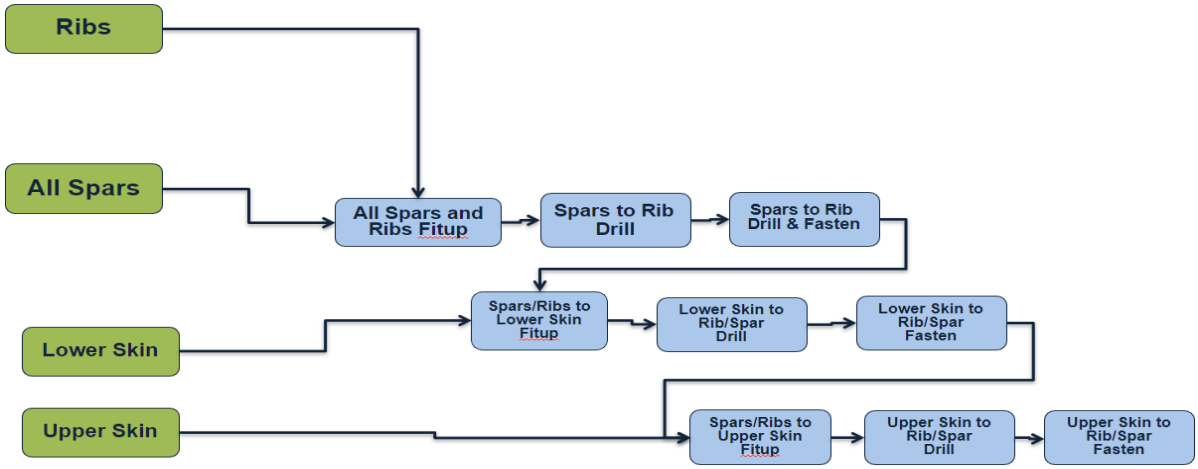


Figure 123. Process flow chart for the final assembly for the process using the Autoclave cure.

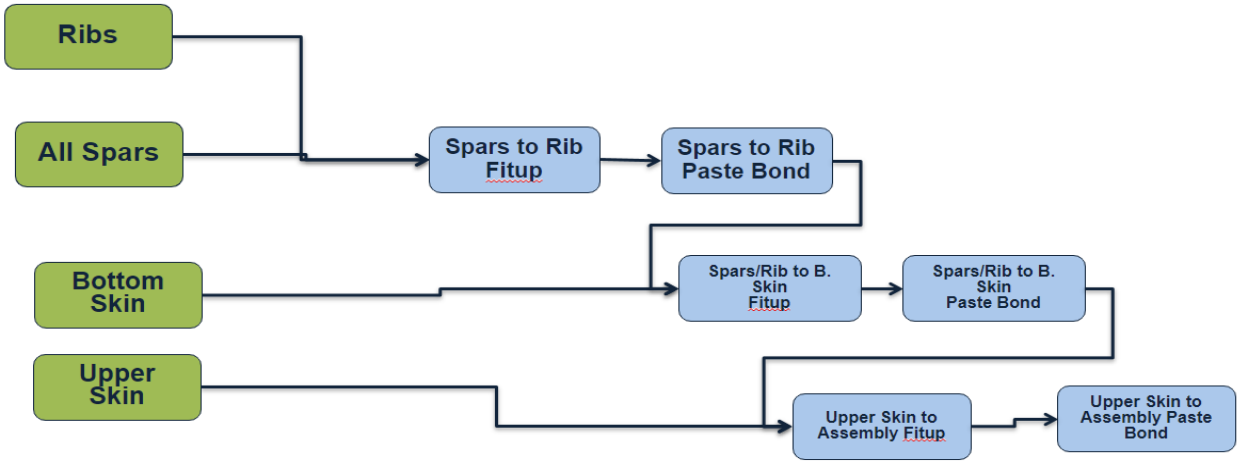


Figure 124. Process flow chart for the final assembly for the process using the VaRTM cure.

Composite Subassemblies: Creating Subassembly Lines

The base components used to create our subassembly lines include “Source”, “Separator”, “Path” and “Server”. A “Source” should be added for each process line or part type, a “Separator” is for each step that separates entities from a parent entity, “Path”’s are used to connect a server’s input to another server’s output, and a “Server” is for each step in the subassembly that can be modelled with a processing time or sequence of subtasks. These components can be found in the “Facility” tab under the “Standard Library”, shown in Figure 125.

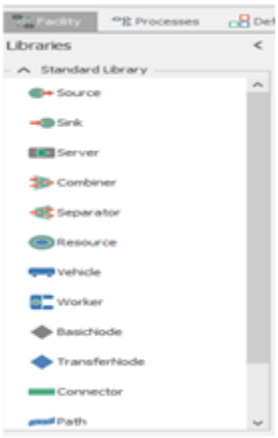


Figure 125. Standard library of facilities

Each sub-assembly line has a Tool storage, an HLU station which includes the inspection process, a Bag station, a Debag station, and a Trim station. Each sub-assembly has one Tool Cleaning, one Tool Preparation station, and one Cure station (either Autoclave or VaRTM depending on the model), shown in Figure 121 and Figure 122. All these stations are modelled by servers except for the Debag stations, which uses a “Separator”, which allows the tool outputs to be redirected to the Tool Cleaning. As seen in Figure 121 and Figure 122, the servers are all connected by paths.

From Figure 126, five process lines can be identified. These are the aileron spar, main leading edge spar, sub leading edge spar, main tail end spar, and sub tail end spar. Therefore, four “Source” objects need to be inserted and labelled “Tool Storage” (shown in Figure 127). The source objects must be connected to two servers before the creation of the process servers. They are the “Tool Clean” and “Tool Prep” servers.

