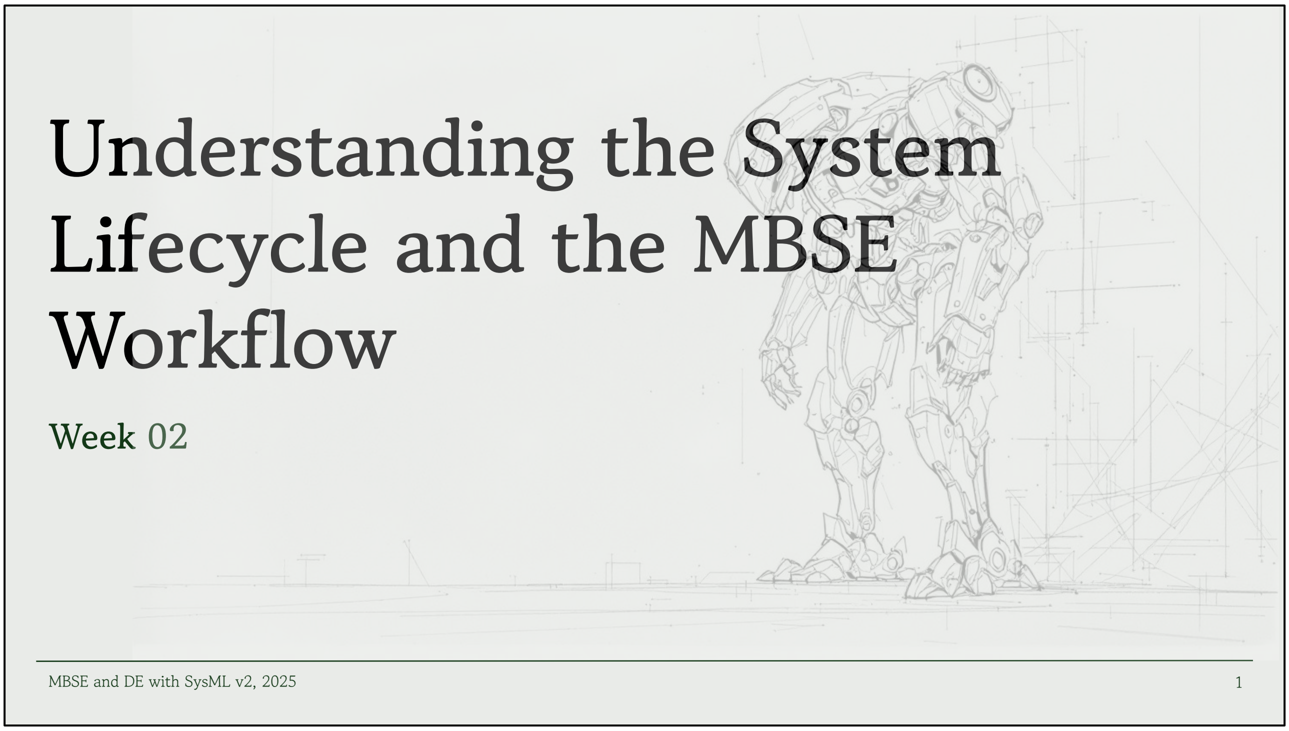
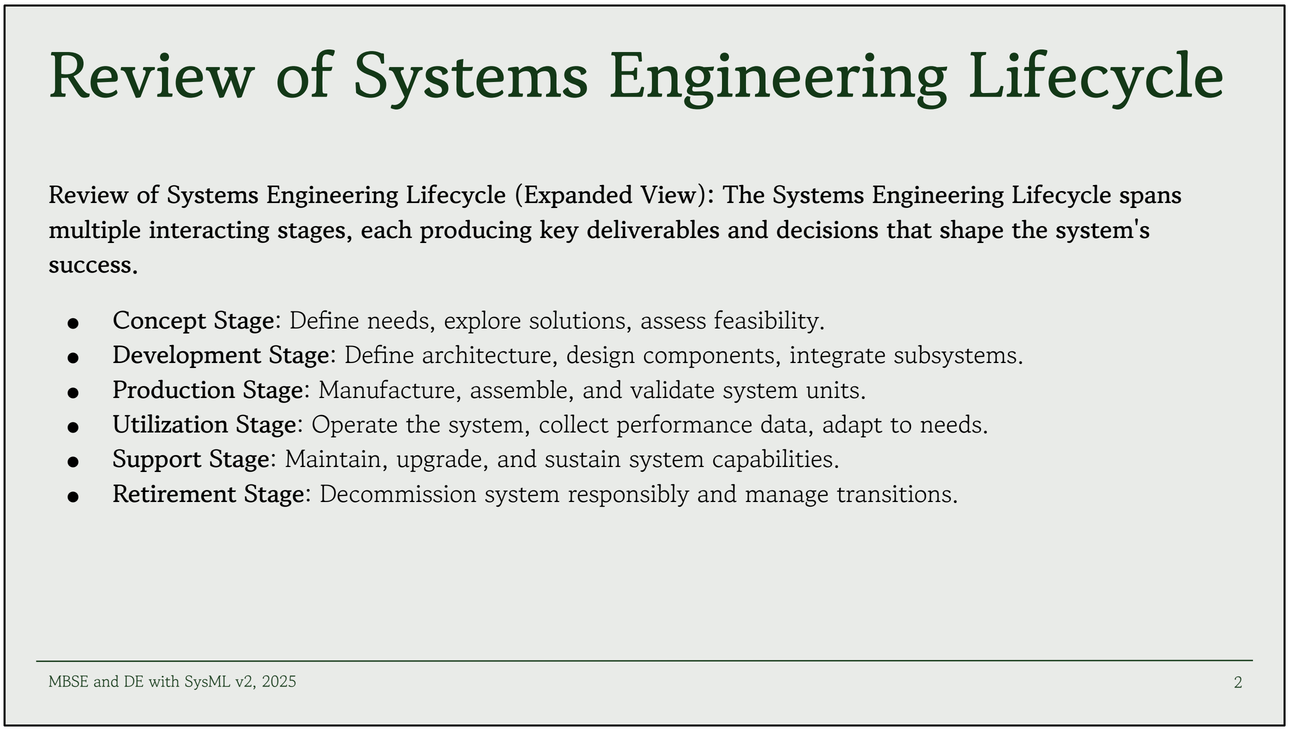


Understanding the System Lifecycle and the MBSE Workflow

Week 02



Review of Systems Engineering Lifecycle

Review of Systems Engineering Lifecycle (Expanded View): The Systems Engineering Lifecycle spans multiple interacting stages, each producing key deliverables and decisions that shape the system's success.

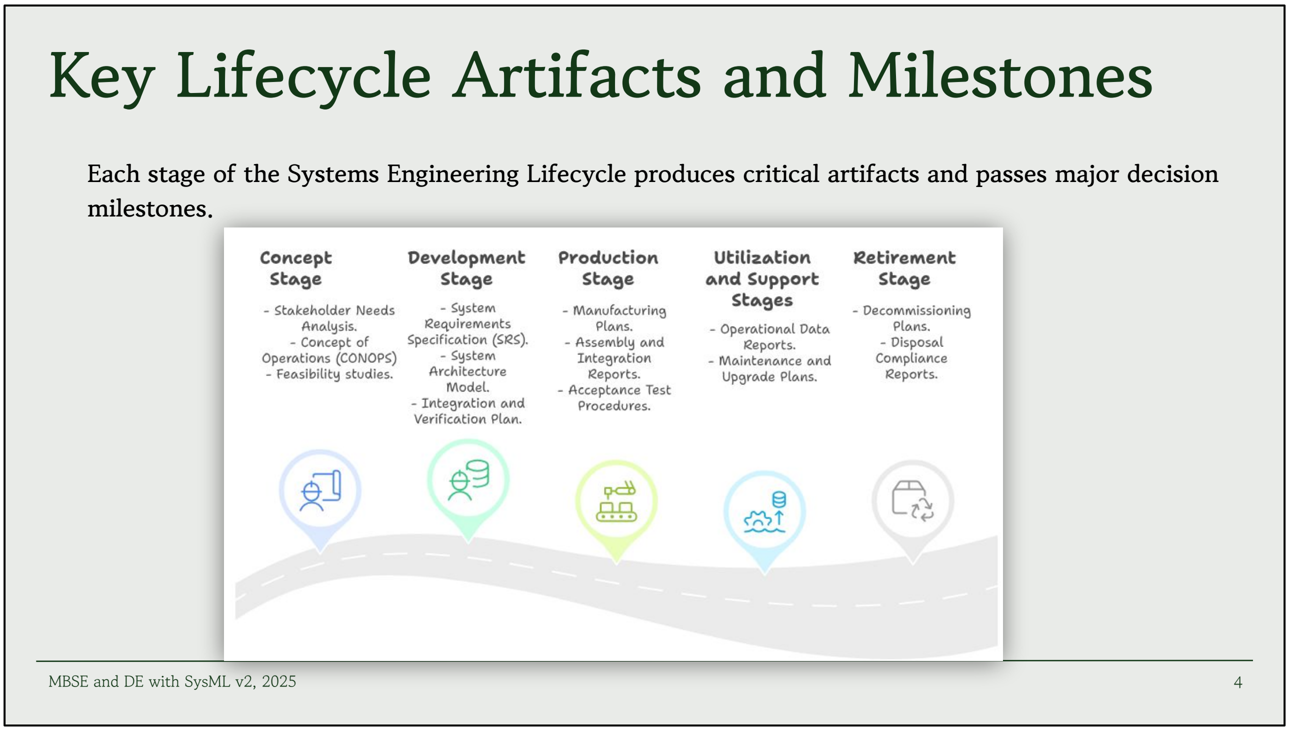
- **Concept Stage**: Define needs, explore solutions, assess feasibility.
- **Development Stage**: Define architecture, design components, integrate subsystems.
- **Production Stage**: Manufacture, assemble, and validate system units.
- **Utilization Stage**: Operate the system, collect performance data, adapt to needs.
- **Support Stage**: Maintain, upgrade, and sustain system capabilities.
- **Retirement Stage**: Decommission system responsibly and manage transitions.

