Model-Based System Engineering using System Composer

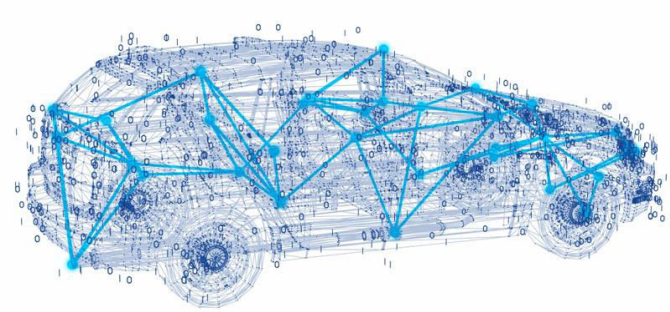
- Adding the code parts to the live scripts. in the examples: Untill now, it is not a concern.
- Progressing (Need a meeting with tadele) : work on the exc, and import them to the live script in the same seq of the slides.
- Not yet: appendix for setting up the mini-drone.
- bring a problem statement / challenges the mbse engineers face.
- Digitalization
- Template for requirements file.
- Update in the req. file (Word?) will update the req. in system composer.
- Steroetypes
- Views
- Analyses

Table of Contents

Introduction

What is Systems Engineering?

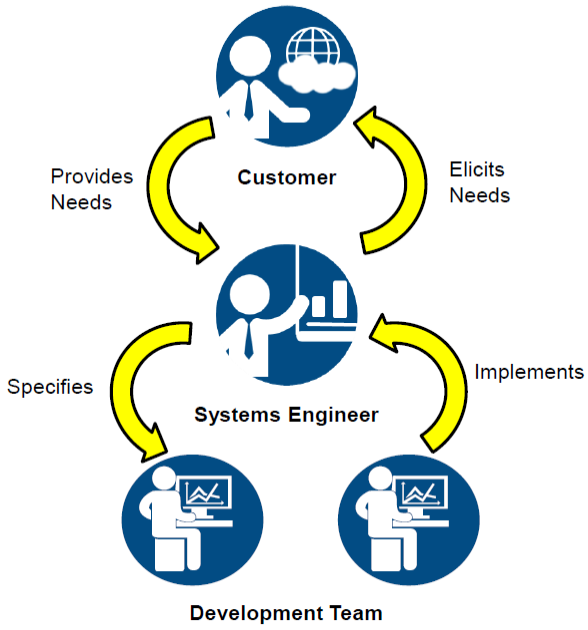
Systems Engineering is an interdisciplinary field of engineering that addresses the design and management of complex systems throughout their life cycles. A complex system comprises interconnected and interdependent components, leading to challenges in understanding, designing, and maintaining its behavior and performance.



For instance, consider the complexity of a car system. The figure depicts various subsystems such as the engine, transmission, electrical systems, and more, all working together to enable the car's functionality. This illustration demonstrates how different components interact, influencing the overall performance of the car. As the complexity of the system increases, so does the need for a systematic and organized approach, which is where Systems Engineering plays a vital role.

In this course, we will explore the principles, methodologies, and tools used in `SystemEngineering.mlx` to tackle such intricate systems efficiently. Through a model-based approach and leveraging tools like System Composer, we will learn how to manage requirements, design architectures, conduct analyses, and ensure successful system development, all while addressing the challenges presented by complex systems like the car example.

The Role of the Systems Engineer:

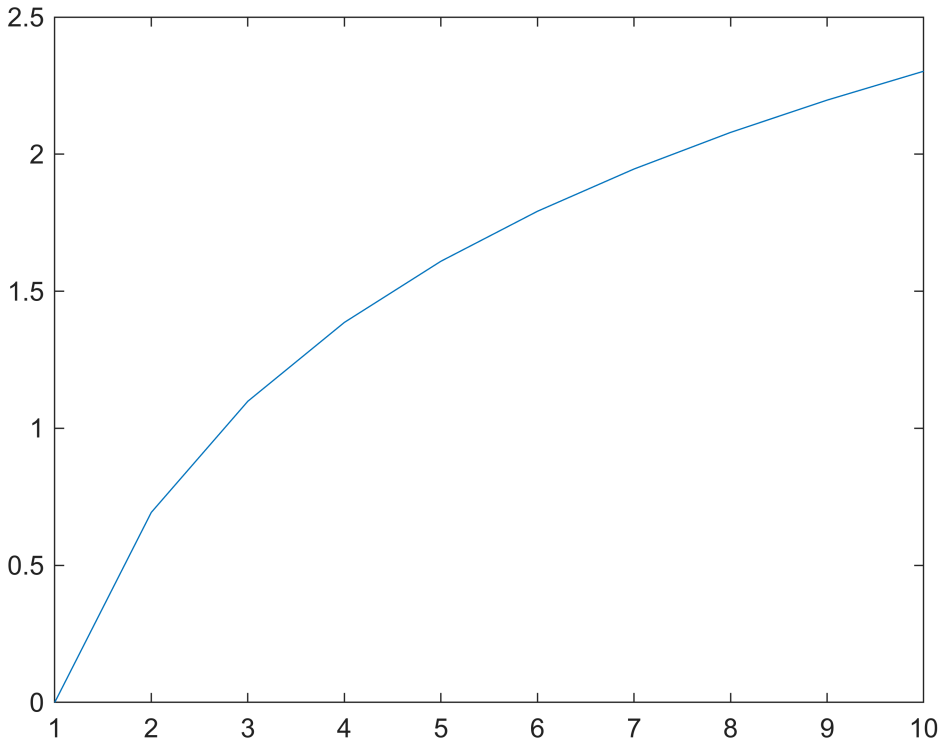


System engineers are crucial in the development lifecycle, with a primary focus on requirements engineering, for more information please refer to `mlx`-file `SystemEngineers.mlx`. They gather, analyze, and define system requirements, translating stakeholders' needs into technical specifications. System engineers design the system architecture to meet functionality, performance, and safety requirements. They oversee subsystem integration, conduct verification and validation to ensure compliance, and provide ongoing maintenance and support.