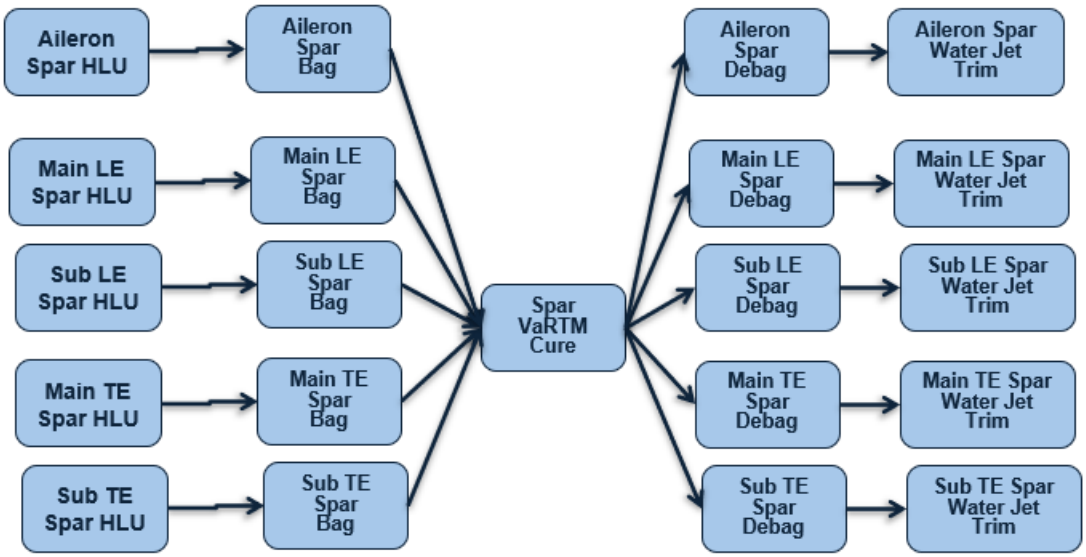


Figure 126. VaRTM Spar Subassembly

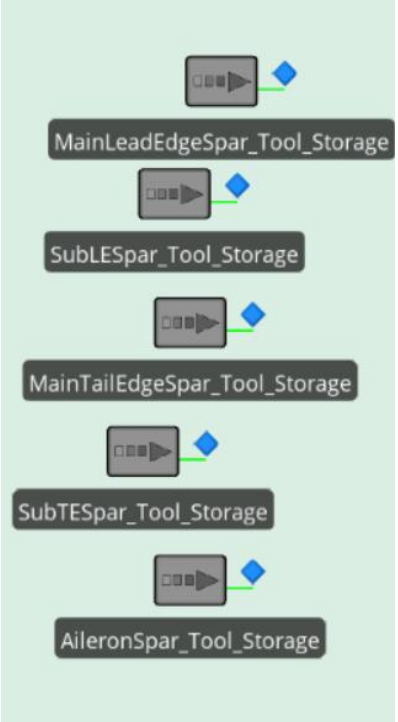


Figure 127. Tool Storage's created for each Spar part

Each subassembly receives one of each; the spar subassembly will have one "Spar Tool Clean" and one "Spar Tool Prep" server. Each process will receive their own servers for the hand lay up, bag, and water jet trim process. Since the debag process separates two parts, a "Separator" must be inserted to represent the process. The VaRTM is shared between processes, therefore; only one VaRTM server is needed.

After the insertion of these servers, paths must be connected between the inputs and outputs of the servers. All the tool storage units should connect to the input at "Spar Tool Clean". Similarly, the output at "Spar Tool Prep" must be connected to all five "Hand Lay Up" servers. The outputs of the "Bag" servers should connect to the input at the Spar VaRTM, and the output at Spar VaRTM should connect to the inputs of the "Debag" servers. The "Parent Output" of the "Debag" servers should connect to the "Water Jet Trim" servers. The "Member Output" of the "Debag" servers should connect to the input at "Spar Tool Clean" to clean and reuse entities.

The steps above are repeated for the other sub-assemblies, including the Rib and Skin subassemblies for both the VaRTM and Autoclave models. The completed subassemblies for the VaRTM process are pictured below in Figure 128. The Autoclave model should have a similar structure.

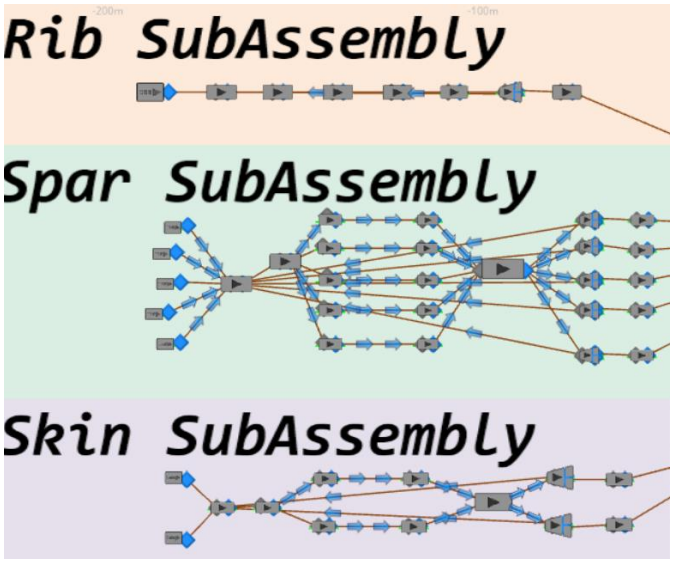


Figure 128. Zoomed out view of the final sub-assembly lines

Composite Subassemblies: Creating the Final Assembly

According to the processing diagram in **Figure *123*** and **Figure *124***, the final assembly will need a few main components including a "Combiner", which groups entities together with a parent entity, and a "Sink", which denotes the end of a process line. Similar to the subassembly lines, the final assembly also utilizes "Servers" and "Paths".