As we begin Week 2, let's deepen our understanding of the **Systems Engineering Lifecycle**, which we first introduced briefly in Week 1.
The Systems Engineering Lifecycle is more than just a sequence of activities — It represents a structured way of thinking and working that ensures system success across decades of operation.
Let’s explore each stage in more detail:

- The **Concept Stage** is where it all begins:
  - We engage stakeholders, define needs, explore possible solutions, and assess technical, operational, and economic feasibility.
  - Outputs include concept of operations documents (CONOPS), preliminary system goals, and initial feasibility studies.
- The **Development Stage** turns concepts into formal system definitions:
  - Engineers define system architectures, allocate functions to subsystems, develop detailed designs, and plan integration strategies.
  - Verification and Validation planning also starts here — even before hardware exists.
- The **Production Stage** transforms designs into reality:
  - This includes manufacturing, assembling, and verifying system elements against design specifications.
  - Managing supply chains, quality assurance, and production test procedures are key focuses.
- The **Utilization Stage** begins once the system is delivered and operational:
  - Engineers monitor system performance, support users, and collect operational data to drive improvements.

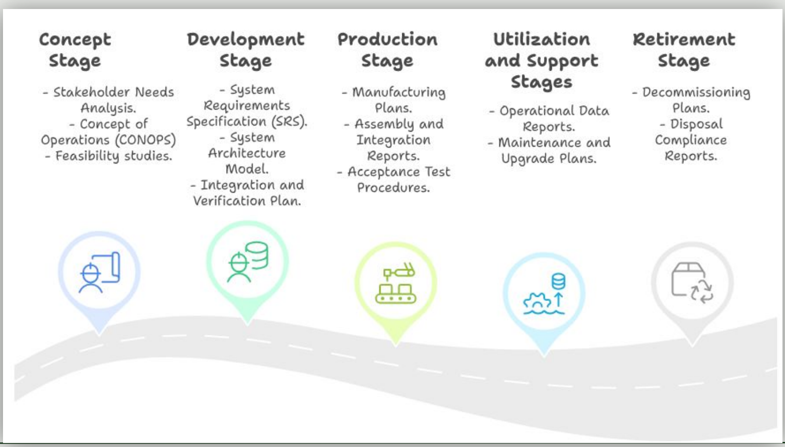
- The **Support Stage** overlaps with utilization:
    - Activities include maintenance, repair, system upgrades, and lifecycle extension planning.
    - Support ensures system availability, reliability, and cost-effectiveness across its lifespan.
- Finally, the **Retirement Stage** manages end-of-life transitions:
    - Systems are decommissioned, replaced, or repurposed.
    - Environmental, safety, legal, and strategic considerations must be handled responsibly.

Importantly, **these stages are not isolated**: Feedback from later stages often drives updates to earlier phases.
For example, operational lessons learned during Utilization may trigger retrofits or improvements during Support or even new design initiatives.
This lifecycle structure provides the backbone for Systems Engineering activities — and, as we'll soon see, **MBSE provides powerful support for managing these activities efficiently and traceably.**
Understanding this lifecycle deeply is critical for applying MBSE correctly and effectively.



Key Lifecycle Artifacts and Milestones

Each stage of the Systems Engineering Lifecycle produces critical artifacts and passes major decision milestones.


Concept Stage
- Stakeholder Needs Analysis.
- Concept of Operations (CONOPS)
- Feasibility studies.
Development Stage
- System Requirements Specification (SRS).
- System Architecture Model.
- Integration and Verification Plan.
Production Stage
- Manufacturing Plans.
- Assembly and Integration Reports.
- Acceptance Test Procedures.
Utilization and Support Stages
- Operational Data Reports.
- Maintenance and Upgrade Plans.
Retirement Stage
- Decommissioning Plans.
- Disposal Compliance Reports.

As we deepen our view of the Systems Engineering Lifecycle, it's critical to recognize that **each phase produces important deliverables — called artifacts — and passes major decision points — called milestones.** These artifacts and milestones are how we **control system development**, **communicate across teams**, and **manage risk** across the lifecycle.Let's walk through the stages:

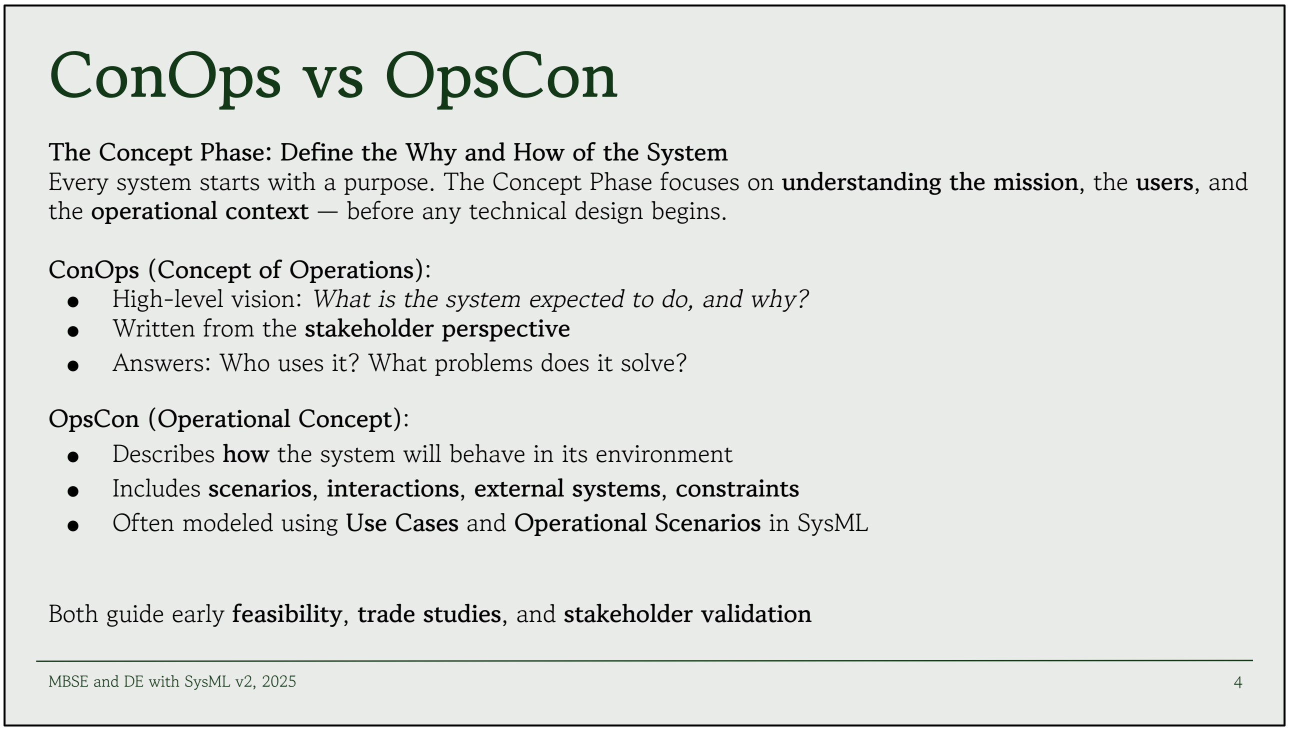
- During the **Concept Stage**, we produce:
  - A **Stakeholder Needs Analysis**: capturing what the users and other stakeholders require from the system.
  - A **Concept of Operations** (**CONOPS**): describing how the system is envisioned to be used operationally.
  - **Feasibility Studies**: assessing technical, operational, and economic viability.
- During the **Development Stage**, we produce:
  - A **System Requirements Specification** (**SRS**): a formal document defining what the system must do.
  - A **System Architecture Model**: often built using SysML (and in our course, SysML v2), showing the structure and behavior of the system.
  - An **Integration and Verification Plan**: mapping how we will assemble the system and prove that it meets its requirements.
- During the **Production Stage**, we produce:
  - **Manufacturing Plans**: defining how the system or its components will be built.
  - **Assembly and Integration Reports**: documenting how system parts are combined and checked.
  - **Acceptance Test Procedures**: defining tests that must be passed before delivery to customers.

- During the **Utilization and Support Stages**, we produce:
    - **Operational Data Reports**: gathering field performance data.
    - **Maintenance and Upgrade Plans**: ensuring system sustainability and continual performance.
- Finally, during the **Retirement Stage**, we produce:
    - **Decommissioning Plans**: describing how the system will be safely retired from service.
    - **Disposal Compliance Reports**: ensuring environmental, safety, and regulatory obligations are met.

At the end of each major stage, a **milestone review** typically happens: For example, Concept Review, System Requirements Review (SRR), Preliminary Design Review (PDR), Critical Design Review (CDR), and others.

These artifacts and milestones are the checkpoints:

- They ensure that the system is evolving correctly.
- They help leadership and stakeholders make go/no-go decisions.
- They enable risk management and lifecycle control.



ConOps vs OpsCon

The Concept Phase: Define the Why and How of the System
Every system starts with a purpose. The Concept Phase focuses on **understanding the mission**, the **users**, and the **operational context** — before any technical design begins.