Effective collaboration with stakeholders and other engineering disciplines is essential. System engineers use techniques like interviews and workshops to elicit requirements and manage documentation for traceability. Clear communication helps them understand stakeholders' expectations and convey design decisions and trade-offs.

Requirements engineering is fundamental to successful systems development, and system engineers are vital in capturing and managing requirements to meet stakeholder expectations.

```
plot(1:10, log(1:10))
```



Model-Based Systems Engineering

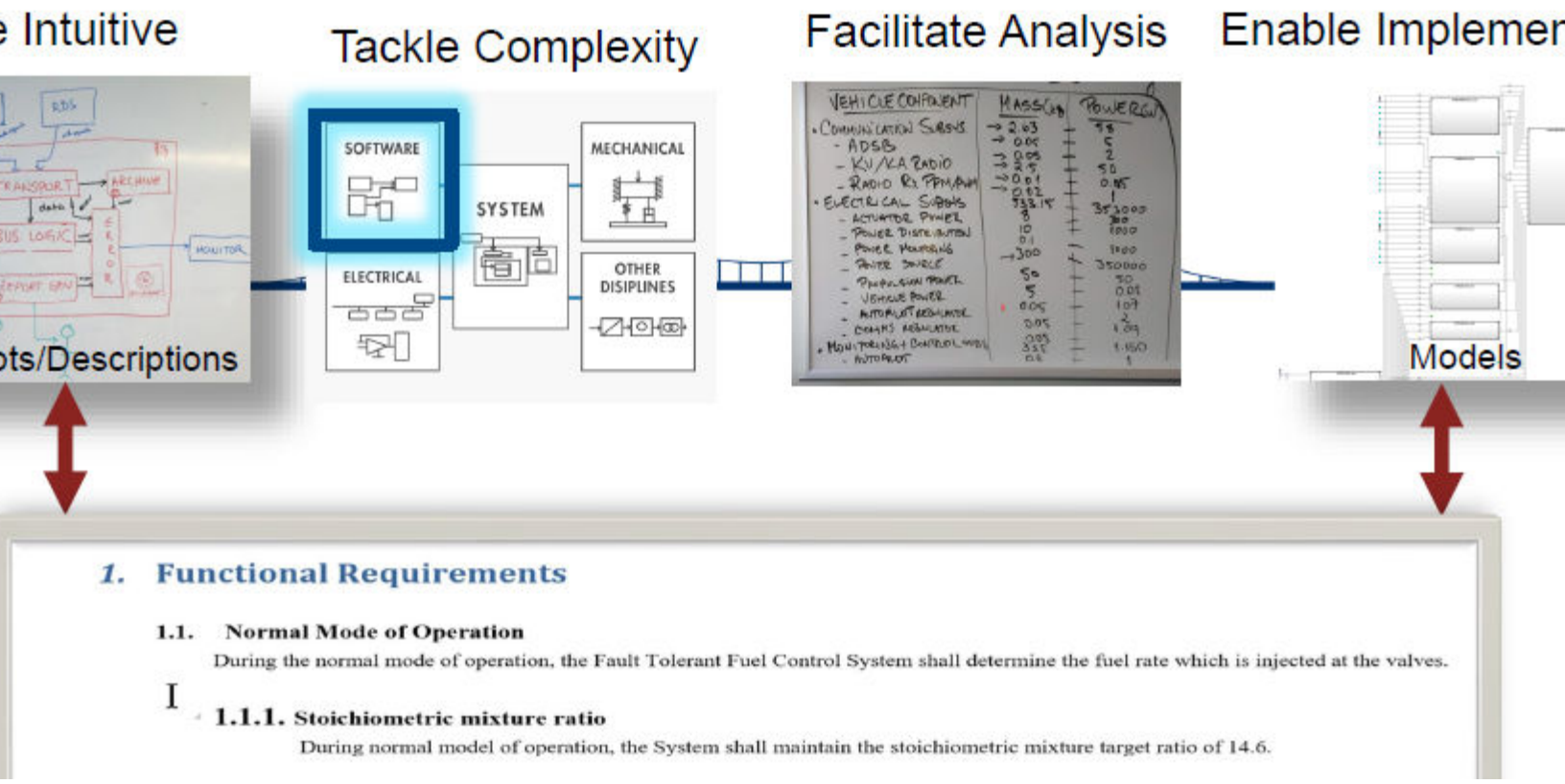
Model-Based System Engineering (MBSE) is an approach that utilizes models as central artifacts to capture system requirements, design, and behavior. One tool that supports MBSE is System Composer, a powerful modeling and simulation tool developed by MathWorks.

System Composer provides a collaborative environment for system engineers to create and manage system architecture models, define system requirements, and conduct simulations and analyses. It integrates seamlessly with other engineering tools, enabling interdisciplinary collaboration and data exchange.

By leveraging System Composer in MBSE, system engineers can benefit from the following advantages:

- Enhanced Visualization and Understanding: System Composer enables engineers to create visual models that represent system architectures, requirements, and interconnections. This visual representation enhances understanding and facilitates effective communication among stakeholders.
- Improved Requirements Management: System Composer offers features for capturing, organizing, and managing system requirements. It supports traceability between requirements, design elements, and verification activities, ensuring that all requirements are properly addressed and verified.

- Early Validation through Simulation: System Composer provides simulation capabilities to validate system behavior and performance early in the development process. Engineers can simulate the system's response to different scenarios, enabling quick identification and resolution of potential issues.
- Analysis and Optimization: System Composer includes built-in analysis tools that enable engineers to assess system performance, identify bottlenecks, and optimize system designs. These capabilities support decision-making and help ensure that the system meets performance targets.
- Collaboration and Integration: System Composer facilitates collaboration among team members by allowing the sharing of models, version control, and change management. It also integrates with other engineering tools, enabling seamless data exchange and collaboration across disciplines.