An Example Final Assembly: VaRTM Final Assembly

The output of the "Rib Water Jet Trim" was connected to the "Parent Input" of the "Spars to Rib Fitup" "Combiner". The outputs for all the spars' "Water Jet Trim" were connected to the "Member Input" of the "Spars to Rib Fitup". The output of the "Spars to Rib Pastebond" "Server" was connected to the "Parent Input" for the "Spars/Rib to B. Skin Fitup". The output of the lower skin's "Water Jet Trim" was the "Member Input" for the same fitup. Finally, the output of the "Spars/Rib to B. Skin Pastebond" "Server" was connected to the "Parent Input" for the "Upper Skin to Assembly Fitup". The output of the "Upper Skin to Assembly Paste Bond" was connected to the sink. The final facility view of the final assembly is shown below in Figure 129.

It should be noted that the travel time between servers can be specified by adjusting the length of the path or travel speed of an entity.

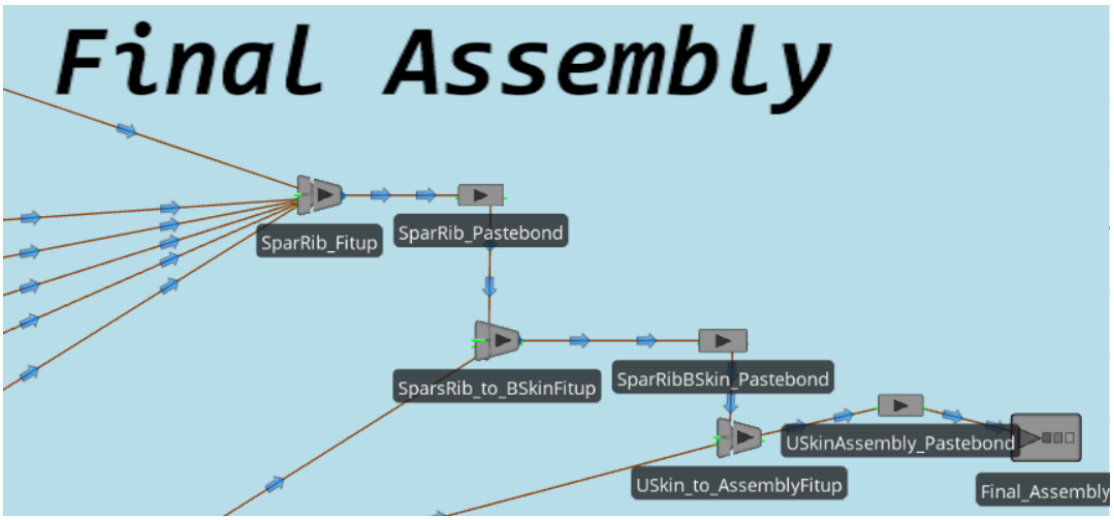


Figure 129. Final Assembly of the VaRTM model.

Composite Subassemblies: Adding Workers

To add a "Worker" that represents workers of the assembly, from the facility "Standard Library", select "Worker" and drag and place the worker into the assembly. Click on the worker and set the properties. In its properties, change the "Capacity Type" to WorkSchedule, and set the "Initial Work Schedule" to the value we want (ours is from our file bind). To set the initial and final population to the worker, right click on "Initial Number In System", and hover on "Set Reference". Then, select "Create New Referenced Property", which should redirect to "Property" in the model's "Definitions" tab (Figure 130). The default value can then be set to any value since the worker amount value can now be modified within each experiment.

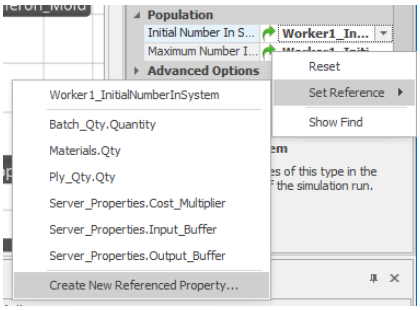


Figure 130. Steps to adding workers

Definitions

Definitions: Entities

Several "ModelEntity" objects need to be inserted into the facility. These "ModelEntity"'s can be selected from "Project Library" in "Libraries" side tab under the "Facility" tab, shown in Figure 131. To create a model entity, simply select the "MidelEntity" and drag it into the workspace. For each subassembly line, the following entities are needed: an entity for the each tools in "Tool Storage", an entity for the mold is needed for every "Hand Lay Up" , an entity for the cured part is needed for each subassembly line although there is only a single "VaRTM" or "Autoclave" server. Additionally, the part itself needs an entity for every "Debag" server.

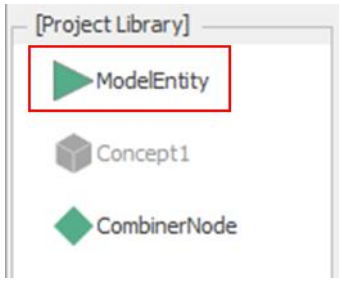


Figure 131. The Project Library in “Libraries”, with “ModelEntity” shown in a red square.

These separate entities help with distinguishing parts by process and those that separate several parts.

Definitions: Material Elements

Material elements are used to define a material consumed or produced in the model system. These materials will be stored in separate storage locations. To Set Material Elements, go to the “Definitions” tab, click on “elements”, then click on “Material” under “Supply” in the top bar. Double click the newly created material to rename, shown below in Figure 132.