ConOps **(Concept of Operations)**:

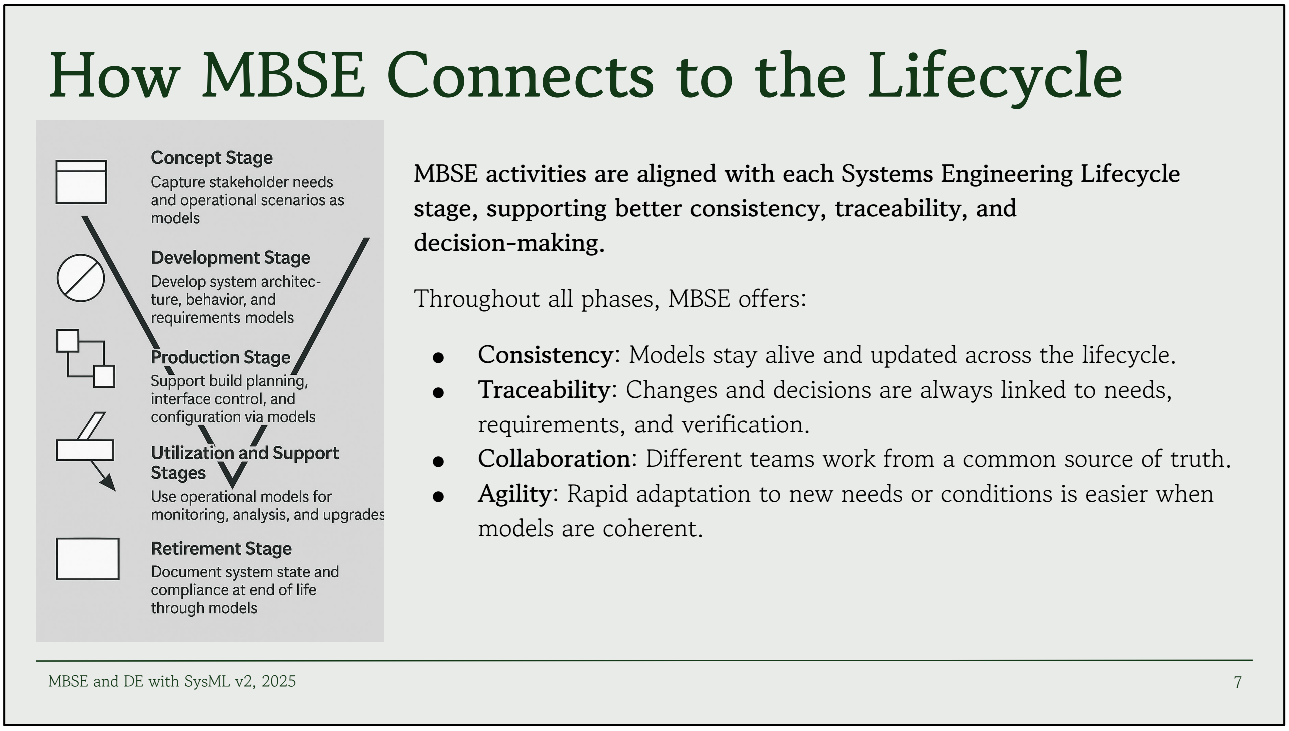
- High-level vision: *What is the system expected to do, and why?*
- Written from the **stakeholder perspective**
- Answers: Who uses it? What problems does it solve?

OpsCon **(Operational Concept)**:

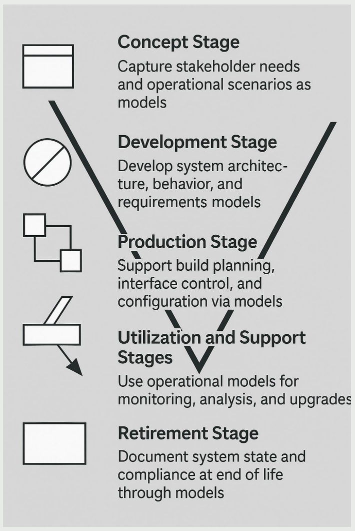
- Describes **how** the system will behave in its environment
- Includes **scenarios**, **interactions**, **external systems**, **constraints**
- Often modeled using **Use Cases** and **Operational Scenarios** in SysML

Both guide early **feasibility**, **trade studies**, and **stakeholder validation**

The Concept Phase is where we define **why** the system is needed and **how** it will be used — before touching architecture.
ConOps is a narrative — it explains the system's mission and value from the user's point of view.
OpsCon goes deeper — it describes operational conditions, key actors, environmental constraints, and usage scenarios.
In MBSE, we use models to formalize both: **Stakeholder needs**, **Use Cases**, and **Scenarios** become the digital expression of ConOps and OpsCon.
A well-defined Concept Phase prevents costly rework later. It grounds the system in real needs and prepares for successful design.



How MBSE Connects to the Lifecycle


Concept Stage
Capture stakeholder needs and operational scenarios as models
Development Stage
Develop system architec-ture, behavior, and requirements models
Production Stage
Support build planning, interface control, and configuration via models
Utilization and Support Stages
Use operational models for monitoring, analysis, and upgrades
Retirement Stage
Document system state and compliance at end of life through models

MBSE activities are aligned with each Systems Engineering Lifecycle stage, supporting better consistency, traceability, and decision-making.

Throughout all phases, MBSE offers:

- **Consistency**: Models stay alive and updated across the lifecycle.
- **Traceability**: Changes and decisions are always linked to needs, requirements, and verification.
- **Collaboration**: Different teams work from a common source of truth.
- **Agility**: Rapid adaptation to new needs or conditions is easier when models are coherent.

Now that we've reviewed the Systems Engineering Lifecycle and its artifacts, it's time to connect **how MBSE directly supports and aligns** with each stage of the lifecycle.
MBSE is not just a 'design tool' — it **traces across the entire lifecycle** from initial concept to final retirement, ensuring continuity, consistency, and data-driven decision-making.
Let's walk through each phase:

- During the **Concept Stage**,
  - MBSE helps capture **stakeholder needs** using needs models.
  - It models **operational scenarios** — describing how the system would behave and interact with users and environments.
  - These early models clarify what is needed and guide trade studies.
- During the **Development Stage**,
  - MBSE creates detailed **requirements models**, **architecture models**, **behavior models**, and **constraint models**.
  - Instead of disjointed documents, system structure and behavior are fully captured and traceable.
  - Verification and validation planning are embedded in the model early.
- During the **Production Stage**,
  - MBSE models support **build planning**, **interface control**, and **configuration management**.
  - Manufacturing and integration activities are connected to model-driven specifications.
- During the **Utilization and Support Stages**,
  - Operational models allow real-time system monitoring, anomaly analysis, and performance prediction.

    - Upgrades can be simulated in the model before they are implemented physically, reducing risks.
- Finally, during the **Retirement Stage**,
    - MBSE models provide critical documentation of system configuration, verification history, and compliance evidence.
    - Decommissioning can be planned and validated systematically.

Throughout all phases, MBSE offers:

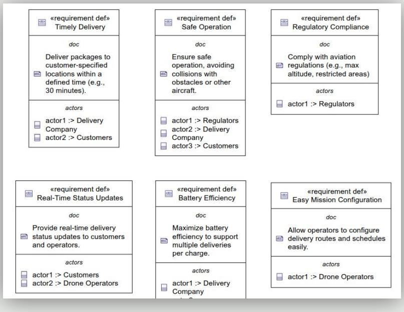
- **Consistency**: Models stay alive and updated across the lifecycle.
- **Traceability**: Changes and decisions are always linked to needs, requirements, and verification.
- **Collaboration**: Different teams work from a common source of truth.
- **Agility**: Rapid adaptation to new needs or conditions is easier when models are coherent.

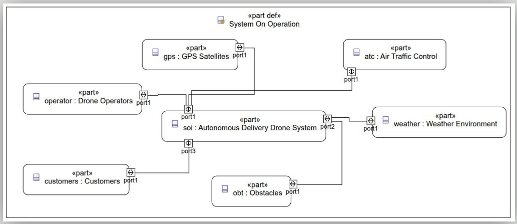
In short: **MBSE is not a side activity — it is integrated into every phase of the system lifecycle**, and is crucial for making Digital Engineering achievable in real practice.

MBSE in Concept Stage

During the Concept Stage, MBSE captures stakeholder needs, operational scenarios, and early system goals in structured, traceable models.

- Identify and model **stakeholders and their needs**.
- Define **operational use cases** and **mission scenarios**.
- Model early **system boundaries** and **interactions**.
- Support **trade studies** and **feasibility analysis** using models.
- Lay foundation for requirements refinement and system design.


«requirement def» Timely Delivery
doc
Deliver packages to customer-specified locations within a defined time (e.g., 30 minutes).
actors
actor1 :> Delivery Company
actor2 :> Customers
«requirement def» Safe Operation
doc
Ensure safe operation, avoiding collisions with obstacles or other aircraft.
actors
actor1 :> Regulators
actor2 :> Delivery Company
actor3 :> Customers
«requirement def» Regulatory Compliance
doc
Comply with aviation regulations (e.g., max altitude, restricted areas)
actors
actor1 :> Regulators
«requirement def» Real-Time Status Updates
doc
Provide real-time delivery status updates to customers and operators.
actors
actor1 :> Customers
actor2 :> Drone Operators
«requirement def» Battery Efficiency
doc
Maximize battery efficiency to support multiple deliveries per charge.
actors
actor1 :> Delivery Company
«requirement def» Easy Mission Configuration
doc
Allow operators to configure delivery routes and schedules easily.
actors
actor1 :> Drone Operators


«part def» System On Operation
«part» gps : GPS Satellites
port1
«part» atc : Air Traffic Control
port1
«part» operator : Drone Operators
port1
port1
«part» soi : Autonomous Delivery Drone System
port2
port3
port1
«part» weather : Weather Environment
«part» customers : Customers
port1
«part» obt : Obstacles
port1

Let's now look specifically at **how MBSE supports the Concept Stage** — the very beginning of the Systems Engineering Lifecycle.
In the Concept Stage, the main goals are to:

- Understand what the stakeholders truly need.
- Explore what the system might do and how it might be used.
- Assess feasibility before committing large resources to development.

MBSE plays a critical role here by allowing engineers to capture these early ideas **in structured, analyzable models** — rather than relying only on text documents or assumptions.
First, MBSE helps us **identify and model stakeholders and their needs**:

- Using models like stakeholder diagrams, needs models, and concern models,
- We ensure all key users and influencers are considered — not just the obvious ones.