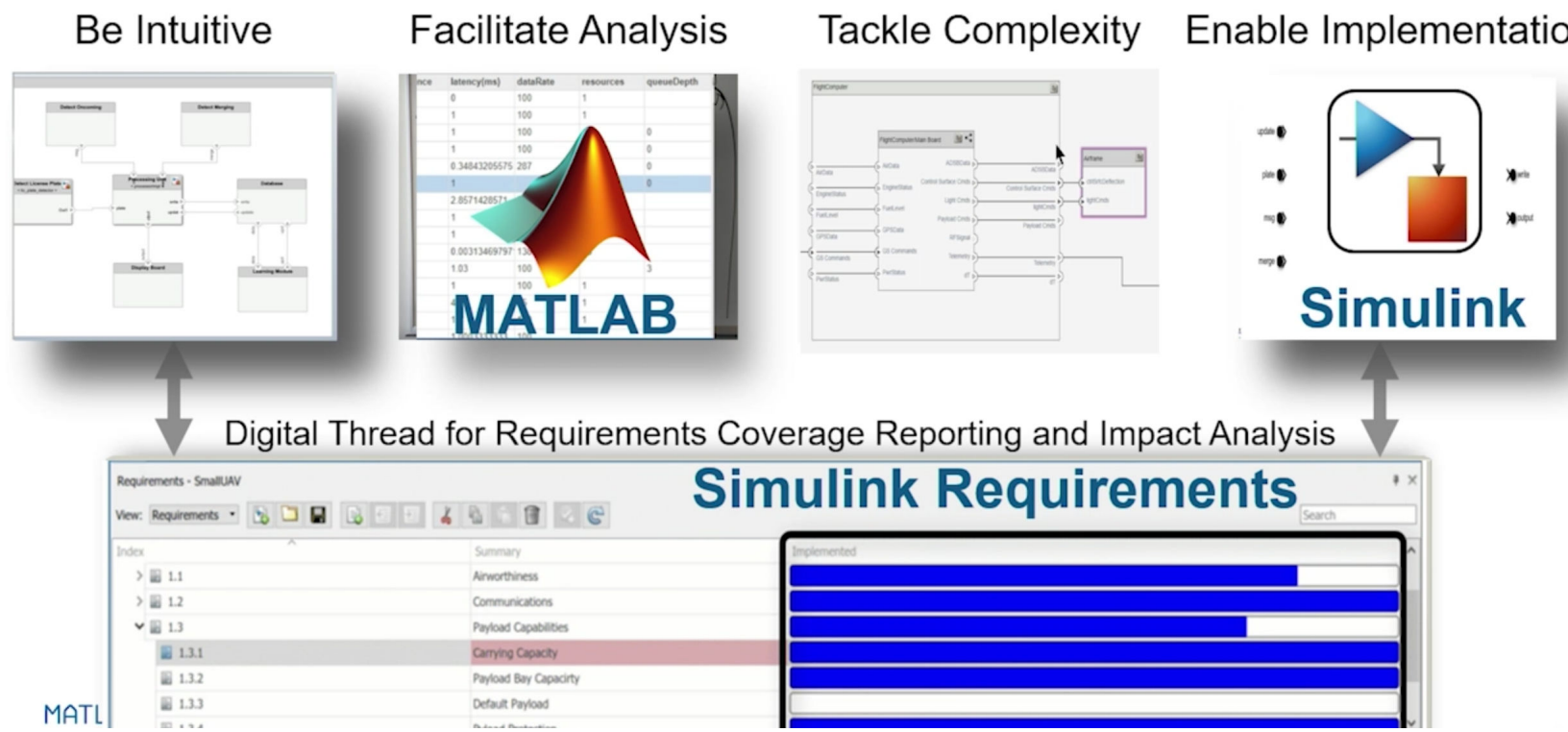
1. Functional Requirements

1.1. Normal Mode of Operation

During the normal mode of operation, the Fault Tolerant Fuel Control System shall determine the fuel rate which is injected at the valves.

1.1.1. Stoichiometric mixture ratio

During normal model of operation, the System shall maintain the stoichiometric mixture target ratio of 14.6.

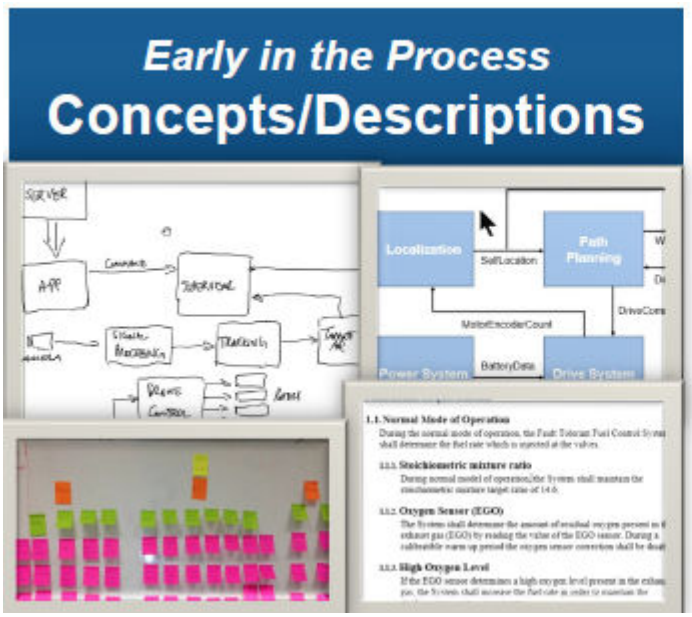


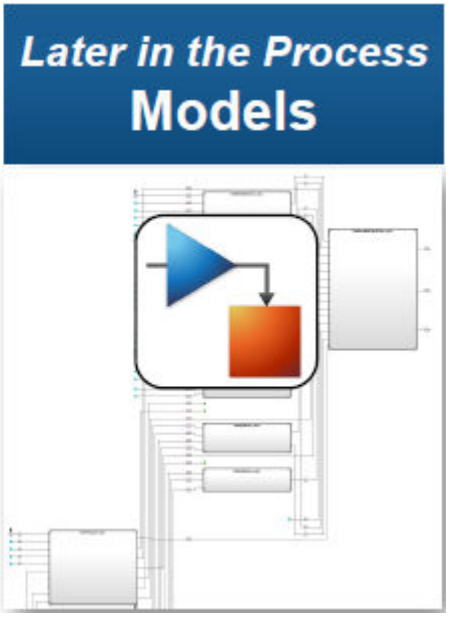
Intuitively design system and software architectures

The system composer in short, is an expressive canvas for specifiying the designs.

