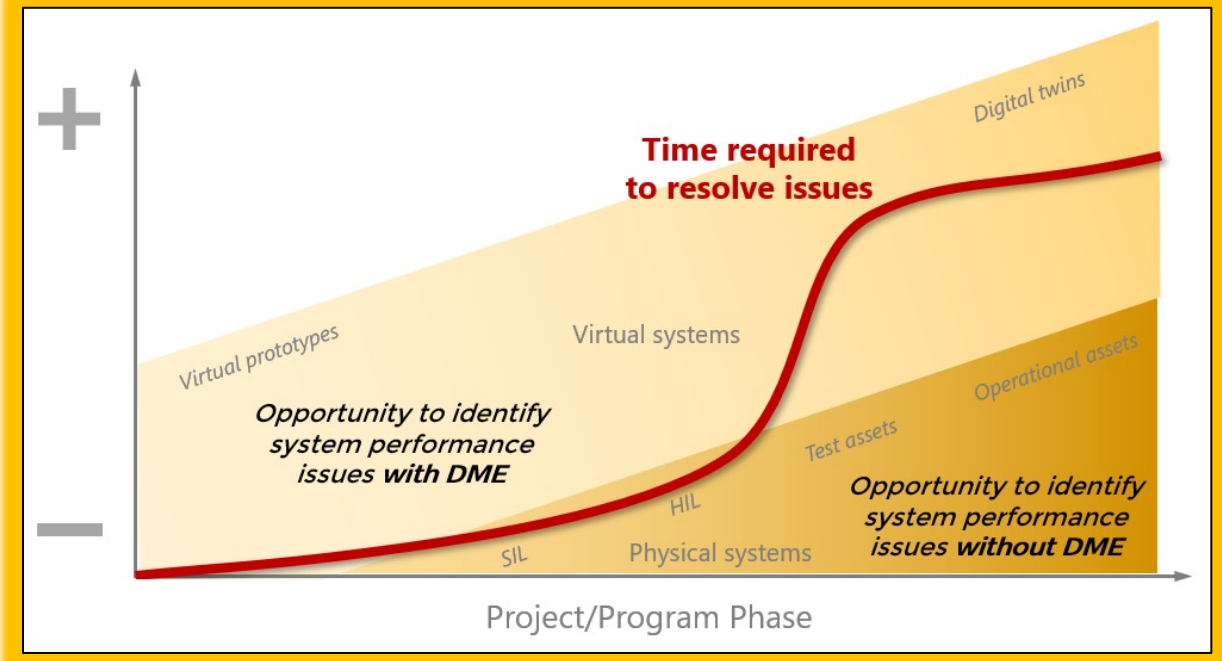
Integrating MBSE with Digital Mission Engineering