Second, MBSE enables us to **define operational use cases and mission scenarios**:

- What sequences of actions will the system support?
- What environments and constraints must it operate within?
- These behaviors are captured early as **scenarios and interaction models**.

Third, MBSE models help **define early system boundaries and external interfaces**:

- What is 'inside' the system versus 'outside'?
- What actors (users, other systems, environment factors) does the system interact with?
- This helps avoid critical errors later when architecture decisions are made.

Fourth, MBSE models support **trade studies and feasibility analysis**:

- By modeling different concepts quickly, engineers can evaluate options for feasibility, cost, risk, and performance.
- Models can guide early simulation to compare concepts quantitatively.

Finally, MBSE lays the **foundation for formal requirements development**:

- Needs and scenarios captured during the Concept Stage flow naturally into detailed requirements models during Development Stage.

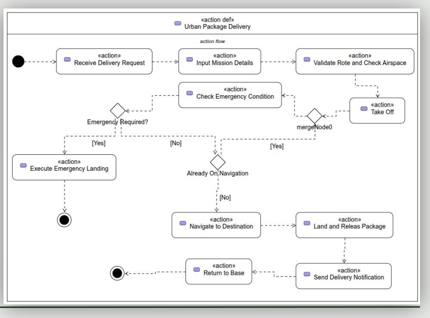
In short: **MBSE in the Concept Stage creates clarity early** —

- Reducing the risk of building the wrong system,
- Improving decision-making speed,
- And ensuring that stakeholder needs are central from Day 1.

MBSE in Development Stage

During the Development Stage, MBSE defines system requirements, architecture, behavior, constraints, and verification models — forming the authoritative system description.

- Refine stakeholder needs into **formal system requirements**.
- Model **system architecture** (structure, interfaces, allocations).
- Model **system behavior** (use cases, activities, state transitions).
- Define **constraints and parameters** for engineering trade-offs.
- Plan and model **verification and validation** strategies.


«action def»
Urban Package Delivery
action flow
«action» Receive Delivery Request
«action» Input Mission Details
«action» Validate Rote and Check Airspace
«action» Check Emergency Condition
«action» Take Off
Emergency Required?
mergeNode0
[Yes]
[No]
[Yes]
«action» Execute Emergency Landing
Already On Navigation
[No]
«action» Navigate to Destination
«action» Land and Releas Package
«action» Return to Base
«action» Send Delivery Notification

Now let's see how MBSE supports the **Development Stage** — where system concepts are refined into formal definitions ready for production and testing.
The Development Stage transforms early ideas into **detailed, integrated system models** that define **what** the system must do, **how** it will be built, and **how** we will prove it works. MBSE plays a central role here:
First, MBSE allows engineers to **refine stakeholder needs into formal system requirements**:

- Needs and concerns captured during Concept Stage are translated into detailed, testable requirements.
- Requirements are connected directly to stakeholder concerns, system functions, and design elements inside the model.

Second, MBSE enables **modeling of system architecture**:

- We define **what parts the system is composed of** (blocks, parts, components).
- We define **how these parts interact** (interfaces, connections, protocols).
- We model **allocations** — mapping functions to physical or logical components.

Third, MBSE models **system behavior**:

- Use case views describe user-system interactions.
- Activity views describe internal workflows.
- State machines describe operational modes and transitions.
- Interaction models describe message flows between components.

Fourth, MBSE allows us to **define engineering constraints and parameters**:

- Performance limits, safety margins, environmental tolerances —
- These constraints are modeled explicitly, allowing trade studies, simulations, and optimization analyses.

Fifth, MBSE supports early planning for **Verification and Validation** (V&V):

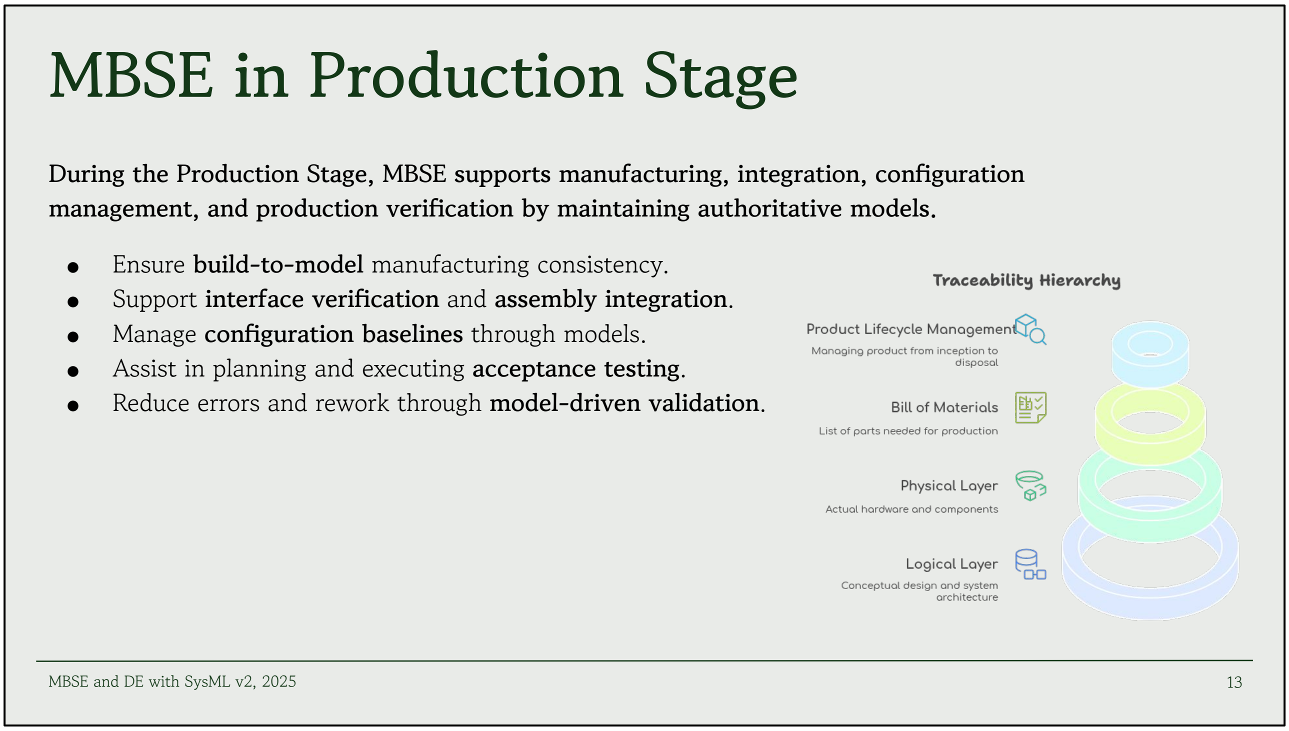
- Test cases, success criteria, and verification methods are modeled early.
- Traceability ensures that every requirement has a planned method for verification.

Importantly, MBSE ensures that **requirements, structure, behavior, constraints, and V&V plans are interconnected**:

- Changes in one part ripple consistently across the model.
- Impact analysis becomes structured and reliable.

By the end of the Development Stage, the MBSE model serves as the **single authoritative source of truth** for what the system is, what it must do, and how it will be tested.

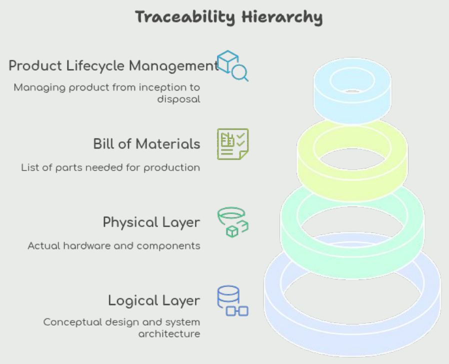
In short: **MBSE transforms Development from a document-driven guessing game into a precise, traceable, model-driven engineering discipline.**



MBSE in Production Stage

During the Production Stage, MBSE supports manufacturing, integration, configuration management, and production verification by maintaining authoritative models.

- Ensure **build-to-model** manufacturing consistency.
- Support **interface verification** and **assembly integration**.
- Manage **configuration baselines** through models.
- Assist in planning and executing **acceptance testing**.
- Reduce errors and rework through **model-driven validation**.


Traceability Hierarchy
Product Lifecycle Management
Managing product from inception to disposal
Bill of Materials
List of parts needed for production
Physical Layer
Actual hardware and components
Logical Layer
Conceptual design and system architecture

Now that the system design is complete, we move into the **Production Stage** — where physical components are manufactured, assembled, and verified.
MBSE continues to play a crucial role here, even after development models are 'finished.' Let's see how:
First, MBSE ensures **build-to-model manufacturing consistency**:

- Instead of relying solely on documents, manufacturing processes are often directly based on the verified architecture, interface, and component models.
- Tolerances, configurations, and assembly instructions can be extracted from the authoritative system model.

Second, MBSE supports **interface verification and assembly integration**:

- Interface models ensure that parts fit together physically, electrically, thermally, or logically.
- Any mismatches can often be detected early through model-based simulation or constraint checking before physical integration.

Third, MBSE helps manage **configuration baselines**:

- The model becomes a reference for what versions of parts, interfaces, and functions are being produced and assembled.
- This supports quality control, version management, and traceability.