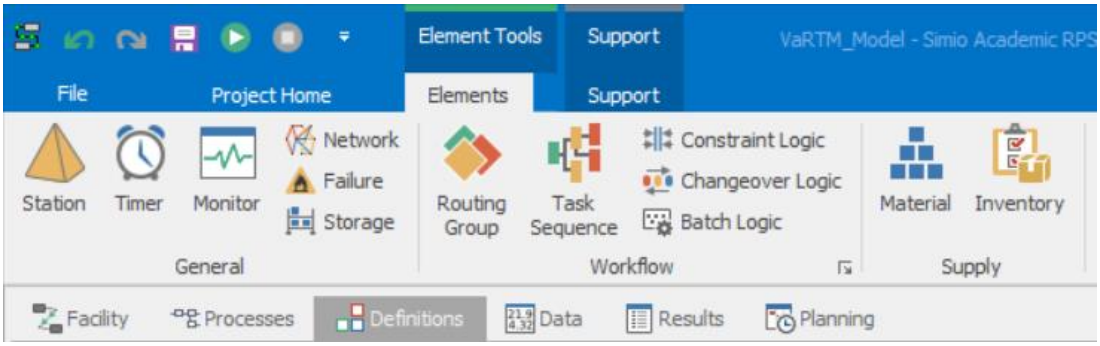


Figure 132. How to find the “Materials” tab in “Definitions” in Simio

For the Autoclave Cure, the final “Material Elements” used can be seen in Figure 133, and the “Material Elements” required include the raw materials: Main Leading Edge Spar, Sub Leading Edge Spar, Main Trailing Edge Spar, Aileron Spar, Sub Trailing Edge Spar, Rib, Upper Skin, Lower Skin and Pre, highlighted in orange; the materials post-trim at the end of each subassembly lines include the parts with the word “Final” tagged in front of the parts, e.g. ‘Final_Main_Spar’, highlighted in green; materials for the Final Assembly include the Fitup parts, Fitup Spar & Rib, the Fitup Spar-Rib-Lower-Skin part, and the Fitup Spar-Rib-Lower-Skin-Upper-Skin part, and the Fastened parts, Fastened Spar-Rib, Fastened Spar-Rib-Lower-Skin, Fastened Spar-Rib-Lower-Skin, and the Final Wing/Wingbox, highlighted in blue. The raw materials and post-trim parts are the same as those for the VaRTM model, however, the VaRTM model does not have the fastened parts due to the paste bond process.

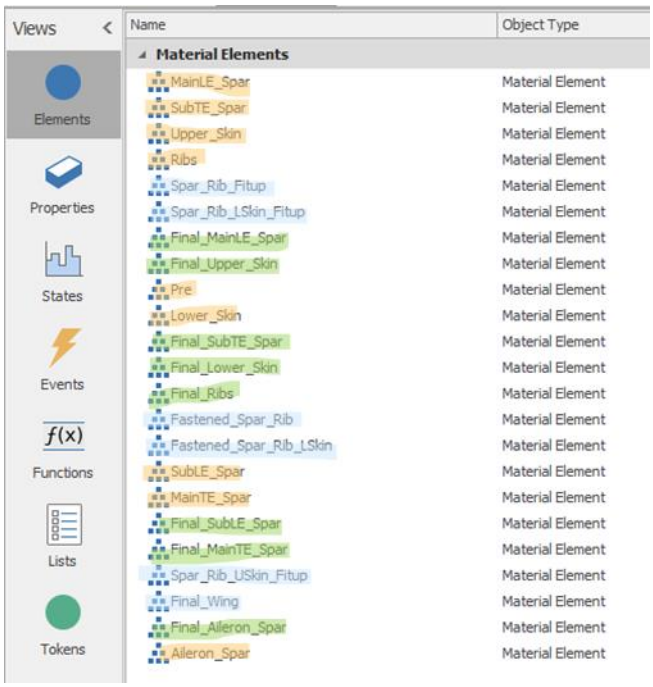


Figure 133. Material Elements used in the Autoclave model

Data Tables, Inputs, and File Bindings

Data Tables

Several tables need to be created under the "Data" tab. The process time values, capacity, buffers, work schedules, cost multipliers, tool quantities, and batch quantities are all input values that must be imported for experiment runs. These inputs are provided in the form of an Excel sheet that may be edited with the inputs adjusted as needed. The data tables requiring information from the Excel sheet must be bound to the Simio table. To import an Excel table to Simio, select the "Data" tab, select the "Schema" tab under "Table Tools", and select "Add Table", shown in Figure 134. The table can be renamed as desired. For each table, we can define the properties of the corresponding columns and the data types, too. For example, to define the property or column as an object, select "Object Reference Property" and then "Object" (more of this will be explained below).

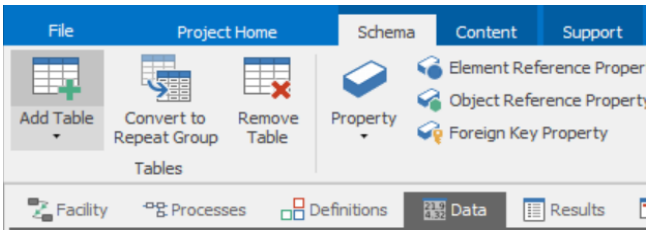


Figure 134. How to add a Table from the "Data" tab in Simio.

File Binding

To bind the tables to an external Excel file, select a table, and select "Create Binding", Under "Create New Import Binder" click "Excel Data Importer" and specify the file to bind the model with. Under the "Import Binding – Property Editor", select the Excel data file or named specific range

from which to pull information. For the model to connect and extract inputs from the Excel file without errors, the model needs an option titled “Auto-set Table Row Reference” turned on. This can be done by clicking on the column header to display properties, expanding the “Advanced Options” line, and set “Auto-set Table Row Reference” to True.

To import the data after binding the tables, select the “Import Table” button which is under the “Content” tab, shown in Figure 135. Simio allows automatic data imports, to control the frequency of an automatic data import, select “Binding Options” and select “Automatic”. If “Automatic” is selected, the model will import the data form the Excel file upon every run. Below are instructions on how to add specific tables.