This expressve canvas preps analogus to a top level description of what the system does. and that is called an Architecture.

The traditionall method s capturing this canvas on a white-board or on sticky-notes, whereas System Composer allows for digitalizing this aspect.



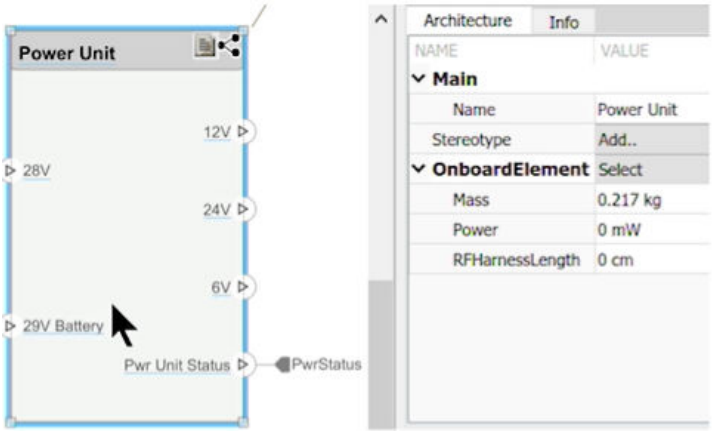


Perform trade studies based on data driven analysis to optimize achitectures

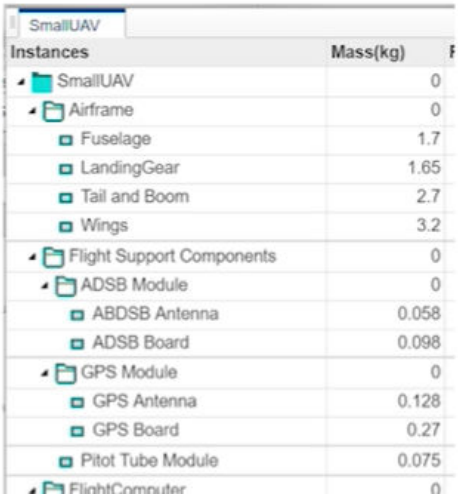
Furthermore, the Sysem Composer gives the ability to specialize the things we skitched on the the aforementioned canvas and provide additionaol information about them.

Stereotyping as called in the systm engineering world. We can also do anlyses of our system with the additional data that we spcified earlier.

Add custom data



Create analysis model

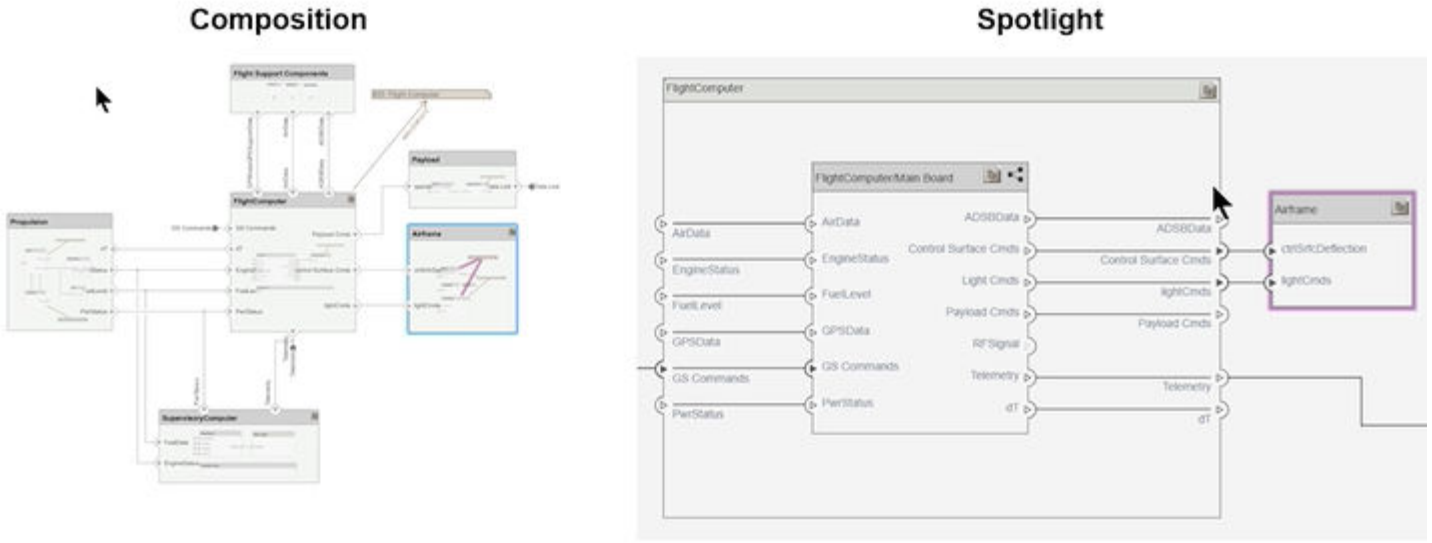


Tackl Architecture complexity: with spotlight

Moreover, we can start managing complixities. To illustrat, as we start building on top of the basic sketches, by adding additional data,

and the updates from all the teams that are working on the project, our design becomes bigger and more complex.