Fourth, MBSE assists in **planning and executing acceptance testing**:

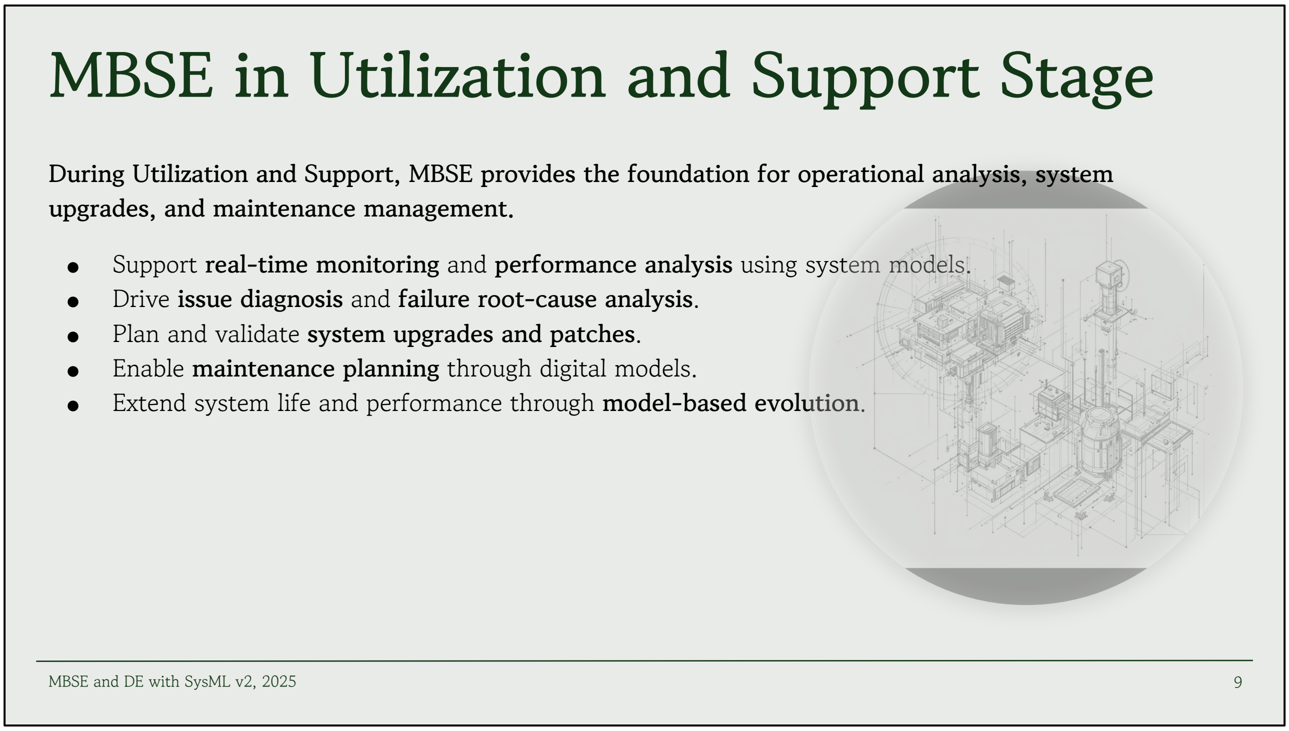
- Acceptance Test Procedures (ATP) can be derived from model-based verification criteria.
- Test cases are linked to system requirements and structures through traceable model elements.

Fifth, MBSE helps **reduce errors and rework**:

- By validating assembly paths, component interactions, and interface assumptions before actual production,
- MBSE minimizes costly surprises, late-stage modifications, and integration failures.

Ultimately, the model-driven production approach saves time, improves quality, and ensures that what is built in the real world matches the system defined and validated in the virtual world.

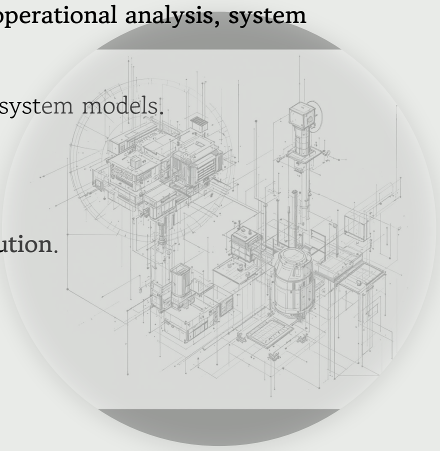
In short: **MBSE transforms Production from a reactive build-test-fix cycle into a proactive build-right-first-time process.** It bridges the gap between Engineering and Manufacturing in a digital, traceable, and integrated way.



MBSE in Utilization and Support Stage

During Utilization and Support, MBSE provides the foundation for operational analysis, system upgrades, and maintenance management.

- Support **real-time monitoring** and **performance analysis** using system models.
- Drive **issue diagnosis** and **failure root-cause analysis**.
- Plan and validate **system upgrades and patches**.
- Enable **maintenance planning** through digital models.
- Extend system life and performance through **model-based evolution**.



Even after the system has been manufactured, delivered, and deployed, **MBSE continues to add value during Utilization and Support stages**.
Here's how MBSE helps during real-world operations:
First, MBSE supports **real-time monitoring and performance analysis**:

- If the system is connected to a Digital Twin (or a partial model), real-world data can be compared against model predictions.
- Trends, anomalies, and degradations can be detected earlier, preventing failures and improving operations.

Second, MBSE drives **issue diagnosis and failure analysis**:

- When problems occur in operation, having a precise system model helps engineers trace symptoms back to causes.
- Interfaces, configurations, and behaviors can be reviewed and simulated to diagnose root causes faster and more accurately.

Third, MBSE enables **planning and validating system upgrades**:

- When new mission needs, technology updates, or safety improvements arise,
- Proposed changes can first be evaluated in the model before touching the real system — reducing operational disruption and risk.

Fourth, MBSE supports **maintenance planning and optimization**:

- Understanding how components degrade, interact, and fail helps engineers optimize preventive maintenance schedules.
- Reliability-centered maintenance planning can be based on model-derived criticality and impact analysis.

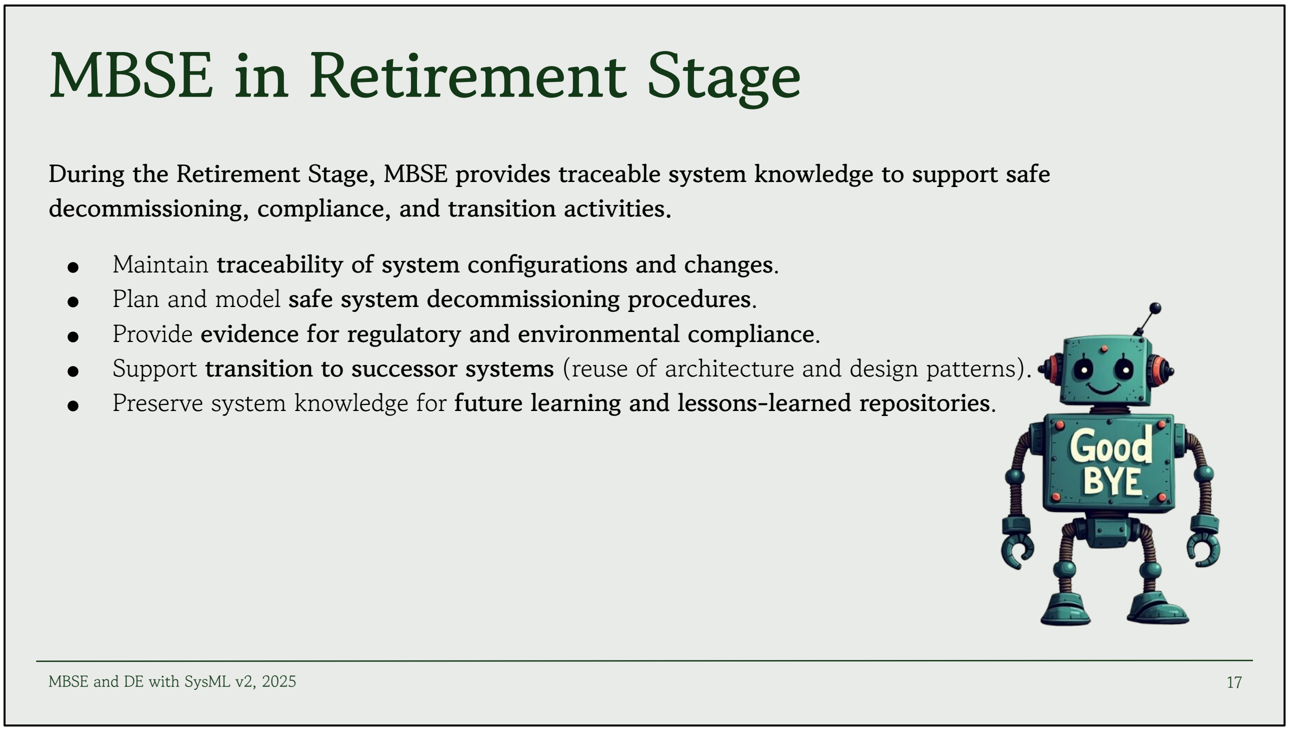
Fifth, MBSE helps **extend system life and performance**:

- Through model-based what-if analysis, systems can be adapted, evolved, or repurposed

- over time —
- Instead of being replaced prematurely, systems can deliver value much longer at lower cost.

Importantly, models created and maintained across the lifecycle allow engineers and operators to move from **reactive repair** to **proactive system management**.

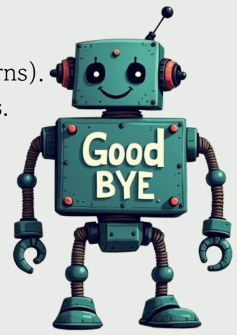
In short: **MBSE transforms Utilization and Support from maintenance guesswork into data-driven, model-supported system optimization. I**t closes the loop between design intent and operational reality — a cornerstone of true Digital Engineering.