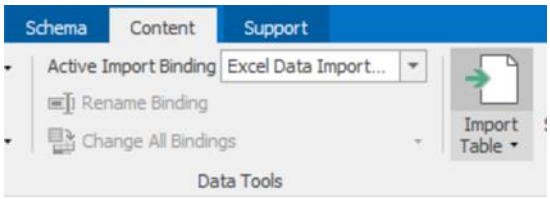


Figure 135. Where to find “Import Table”

Specific Tables

Adding Server_Times Reference Table

The Server_Times schedule is a table with each server, their processing times, the buffer capacities, and cost multipliers. First, create a new “Table”. Then, add the first column by selecting “Object” in “Object Reference Property” to be the column for the Servers. The next column added is the scheduling for each server, which can be added by selecting “Property” and then selecting “Schedule”. For the buffer capacity and batch quantity of each server, select “Property” and then “Real”. For the cost multiplier, input buffer, and output buffer column, select “Property” and then “Integer”. For the processing time column, select “Property” and then “Expression”. The Excel Simio bind should concatenate the mode with a triangular distribution to form the expression. Once the table is set up, bind the Simio model to the Excel input file as described above.

Adding Tool_Quantity Reference Table

The Tool_Quantity table shows the model how many of each tool the model will start with. First, create a new “Table”. Create a column for the storage object by selecting “Object Reference Property” and then “Object”, then add an integer-based column by selecting “Property” and “Integer” for a column describing the quantity of tools per storage. This is all the information needed for this table, and so all that is needed is to bind an Excel file to the table if needed.

Adding Batch_Qty Reference Table

The batch quantity tells the model how much of an entity must be batched. Create a new “Table”, and select “Object Reference Property”, and then “Object”, as the entity identifying column. Then, select “Property” and “Integer” as the quantity of objects required for batching. Then select “Element Reference Property” and then “Material Property” to create a table as the material associated with the batching process. For our model in particular, since each full wingbox requires six ribs, the “Rib_Part” entity is selected for the object and the quantity is set to 6. This ensures only fully cured and trimmed ribs are a part of the final assembly. The material “Final_Rib” should be set as the material associated with the part.

Adding Destination Reference Table

The destination reference table is used to tell a server that processed multiple parts where/ which station each of those different parts or entities move to next. To create this table, create a new “Table”, and add a column for the entity by selecting “Object Reference Property” and selecting “Object”. Next, to create a column for the destination “Node”’s, select “Object Reference Property” and then “Node” as the column identifying the destination.

For each subassembly’s tool entity, the destination should be to the subassembly lines’ input at the “Hand Lay Up” server. For each cure entity, the destination should be to the input at each of the subassembly lines’ “Debag” server. The part entity destination is based upon the final assembly destination. For example: the “Rib_Part” destination is the “Parent Input” at the “SparRib_Fitup” server, and all the spar parts are sent to the “Member Input” at the “SparRib_Fitup”.

Adding Materials Reference Table

For the materials table, the only material we give a material quantity for is the “Pre” starting material. This quantity is se t to 10,000,000 to allow simulation runs assuming the necessary prep material is available for the set simulation time (one month). To create this table, add a “Table”. In the table, select “Element Reference Property” and then “Material” as the material identifying column. Then, add “Property” and “Integer” as the quantity of the material.

Work Schedules

The work schedules are used to define the model resource capacities over time by creating different work schedules. This allows us to have more operation on specific servers depending on the schedule. To access the work schedules, select “Planning” and select “Work Schedules” as shown in Figure 136. Make sure to select “Pattern Based”. To create a new day pattern, select “Day Pattern”. For each day pattern, click the “+” button to expand the “Day Pattern”, and add working shifts by setting a start and end time or duration.

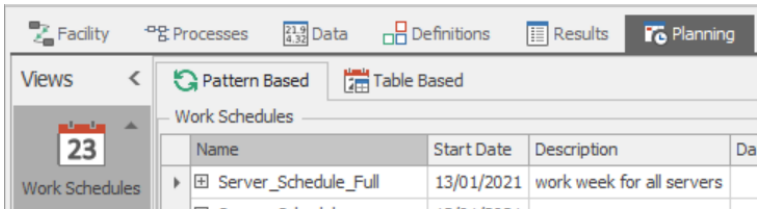


Figure 136. Where to find “Work"

The “Value” is the capacity the process stations have on shift and can be called from the Server_Times table by “Server_Times.Capacity”. The “Cost Multiplier” factors the cost associated with the work period and resources and can be called from the Server_Times table by “Server_Times.Cost_Multiplier”

Under the “Pattern Based” work schedule, set the shifts to the correct days associated with those shifts as shown in Figure 137 below. In Figure 137, it is also shown that Day Pattern’s can be created and customized, and their work periods can be customized through start time, duration, and times, as well as values and cost multipliers. For the VaRTM and Autoclave, the times were adjusted by the utilization and capacity of the assemblies.