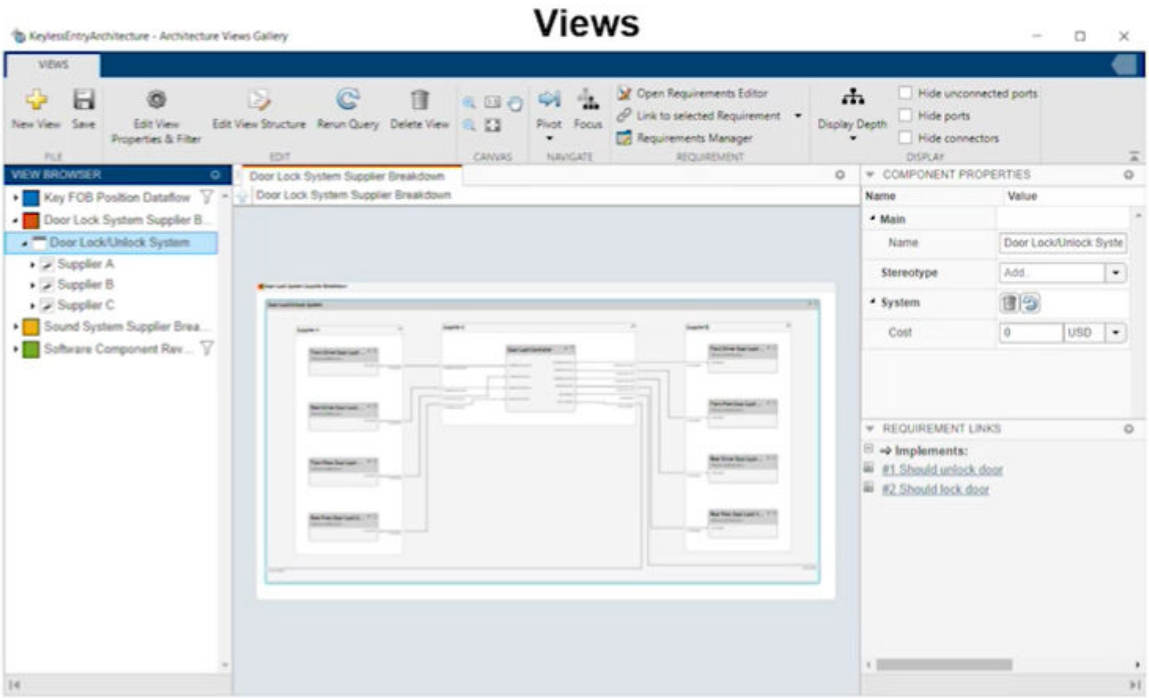
For that we need to understand how can we look at that system and how do we make sense of it. in other words, spotlighting. We do that by getting iside the specific aspects of the system.



Tackl Architecture complexity: with views

if you recall when we talked about stereotypes, we can add additional data. We can then use the information we find in those stereotypes to creat a view

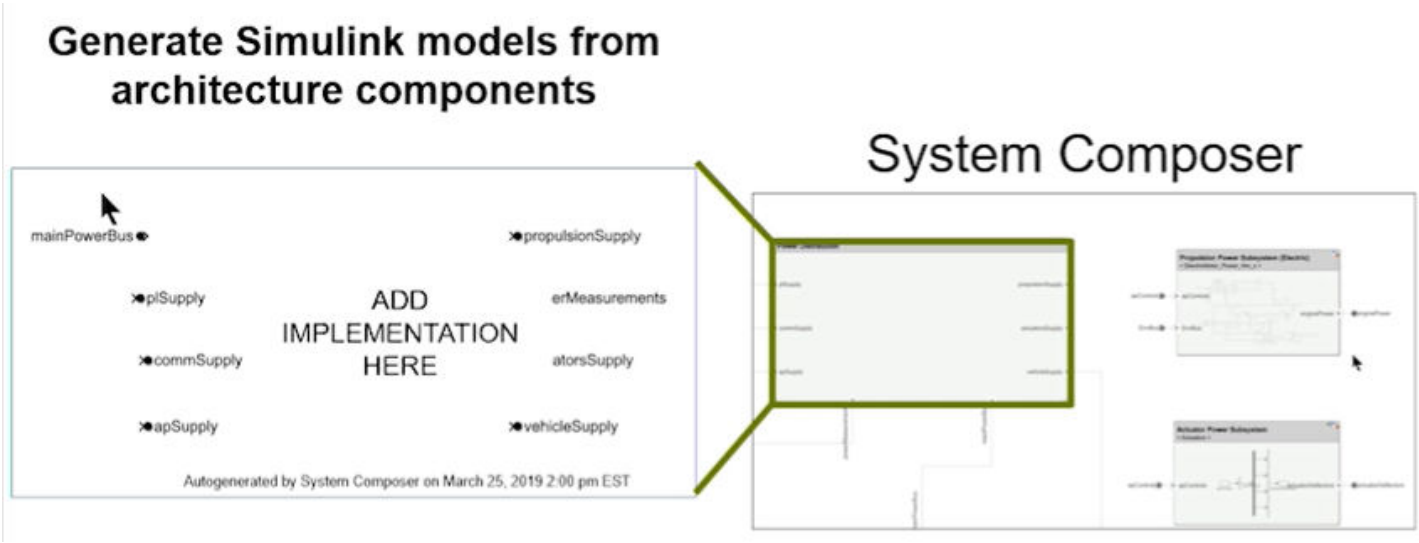
showing only a slice through the design of a preticular context. Example, an electrical engineer can choose to see only the electrical aspects of the system.