MBSE in Retirement Stage

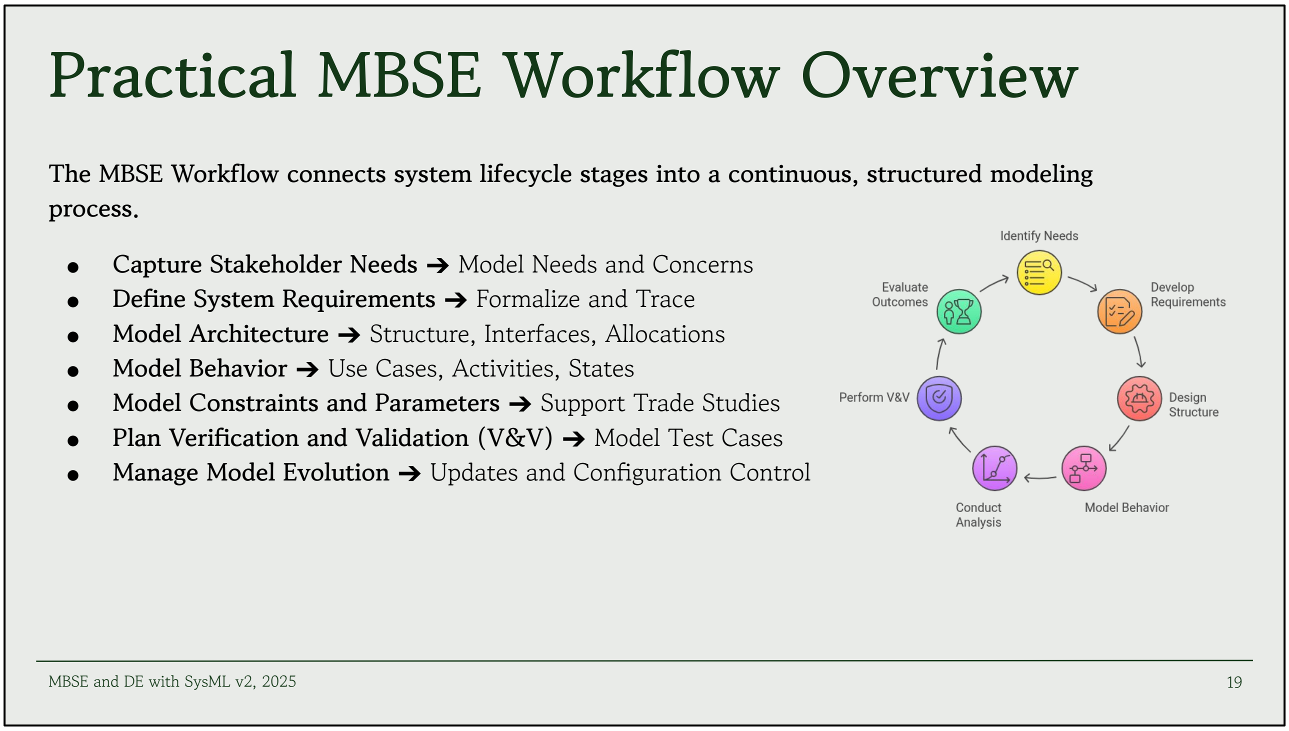
During the Retirement Stage, MBSE provides traceable system knowledge to support safe decommissioning, compliance, and transition activities.

- Maintain **traceability of system configurations and changes**.
- Plan and model **safe system decommissioning procedures**.
- Provide **evidence for regulatory and environmental compliance**.
- Support **transition to successor systems** (reuse of architecture and design patterns).
- Preserve system knowledge for **future learning and lessons-learned repositories**.



Even at the final phase of the system lifecycle — the **Retirement Stage** — **MBSE continues to deliver value**.
Retiring a system is not simply 'turning it off' — It involves careful, often legally and environmentally regulated activities to ensure safety, security, and responsible management of resources. Here's how MBSE supports Retirement:
First, MBSE models maintain **traceability of system configurations and changes**: Throughout the system's life, configurations evolve. Knowing exactly what is in place at end-of-life — based on authoritative models — is crucial for safe shutdown, disassembly, or transition.
Second, MBSE supports **planning and modeling decommissioning procedures**: Engineers can model safe deactivation sequences, resource recovery steps, and hazard mitigation strategies. These activities can be simulated and analyzed before execution.
Third, MBSE helps provide **evidence for compliance**: Many systems (especially in aerospace, defense, nuclear, and infrastructure sectors) must comply with environmental, security, and legal retirement regulations. MBSE models can demonstrate that all required conditions have been met and documented properly.
Fourth, MBSE facilitates **transition to successor systems**: Retirement often coincides with the deployment of new, replacement systems. Architecture patterns, interface definitions, and operational lessons captured in the existing models can accelerate and de-risk the development of next-generation systems.
Finally, MBSE preserves **system knowledge for organizational learning**: Models act as a structured knowledge base. They enable future teams to understand past systems' successes, challenges, and best practices — fueling continuous engineering improvement.
In short: **MBSE transforms system retirement from a chaotic shutdown into a responsible, traceable, and knowledge-preserving process.** It ensures that even at the end of a system's

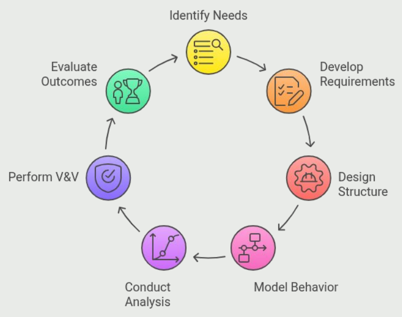
life, engineering excellence is maintained.



Practical MBSE Workflow Overview

The MBSE Workflow connects system lifecycle stages into a continuous, structured modeling process.

- **Capture Stakeholder Needs ➔** Model Needs and Concerns
- **Define System Requirements ➔** Formalize and Trace
- **Model Architecture ➔** Structure, Interfaces, Allocations
- **Model Behavior ➔** Use Cases, Activities, States
- **Model Constraints and Parameters ➔** Support Trade Studies
- **Plan Verification and Validation (V&V) ➔** Model Test Cases
- **Manage Model Evolution ➔** Updates and Configuration Control



Now that we've walked through how MBSE supports each stage of the Systems Engineering Lifecycle, let's talk about what an **actual practical MBSE workflow** looks like on a day-to-day basis.

MBSE is not just 'draw diagrams.' It is a **structured engineering workflow**, aligned with the system lifecycle, executed through modeling activities.

The basic MBSE workflow looks like this:

First, we **capture stakeholder needs**:

- Instead of listing needs in a Word document, we model them.
- Needs are connected to stakeholders, concerns, and initial operational scenarios.

Second, we **define system requirements**:

- We formalize what the system must do, how well it must do it, and under what conditions.
- Requirements are structured, traceable, and validated inside the model.

Third, we **model system architecture**:

- Structure: Parts, subsystems, and their compositions.
- Interfaces: Physical, logical, and communication links between parts.
- Allocations: Which subsystem satisfies which requirement or function.

Fourth, we **model system behavior**:

- Use Case Models: How the system interacts with users or other systems.
- Activity View: How the system internally functions.
- State Machines: How the system transitions between modes and handles events.

Fifth, we **model constraints and parameters**:

- Physical constraints (weight limits, endurance, noise limits).
- Logical constraints (data rates, failure probabilities).

- This supports trade studies and performance optimization.

Sixth, we **plan and model verification and validation** (V&V):

- Instead of writing test plans separately,
- We model Test Cases, Verification Criteria, and link them directly to Requirements and Functions.

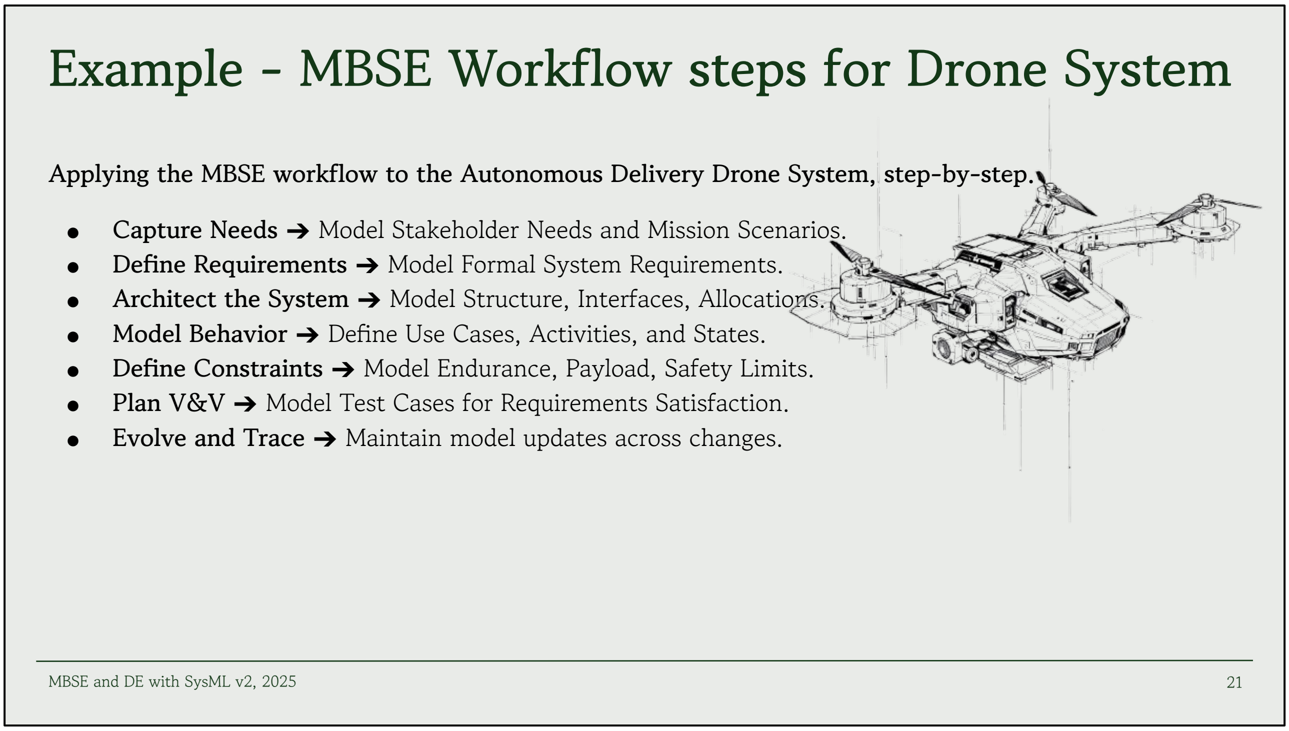
Finally, we **manage model evolution**:

- Requirements change. Architecture evolves. Behaviors get refined.
- MBSE tools allow configuration management, version control, and traceability updates to maintain model integrity.