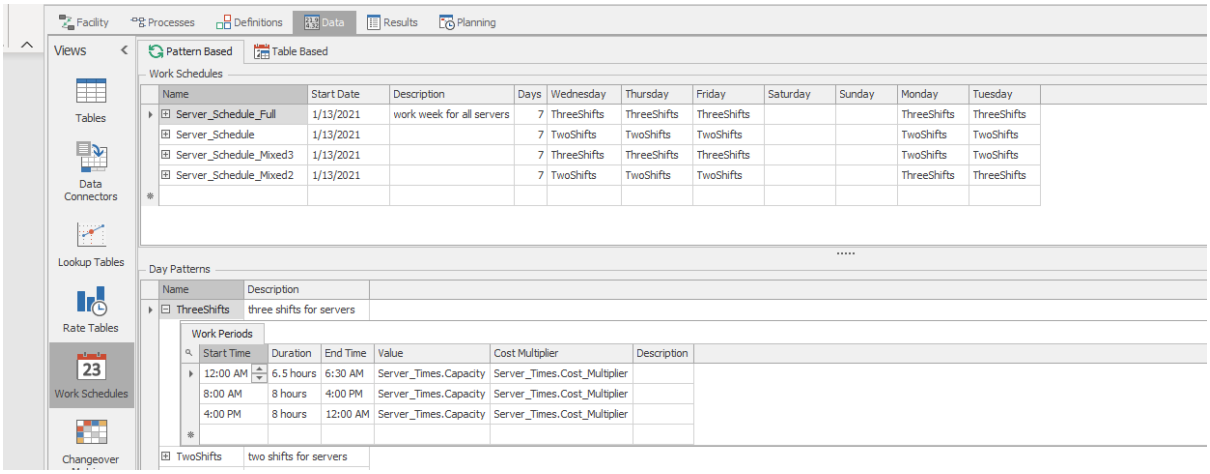


Figure 137. Pattern Based Work Schedule

Server Processing

Processes

The “Processes” tab is used to define standard processes that will be automatically executed in each simulation run. The “Processes” tab can be found next to the “Facility” tab as shown in Figure 138. These process models are comprised of steps, elements, and tokens. The steps are executed by tokens which in turn alter the state of the elements. The behavior of user-created objects can be defined here. Some processes may be triggered by events, automatically by Simio, or within the model logic. In a Server’s “Properties”, “Add-On Process Triggers” allow for specific processes to be triggered at specific points within the model logic. These do not require a triggering event as they are executed by the object. Several categories were created to organize the processes, these include: Consume & Produce, Entity Change, Entity Change +, Final Assembly, and Quality Assurance.

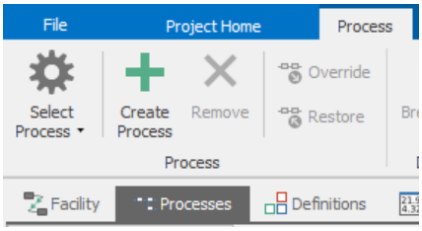


Figure 138. “Process" tab in Simio

Processes can be added to a station by selecting a server on the “Facility” tab, opening the “Add-On Process Triggers” tab under the server properties, selecting the kind of trigger to add, and clicking on the drop arrow to select “Create New”, shown to the right in Figure 139. Processing will trigger the events upon processing while after process will trigger events post processing. Alternatively, the “Processes” tab has a button called “Create Process” listed under “Process Tools – Process”. The name and categories can be adjusted by selecting the white background area of a process. All these processes need to consider the processing of workers.

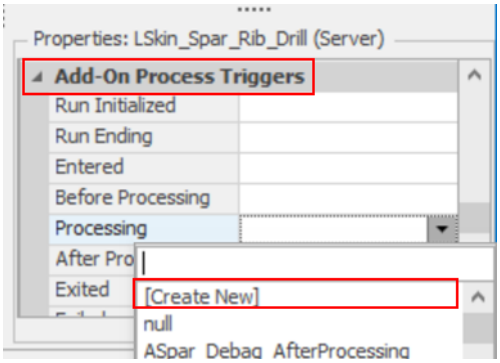


Figure 139. "Add-On Process Triggers"- where to add a process to a server.

Workers

To consider the movement of workers throughout the assembly, each processing must "Seize" workers at the start of a process and "Release" workers at the end of a process. This consideration is integrated into each type of processing that will follow. To add "Seize" into a process, drag "Seize" from the side into the process. In the properties of the seize, click the three dots in Resource Seizes shown in Figure 140. This will open a dialogue box. In the pop-up box, select "Add". Then in the "Properties" select and assign a worker (i.e. Worker 1) as a resource to the "Resource Name", shown in Figure 141. To add "Release", drag "Release" into the process, and change the properties the same way as how the properties for the worker's seize step was done.

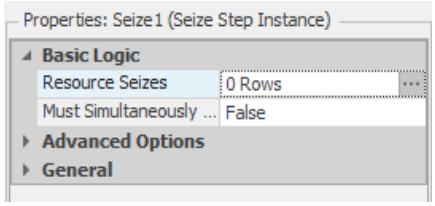


Figure 140. Properties of "Seize" process for the processing of Workers

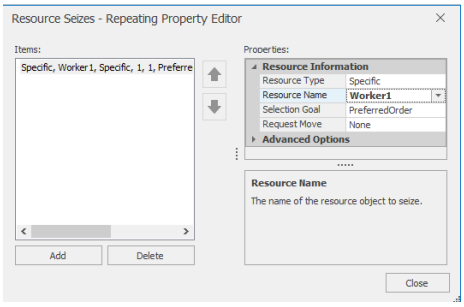


Figure 141. Pop-up dialogue to set "Resource Seizes" for "Seize" process for the processing of Workers.

Consume & Produce

In this process category, one or more materials are used ("Consume") in order to create ("Produce") a new material during a process. During these processes, a "Worker" has to be "Seized" at the very start before the "Consume" step and "Released" at the end of the process, after the "Produce" step. Server processes used in the final assembly "Fitup" processes and the

subassemblies' "Trim"/"WaterJet Trim" are a part of this category. An example is the Aileron Spar Trim processing shown in Figure 142.

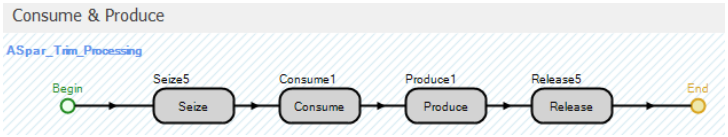


Figure 142. Aileron Spar Trim Processing

The step "Consume" will consume specified quantities of a material. The token is held at "Consume" until all the material consumption requirements are completed. The "Produce" step produces specified quantities of a material. The "Create" step is used to create a new entity, create copies of an entity, or create new token references for existing objects. The "Destroy" step will destroy the entity sent to it. The "Transfer" step allows for entities to be transferred across different locations or location types.

For the **Trim/WaterjetTrim** Processing, create a new processing. Under "Common Steps", drag "Seize" and configure its properties as shown in the ***Workers*** section, then under "All Steps", drag "Consume" into the process. The properties of "Consume" should be set to the subassembly line's main material (i.e. "AileronSpar" for the aileron spar line). Then, drag and place "Produce" in front of the end step. The properties of "Produce" should be set to the final material version of the subassembly line's main material (i.e. "Final_Aileron). Drag, place and add a "Release" step as shown in the ***Workers*** section.