System and software architecute connected to implementations in Simulink

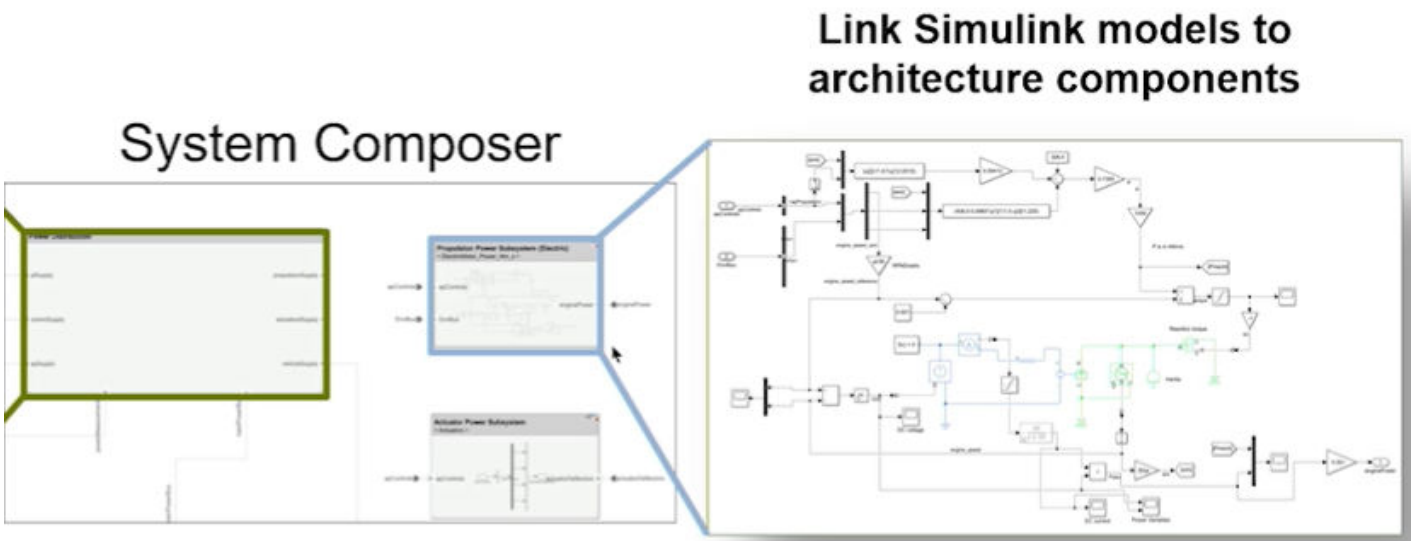
The abstract components that is on our canvas, can have/generate Simulink models underneath them. in this context, the System Composer knows some information about this component,

where we it can create a template to start specifiynig/adding details. use the ability of making changes on our system while being able to directly see how are these changes affecting our system.



We can also work the other way around. if we already have an exisiting model/desing, we can link it to this model to

the architecute component.

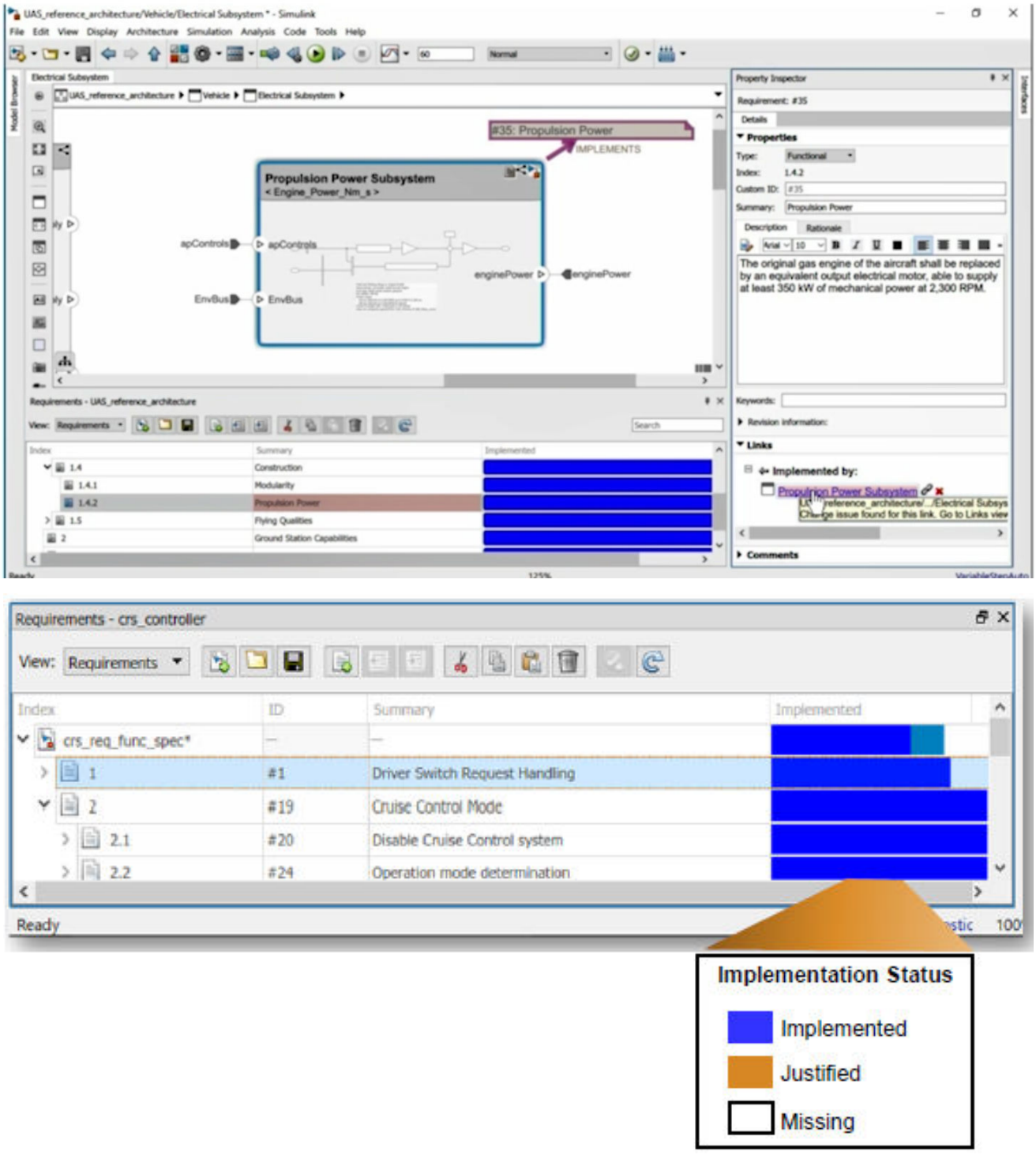


it is also important to know that not all components must have a model linked to them.

Link system models to Simulink Requirements

The Sytem Composer is fully integrated with Simulink Requirements. Meaning, we can link blocks in System Composer models directly to a set of requirements providing traceability.

We can use that traceability to see how is the design work is progressing and to how much does the system meet the requirements.