Aerospace and Defense

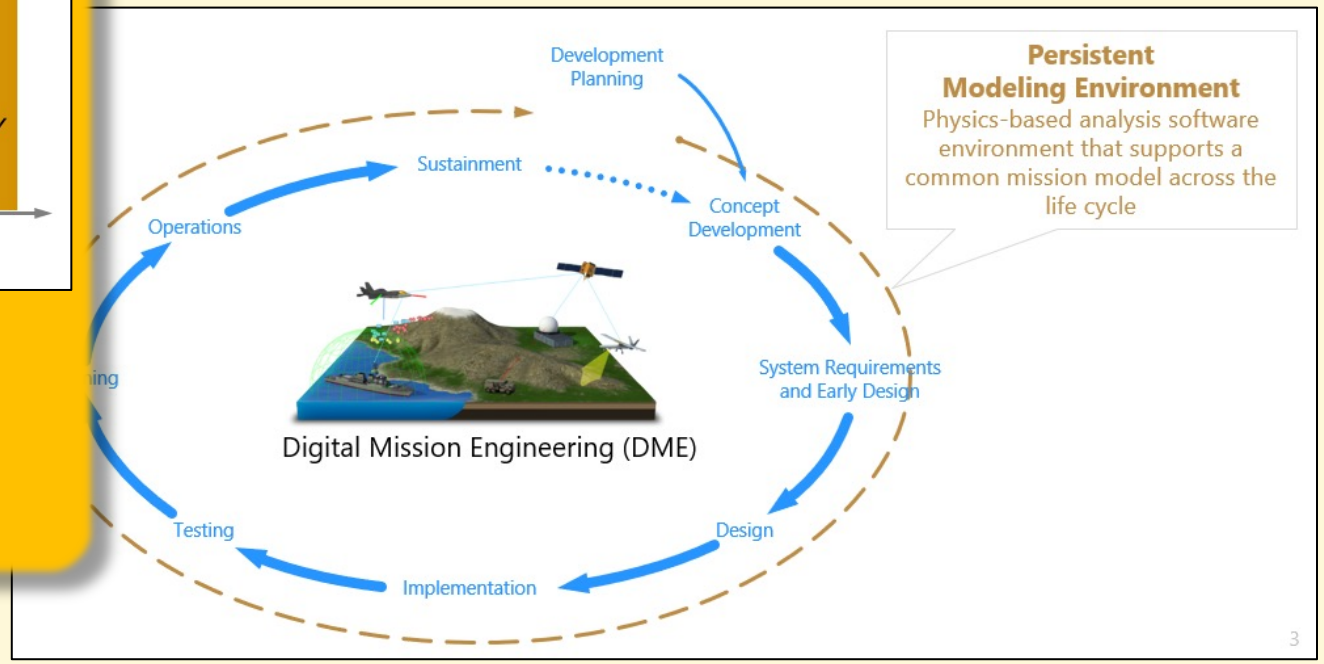
Ansys

Digital Mission Engineering (DME) Accelerates Capability Delivery

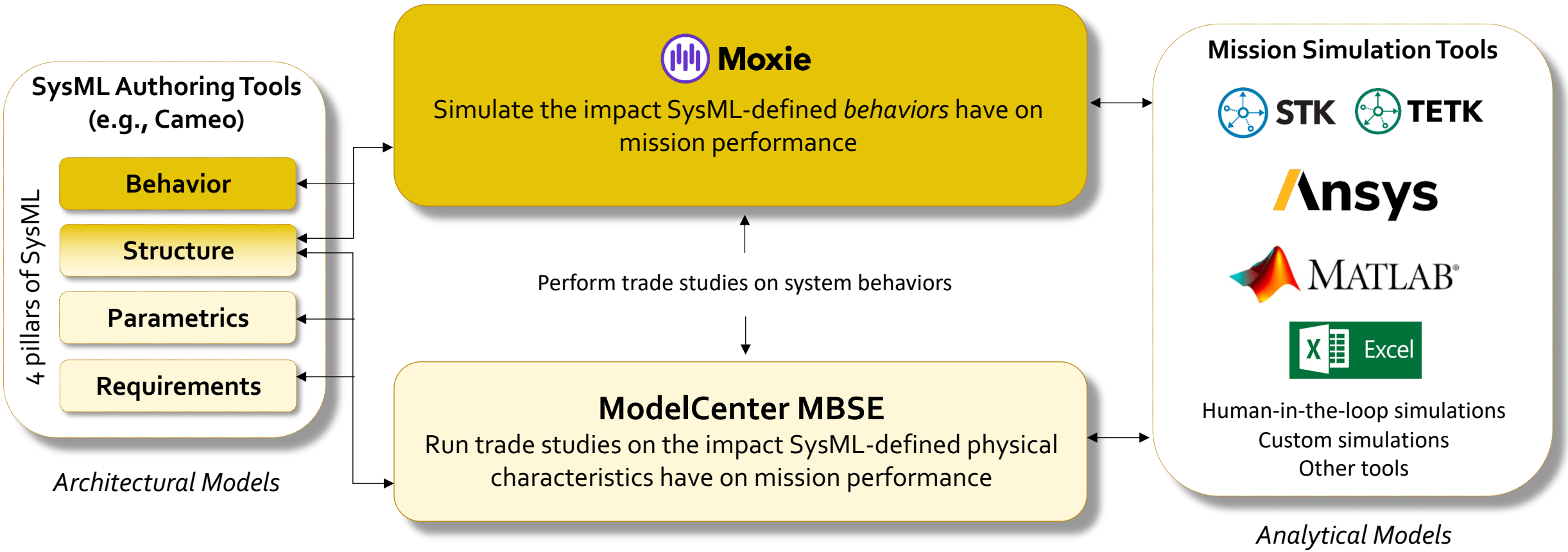


Accomplishing mission-focused performance assessment early in the lifecycle reduces delivery time and cost