For **Fit Up** Processing, create a new process. Under "Common Steps", drag "Seize" and configure its properties as shown in the ***Workers*** section. Then, drag "Consume" into the process. The properties need to be adjusted for material and quantity. The final material version of the part or the partial assembly material should be set as the material consumed. The quantity for the number of materials required can be referenced to the batch quantity table or specified by changing the value. For every other part that are part of the fitup, add an additional "Consume" into the process for every part being attached to the subassembly. Drag and place "Produce" in front of the end step. The properties should be set so that a partial final assembly material or final product is produced. Drag, place and add a "Release" step as shown in the ***Workers*** section.

Entity Change

These are processes that creates and destroy part entities after processing; creates a new object from materials. Servers consisting of entity changes include: Hand Lay Up, the Rib VaRTM, and Debag. An example (the Aileron Spar Hand Layup processing) is shown in Figure 143.

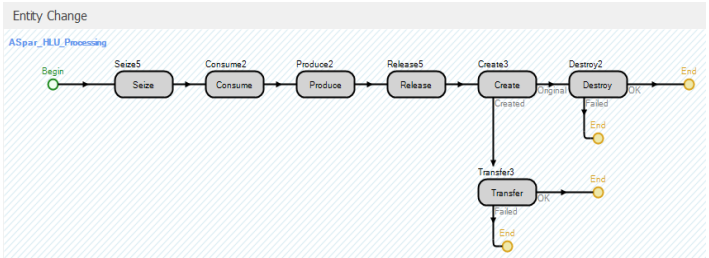


Figure 143. Aileron Spar HLU Processing

To create processing for the **Hand Layup**, create a new process, under “Common Steps”, drag “Seize” and configure its properties as shown in the ***Workers*** section, then drag “Consume” into the process. For the HLU, the material consumed should be “Pre” with its quantity set to 10. Drag “Produce to the right of “Consume”, and the material that the subassembly line is named after should be what is produced (i.e. “AileronSpar” for the aileron spar line). Under “Common Steps”, select “Create” and place it behind the “Produce” step. The mold entity of the subassembly line should be created (i.e. “ASpar_Mold”) in this step. Drag, place and add a “Release” step as shown in the ***Workers*** section. Under “Common Steps”, select “Destroy” and place it to the right of “Create”, where the original entity will be sent to the destroy path. Under “Common Steps”, select “Transfer” and place it below “Create”. The newly created entity will be sent to this transfer path, and the basic logic should set the location type of “To” to “Station”. The station name should specify the output buffer the entity will be sent to (i.e. “ASpar_HLU.OutputBuffer”).

For the **Cure (VaRTM or Autoclave)** Processing, create a new process and under “All Steps”, drag “Seize” and configure its properties as shown in the ***Workers*** section, then drag, place and add a “Release” step as shown in the ***Workers*** section. Then, drag “Create” and place it behind the beginning step. In this process, the cure entity should be created (i.e. “ASpar_Cure”). Drag “Destroy” and place it to the right of “Create”, and drag “Transfer” and place it below “Create”. The basic logic should set the location type of “To” to “Station”, and the station name should specify the output buffer the entity will be sent to (i.e. “Rib_VaRTM.OutputBuffer”).

To create the **Debag** AfterProcessing, create a new process. Under “Common Steps”, select “Create” and place it behind the beginning step. The part entity should be created (i.e. “ASpar_Part”). Drag “Destroy” and place it to the right of “Create”. Then drag a second "Create" and place it to the right of “Destroy”. The tool entity should be created (i.e. “ASpar_Tool”). Then, drag “Transfer” and place one under both “Create” instances. The basic logic should set the location type of “To” to “Station”. The station of the first “Transfer” should be the “ParentOutputBuffer” of the process (i.e. “ASpar_Debag.ParentOutputBuffer”). The station of the second “Transfer” should be the “MemberOutputBuffer” of the process (i.e. “ASpar_Debag.MemberOutputBuffer”), which has a path connected to the Tool_Clean servers.

Entity Change +

These are processes that include a condition-based logic in combination with creating/destroying entities; the VaRTM and Autoclave processes for the Skin and Spar subassemblies are included in this category.

For the Cure Processing (VaRTM/Autoclave), create a new process. Under “Common Steps”, select “Decide” and place it behind the beginning step. The basic logic of “Decide” needs a decide type and condition or probability. For condition based, the condition needs to be in the form of an expression (i.e. “ModelEntity.Is.Aileron_Mold”)—if the condition returns false, the mold entity will be sent downwards, and If the condition returns true, the mold entity will be sent to the right step. The condition and logic of this step is used to determine what mold is used to make a part. Under “Common Steps”, drag “Seize” and configure its properties as shown in the ***Workers*** section, then drag, place and add a “Release” step as shown in the ***Workers*** section. Drag “Create” and place it to the right of “Release”, then the cure entity should be created (i.e. “Aileron_Cure”). Then drag “Destroy” to the right of “Create”, and “Transfer” below “Create” since the entity should be sent to the station output buffer (i.e. “Spar_VaRTM.OutputBuffer”). Then drag additional “Decide” logic below the initial “Decide” process to differentiate between the subasse mbly lines of the entities. For every decide logic, there should be a true and false path available leading to further

"Decide" or "Create" paths, set up as mentioned before. In **Figure** , a sample decision tree for the curing process is shown.