This modeling cycle is **iterative** and **evolves** as the system matures from concept to production to retirement.

In short: MBSE **connects needs to requirements to design to behavior to testing — in one coherent, living model.** It replaces siloed document work with **an integrated, engineering-driven digital process.**

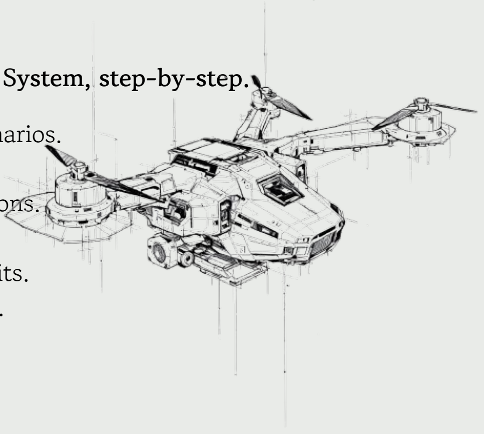
You will experience this workflow yourself in the course project — building your Autonomous Drone System model stage-by-stage using this exact MBSE workflow.



Example - MBSE Workflow steps for Drone System

Applying the MBSE workflow to the Autonomous Delivery Drone System, step-by-step.

- **Capture Needs ➔** Model Stakeholder Needs and Mission Scenarios.
- **Define Requirements ➔** Model Formal System Requirements.
- **Architect the System ➔** Model Structure, Interfaces, Allocations.
- **Model Behavior ➔** Define Use Cases, Activities, and States.
- **Define Constraints ➔** Model Endurance, Payload, Safety Limits.
- **Plan V&V ➔** Model Test Cases for Requirements Satisfaction.
- **Evolve and Trace ➔** Maintain model updates across changes.



Let's now ground the MBSE workflow we just introduced into a **real application**: how you would apply MBSE step-by-step to **our Autonomous Delivery Drone System**.
This mirrors exactly what you will be doing for your course-long project.
Here's how the MBSE workflow looks practically:
Capture Needs We identify our key stakeholders — Customers, Operators, Maintenance Teams, Regulatory Bodies. We model their needs:

- Deliver packages within 5 km.
- Comply with regulations.
- Keep noise under 60 dB.

Define Requirements We formalize these needs into System Requirements. Example: "The drone shall autonomously deliver packages within a 5 km radius without human intervention."
Architect the System We define the system structure using Element: Drone System composed of Flight Controller, Sensor Suite, Battery System, Delivery Mechanism, Communication Module. We define Interfaces: Power, Data, Command links between components.
Model Behavior We model Mission Scenarios: Start Mission ➔ Navigate ➔ Deliver ➔ Return ➔ Land. We create Use Case Views, Activity Views, and State Machines.
Define Constraints We model engineering parameters:

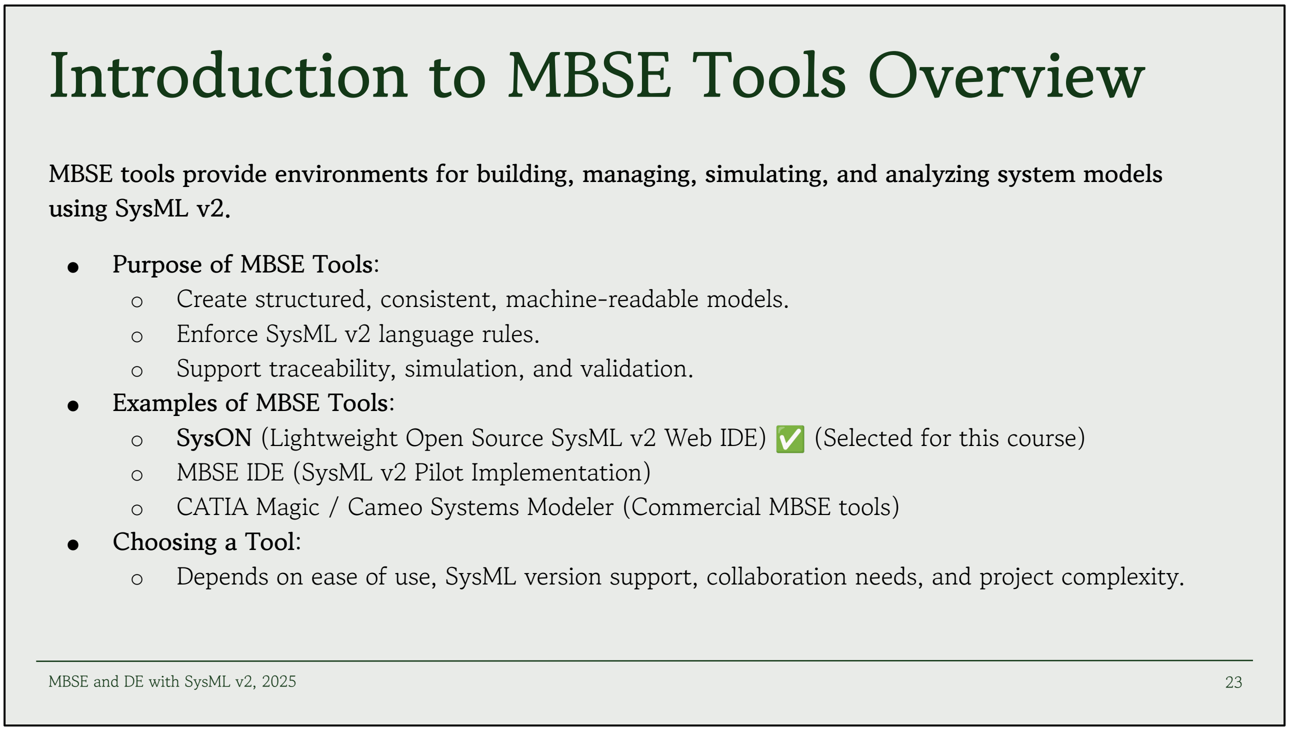
- Maximum payload 2 kg.
- Minimum flight endurance 40 minutes.

Plan Verification and Validation (V&V) We model Test Cases linked to Requirements:

- Test maximum flight range.
- Test noise emission levels.
- Test obstacle avoidance performance.

Evolve and Trace the Model As new needs arise or operational experience is gained, We

update Requirements, Architecture, Behavior, and V&V models — keeping full traceability.
Throughout these steps, the model remains **the single source of truth** — supporting analysis, simulation, review, communication, and decision-making.
In short: By following this MBSE workflow, you turn early ideas into structured, testable, traceable system definitions — ready for production, deployment, operation, and beyond. And you will practice these exact steps week-by-week as you build your own drone system model!"



Introduction to MBSE Tools Overview

MBSE tools provide environments for building, managing, simulating, and analyzing system models using SysML v2.

- Purpose of MBSE Tools:
  - Create structured, consistent, machine-readable models.
  - Enforce SysML v2 language rules.
  - Support traceability, simulation, and validation.
- Examples of MBSE Tools:
  - **SysON** (Lightweight Open Source SysML v2 Web IDE) ✅ (Selected for this course)
  - MBSE IDE (SysML v2 Pilot Implementation)
  - CATIA Magic / Cameo Systems Modeler (Commercial MBSE tools)
- Choosing a Tool:
  - Depends on ease of use, SysML version support, collaboration needs, and project complexity.

Now that we understand the MBSE workflow, let's introduce the **tools** that enable MBSE in practice.
MBSE tools are much more than just drawing tools — they provide **structured, standards-compliant environments** where systems are modeled consistently, validated, and evolved. Their main purposes are:

- To create machine-readable, traceable SysML v2 models.
- To ensure correct modeling syntax and semantics.
- To enable analysis, collaboration, and simulation across the system lifecycle.

Examples of modern MBSE tools include:

SysON (Selected for this course):
- A lightweight, open-source web-based modeling environment for SysML v2.
- Extremely easy to install and run via Docker.
- Provides project creation, model editing, and simple visualization fully within a web browser.
- Perfect for fast learning and classroom deployment.

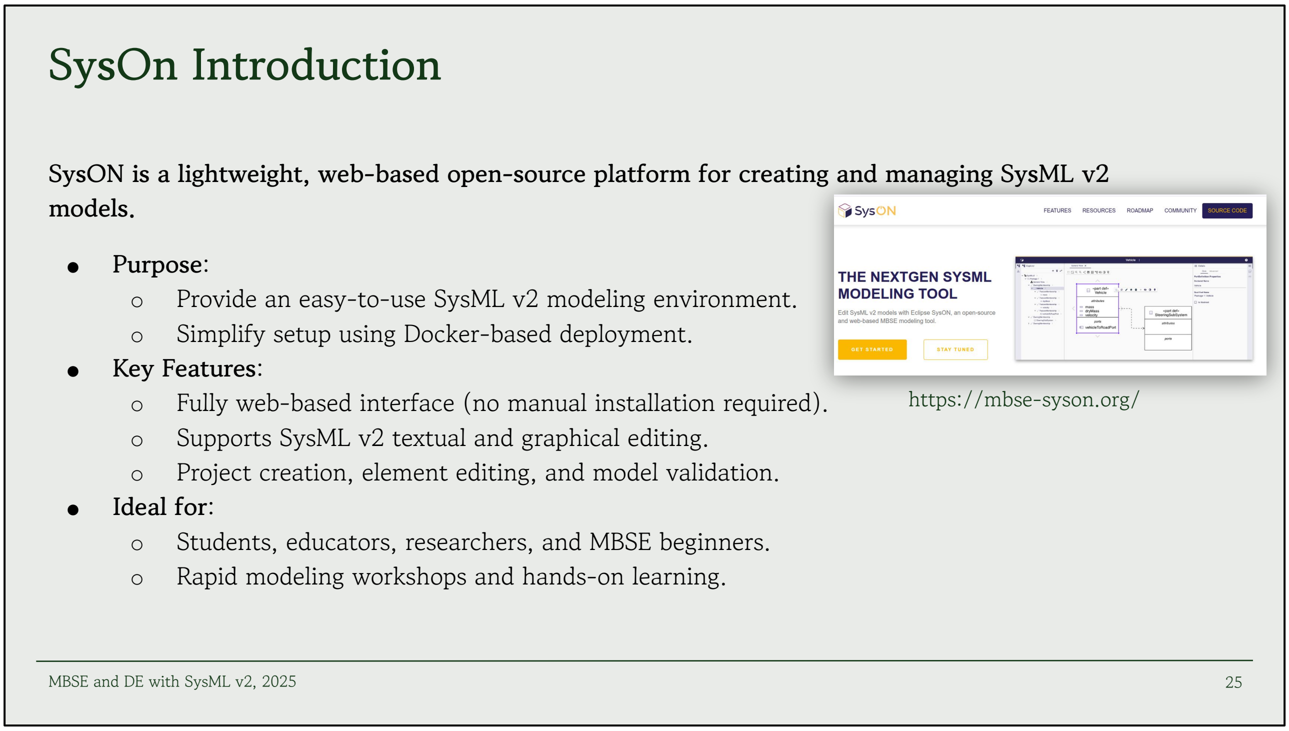
MBSE IDE (SysML v2 Pilot Implementation):
- An open-source Eclipse-based project aimed at piloting the official SysML v2 standard.
- Powerful but requires more complex Java and Eclipse setup, better suited for experimental users.