Continuing mission-focused performance assessment throughout the lifecycle using a common mission model reduces rework and improves stakeholder communication

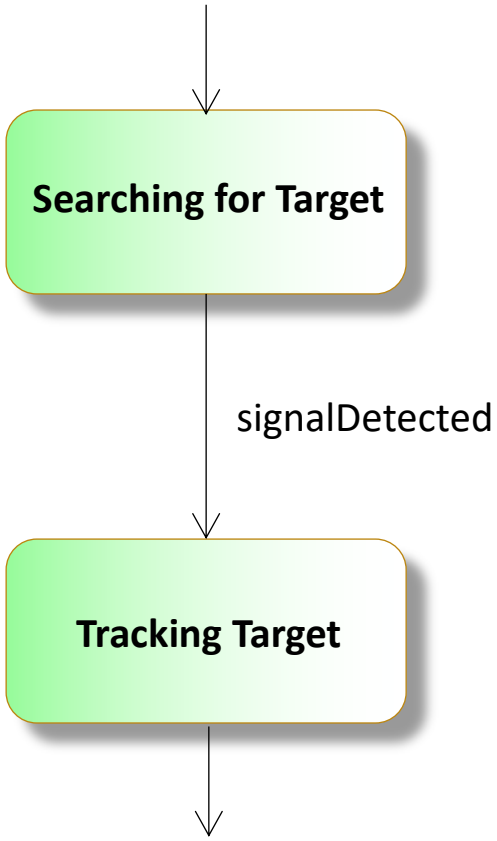


Connect SysML to Digital Mission Engineering

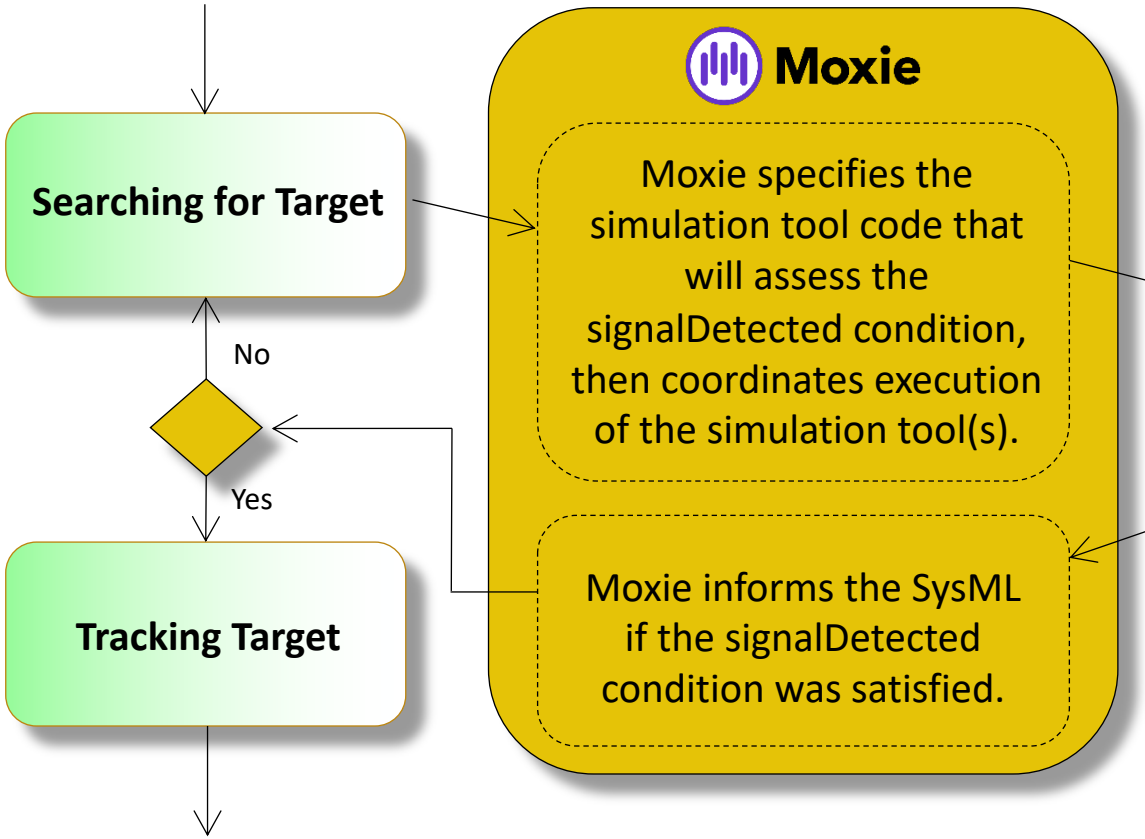