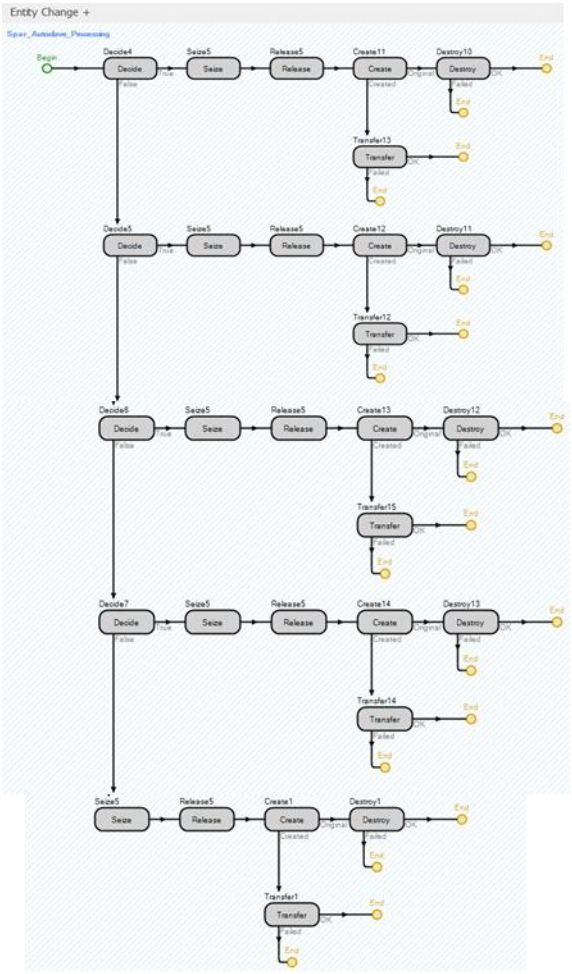


Figure 144. Example decision tree from VaRTM cure processing.

Final Assembly

The final assemblies are made up of creating and destroying entities of partial and full assembly. The Fitup servers have an "AfterProcessing" triggered process associated with them. For this after processing, create a new process and drag "Create" into the process model. The entity type created should be the assembly associated with the step (i.e. SparRib_Fitup would create "Spars_Rib_Part"). Then drag "Destroy" to the right of "Create" and drag "Transfer" below "Create". Like previously, the entity should be transferred to the station's associated output buffer.

Quality Assurance (QA)

The Quality Assurance processes are inspections built into certain process steps. The inspections after the Layup and Cure can be built into the model by creating "AfterProcessing" triggers. The QA category will "Seize" an entity until specific requirements are met. The "Release" step is used to release the seized resources, for example, the Aileron Spar Hand Layup After Processing shown in Figure .

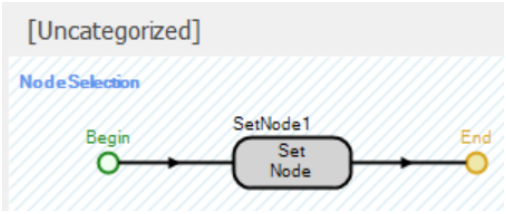
For the Hand Lay Up AfterProcessing, under “Common Steps”, identify and drag “Seize” into the process tree. The basic logic should address the “Resource Seizes” by clicking on the “…” tab. The repeating property editor should open. In the property editor pop-up, add an item; the default properties can be left as is. Under “Common Steps”, the step “Release” should be dragged to the right of “Seize”, the basic logic should address the “Resource Releases” by clicking on the “…” tab. Again, add an item in the repeating property editor and close the tab.

For the VaRTM AfterProcessing, identify and drag “Seize” into the process tree. The step “Release” should be dragged to the right of “Seize”. Utilize the same basic logic properties as the Hand Lay Up.

Figure 145. Aileron Spar HLU After Processing

Uncategorized Processes

The NodeSelection process is an uncategorized process and was created in order to assign the table of destinations to their respective servers. The process requires a single step “SetNode” to be inserted. The “Destination Type” should be “Specific” while the “Node Name” should reference the destination table. (i.e. “Destination.Destination”), shown in Fig**ure** .



Fig**ure 146.** NodeSelection processing

Server & Source Properties

Source Properties

The source properties should address two key items. The first is the “Entity Type” under the “Entity Arrival Logic”. This should be set to the associated tool for the storage/source. The arrival is automatically defaulted to a “Random.Exponential(.25)” “Interarrival Time”. The second thing to address is the number of maximum arrivals. Since it is not possible for the number of tools available to be unlimited, a reference for the quantity of maximum arrivals is used (i.e. Tool_Quantity.Quantity).

Server Properties

The “Process Logic” tab should be the first thing to address for the servers. The Capacity type should be “WorkSchedule” to assign workers as a resource. The “Initial Work Schedule” should reference the schedule column of the Server_Times table. Similarly, the “Processing Time” should reference the processing time column of the Server_Times table. (i.e.

server_Times.Processing_Time). Under “Other Processing Options”, “Immediately Try Seize” should be set to true so that the seize request is not inserted into the server’s allocation queue. This property setting applies only if the server has an input buffer.

Experiments & Results

Experiments are used to setup the extraction of performance values data of selected servers in certain design “Scenario”s. The defined “Scenario”s can then be executed in “Batch Mode”. To create an “Experiment”, right click on the Model we want to create experiments for, and click on “New Experiment”, highlighted in blue in Figure .

Each scenario can apply a different table binding or binding configuration to run during the simulations as long as multiple bindings are made to the table. In order to complete the line balancing for the model, responses such as throughput and utilization were recorded. The excel sheet binding the server properties had the buffer queues, machine capacities, and work schedules varied. The servers overproducing had less resources assigned to it while the servers underproducing took over those resources.

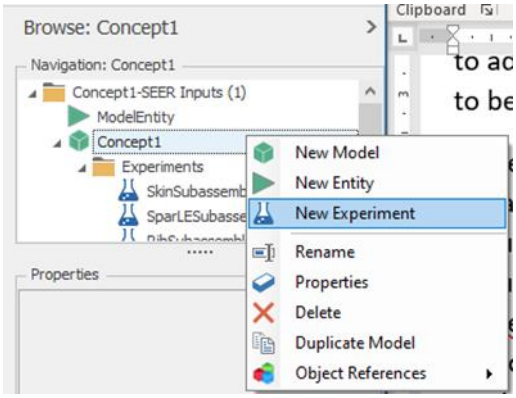


Figure 147. How to create a “New Experiment”