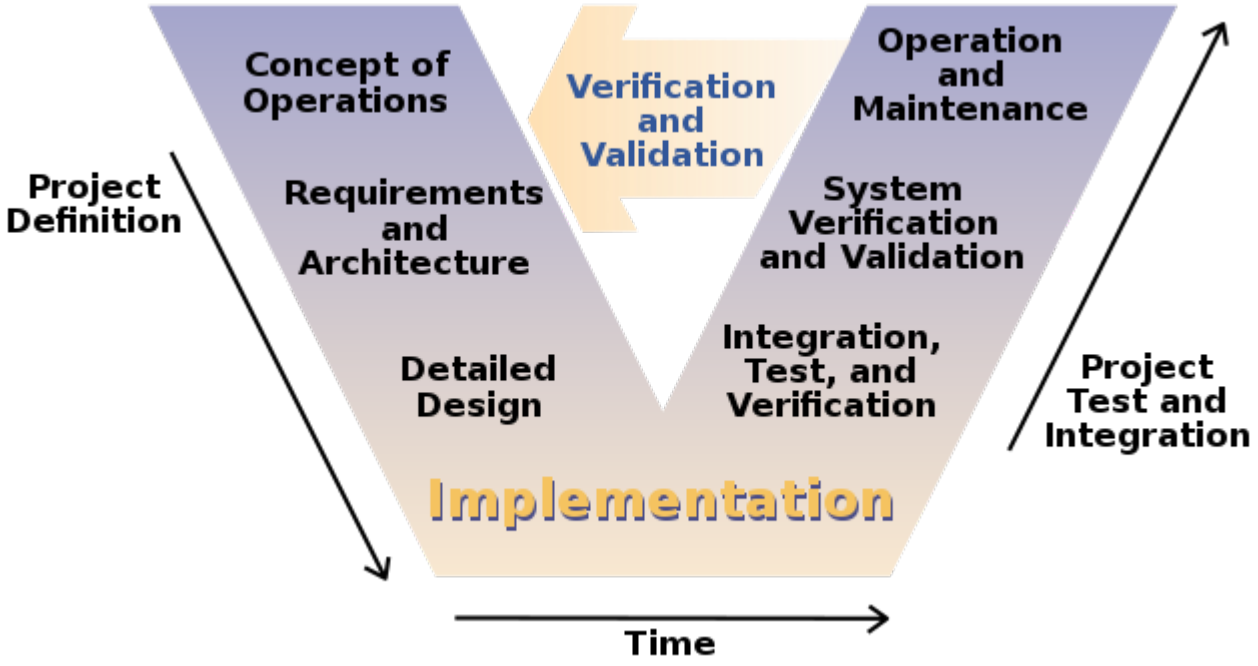
and all in all, we have five key features:

- expressive canvas for describing our designs and there intents
- The ability to do analysis on that design.
- The abilit to manage complixity with spotlights and views of that design.

- Simulation behaviour using Simulink
- Tight integration with Simulink Requirements.as a way of providing traceability.

V-Model in Systems Engineering

Overview of the V-model and its phases.



The V-model provides a structured framework for executing system engineering activities, ensuring a systematic and disciplined approach to the development of complex systems. Let's delve into each phase of the V-model, highlighting the challenges of traditional methods and the advantages of incorporating modern tools like System Composer.

Project Definition Phase:

- Concept of Operations: Defining the system's operational scenarios, user needs, and high-level requirements.

- Requirements and Architecture: Capturing, refining, and managing system requirements, and developing the system architecture.

Traditional Challenges:

- Communication gaps between stakeholders and system engineers.

- Difficulty in managing and tracking evolving requirements.

Advantages of System Composer:

- Enhanced collaboration and communication through a visual modeling environment.

- Integrated requirements management and traceability features.

Implementation Phase:

- Detailed Design: Developing detailed designs for system components, subsystems, and interfaces.

- Implementation: Translating the detailed design into actual system elements, including hardware, software, and firmware.

Traditional Challenges:

- Manual translation of design artifacts into implementation, leading to errors and inconsistencies.

- Limited visibility into the impact of design decisions on the overall system performance.

Advantages of System Composer:

- Model-based design and simulation capabilities to validate design choices early.

- Automatic code generation for efficient implementation, reducing manual effort and potential errors.

Project Test and Integration Phase:

- Integration, Test, Verification: Integrating subsystems, conducting tests, and verifying system behavior against requirements.

- System Verification and Validation: Ensuring that the system meets the specified requirements and performs its intended functions.

- Operation and Maintenance: Supporting the system throughout its operational life, including maintenance, upgrades, and support activities.

Traditional Challenges:

- Lack of traceability between requirements, tests, and system components.

- Manual and time-consuming testing and verification processes.

Advantages of System Composer:

- Seamless traceability between requirements, design, and verification artifacts.

- Automated test generation and execution, enabling faster and more efficient verification.

By incorporating modern tools like System Composer, system engineers can address the challenges associated with traditional systems engineering approaches. System Composer provides an integrated and collaborative environment for modeling, simulation, and verification, enabling a more efficient and effective implementation of the V-model.