CATIA Magic / Cameo Systems Modeler:
- Commercial MBSE tools with rich SysML v1.x support (and emerging SysML v2 features).
- Used in industry for large-scale, simulation-integrated projects.

In this course, we will focus on using **SysON** because:

- It provides a **simple, straightforward modeling experience**.
- It is **easy to deploy** with minimal installation (Docker + YAML file).
- It supports **SysML v2 syntax and concepts** clearly without heavy tool complexity.
- It allows **everyone to be up and running quickly** — saving time for actual modeling practice.

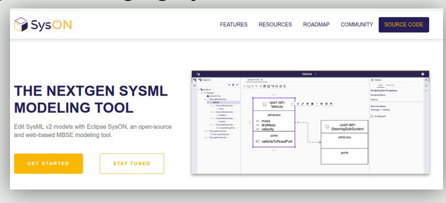
Important note: While MBSE IDE and CATIA Magic are important to know about for future professional work, for our learning purposes, **SysON gives the best balance of simplicity, accessibility, and alignment with SysML v2.**



SysOn Introduction

SysON is a lightweight, web-based open-source platform for creating and managing SysML v2 models.

- Purpose:
  - Provide an easy-to-use SysML v2 modeling environment.
  - Simplify setup using Docker-based deployment.
- Key Features:
  - Fully web-based interface (no manual installation required).
  - Supports SysML v2 textual and graphical editing.
  - Project creation, element editing, and model validation.
- Ideal for:
  - Students, educators, researchers, and MBSE beginners.
  - Rapid modeling workshops and hands-on learning.



https://mbse-syson.org/

Now let's introduce the specific tool we will use throughout this course: **SysON** — an open-source, web-based environment for practicing **SysML v2 modeling**.
SysON was designed to:

- Simplify the adoption of SysML v2 by offering a **modern, lightweight platform**.
- Remove the heavy technical barriers often associated with installing and configuring modeling environments.

Its primary purposes are:

- To allow users to easily **create, manage, and visualize SysML v2 models** through an intuitive web interface.
- To support **both textual and graphical modeling** — allowing users to see the structure of the system clearly.

Key Features of SysON include:

Web-Based Access:

- Once deployed via Docker, you can access SysON from your browser — no separate installation steps for each student.
- This also makes it easy to reset, update, or manage the tool centrally if needed.

Project Creation and Management:

- Create multiple system projects.
- Add Needs, Requirements, Blocks, Behaviors, and more easily.

SysML v2 Compliant Modeling:

- SysON models follow the official SysML v2 standard structures.
- Syntax highlighting and validation tools help ensure model correctness.

Simple Deployment Using Docker:

    - With a few simple commands, SysON can be pulled and launched on any machine with Docker installed.
    - This minimizes installation time and maximizes classroom modeling time.

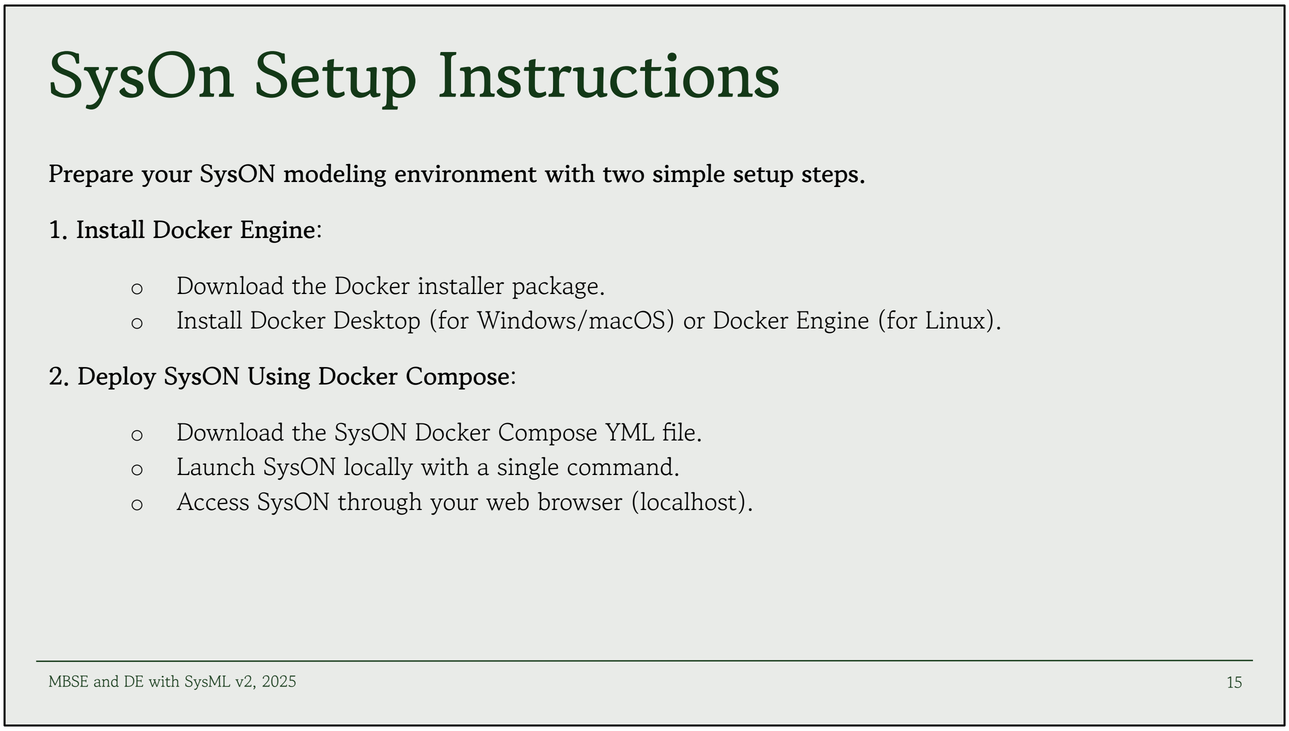
SysON is ideal for:

- Students new to MBSE and SysML v2.
- Educators setting up modeling workshops.
- Researchers prototyping early system models.

In this course, we will use SysON because:

- It offers **fast, consistent setup for everyone**.
- It supports **standards-compliant SysML v2 modeling** needed for professional MBSE practice.
- It is **simple, lightweight, and effective** — focusing our energy on learning modeling, not fighting installations.

By mastering SysON, you will gain the skills needed to build, manage, and validate real SysML v2 system models — a critical step toward becoming a modern Systems Engineer.



SysOn Setup Instructions

Prepare your SysON modeling environment with two simple setup steps.

1. Install Docker Engine:

- Download the Docker installer package.
- Install Docker Desktop (for Windows/macOS) or Docker Engine (for Linux).

2. Deploy SysON Using Docker Compose:

- Download the SysON Docker Compose YML file.
- Launch SysON locally with a single command.
- Access SysON through your web browser (localhost).

Now that we have chosen **SysON** as our modeling tool, let's go through **how we will set up the environment** — quickly and consistently for everyone. Its required you two installations artifacts:

- The Docker installer for your operating system.
- The SysON Docker Compose YML file ready to deploy.

Step 1: Install Docker Engine

- Download the Docker installer.
- If you are on Windows or macOS, install **Docker Desktop**.
- If you are on Linux, install **Docker Engine** manually (we will assist you if needed).
- After installation, verify Docker is working by running `docker --version` from your terminal or command prompt.

Step 2: Deploy SysON Using Docker Compose

- Get **Docker Compose YML file** (download from: https://doc.mbse-syson.org/syson/v2024.11.0/installation-guide/how-tos/install/local_test.html).
- Place the YML file into a new folder on your laptop. docker-compose.yml
- Open a terminal inside that folder and run the command: `docker compose up`
- Docker will launch SysON as a local container.
- After a few seconds, you can open your web browser and access SysON at: http://localhost:8000

Summary of Week 2

This week, we take a step forward to deepen our understanding of MBSE and its relationship to the Systems Engineering Lifecycle. We will also cover an overview and setup of SysML v2, which we will use throughout this course.

- Understand the **full Systems Engineering Lifecycle** in depth.
- Learn **how MBSE supports each lifecycle stage**.
- Understand and apply the **MBSE Workflow** step-by-step.
- Become familiar with **SysON** for modeling SysML v2 systems.
- Successfully **install and verify the modeling tool** ready for Week 3.
- Conduct a **first simple hands-on exploration** of a system model inside SysON.

QUESTION!