Project: Designing a Mini-Drone System

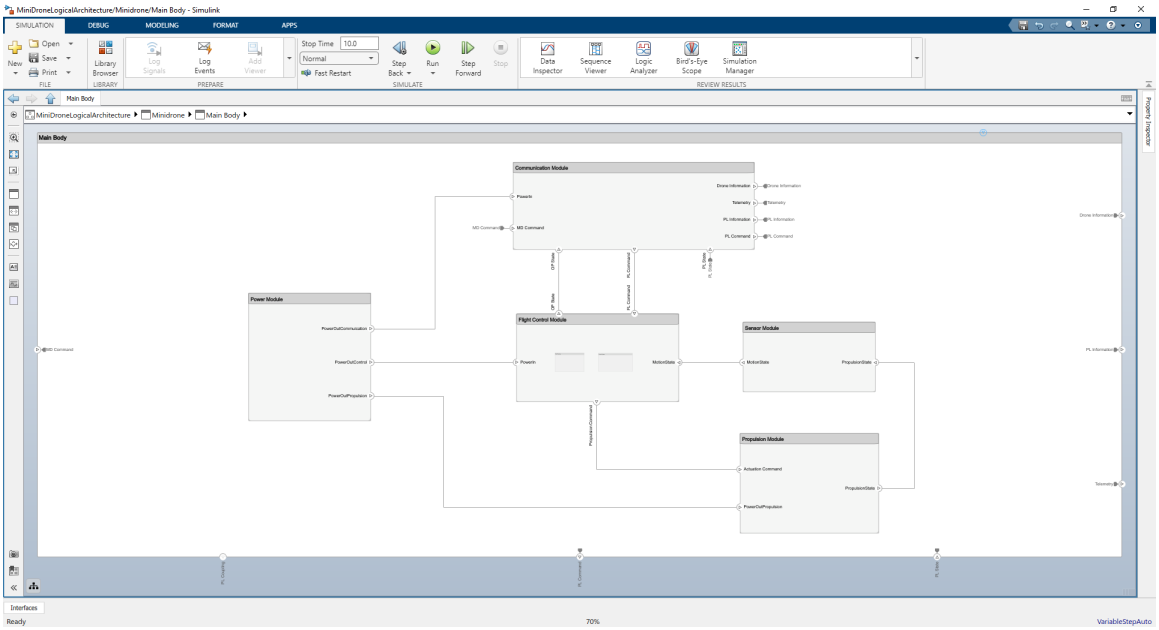
In recent years, mini-drones have become increasingly popular for a wide range of applications, including aerial photography, surveillance, and package delivery. As technology advances, the demand for autonomous mini-drones capable of performing complex tasks is growing. In this project, students will embark on the challenge of designing a comprehensive system for a mini-drone, perform controlled flights, carry a load, execute take-offs and landings, and establish communication with a ground base.

The design of a mini-drone system poses significant engineering challenges, requiring a systematic approach to system modeling and design. Students will need to integrate various components, such as sensors, actuators, algorithms, and communication modules, to enable the mini-drone to perceive its surroundings, make intelligent decisions, and execute precise maneuvers.

To tackle this multidimensional problem effectively, students will utilize System Composer, a powerful modeling and simulation tool provided by MathWorks. System Composer will empower students to visually represent the mini-drone's system architecture, including its components and their interactions. By leveraging the tool's capabilities, students can model the sensors for environment perception, actuators for flight control, load-carrying mechanisms, and communication modules for establishing a connection with the ground base.

System Composer will enable students to simulate and analyze the behavior of the mini-drone under various scenarios, ensuring that it meets the requirements of controlled flight, load-carrying capabilities, take-offs, and landings. They can iterate and refine their designs, considering factors such as real-time data processing, sensor fusion, trajectory planning, collision avoidance, and reliable communication protocols.

Below we can see an example of the system's schematic where the system composer divides the system into different engineering blocks. the aforementioned blocks, do not necessarliy posses a functionality, they may have Sterotypes and Requirements. Further in the course we will discuss what is meant by Sterotypes and Requiements.



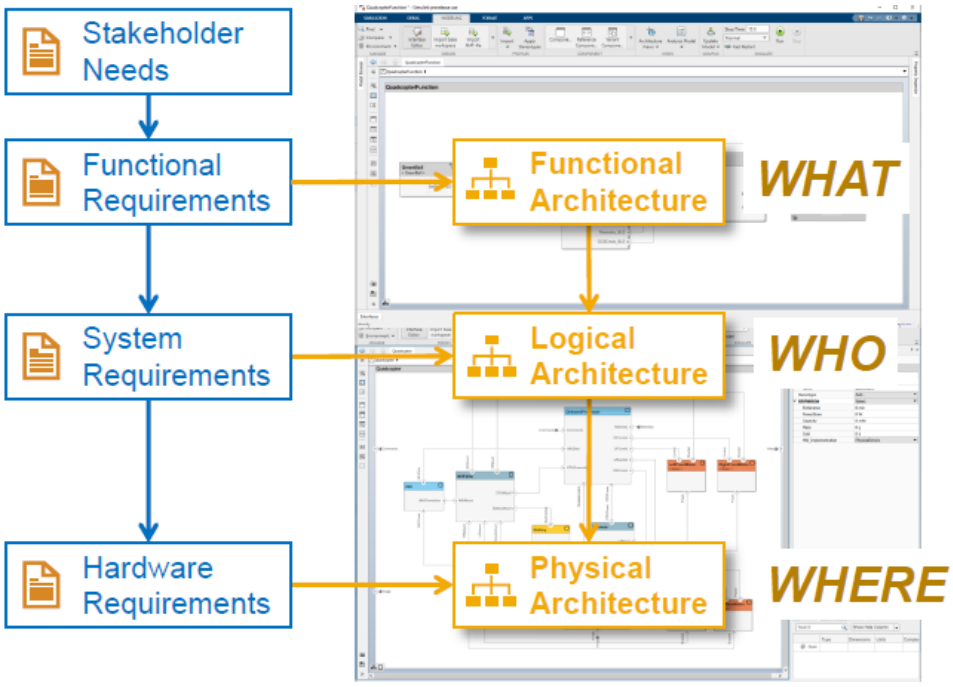
An added advantage is that we can add a running Simulink model to this system architecture to specicialize it with required functionalities.

System Composer Overview:

- Managing System Requirments.
- Constructing architectural models & Linking Architectures
- Applying meta data to architectural elements
- Stereotypes.
- Architecture Views and Analysis.

Typical System Architecture Workflows:

The main reason behind using the system composer lays under the understanding of the system's requirements, architecture, and the need to logically categorize them as a complete system. Below we can observe how are they categorized, and how are they related to each.



Furthermore, Model-Based Systems Engineering (MBSE), and Model-Based Design (MBD), are not the same concepts. MBD is what is built in Simulink, where MBSE is an engineering desciplin. Here comes the role of System Composer™ where it **bridges** the gap between both:

High level questions we need to answer:

- What components are required? How many of each? What is the all upmass? what components are significant in terms of weight? What is safety related? How reliable is the component? How do the components relate to system requirements?
- what functions must be supported? What are the dependencies between functions? What interfaces are required? In what language is the function design to be implemented? etc