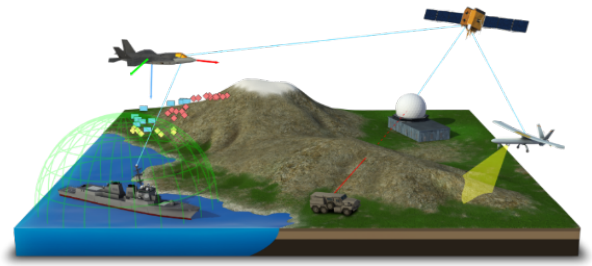
Simulate Behaviors with Moxie

System Engineer view of a state machine in a SysML authoring tool



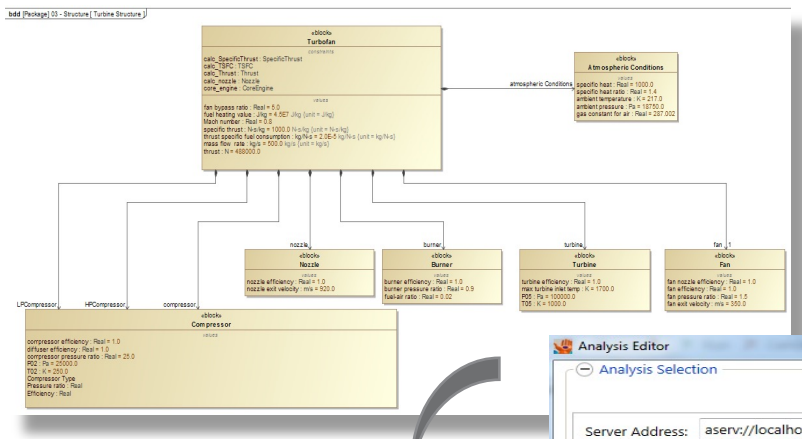
Behind the scenes view of Moxie simulating the *signalDetected* transition

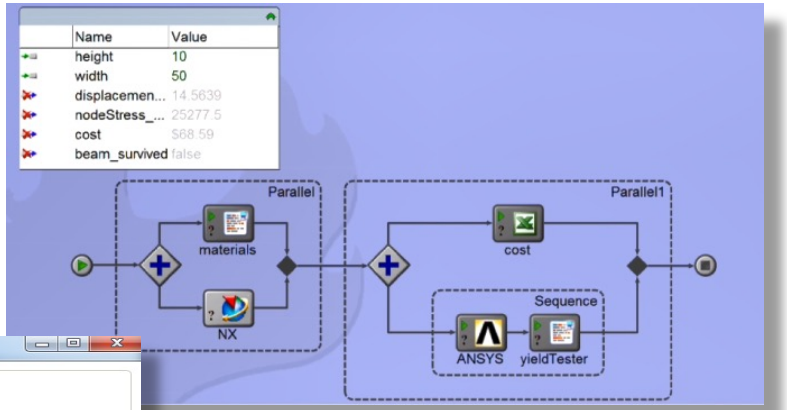




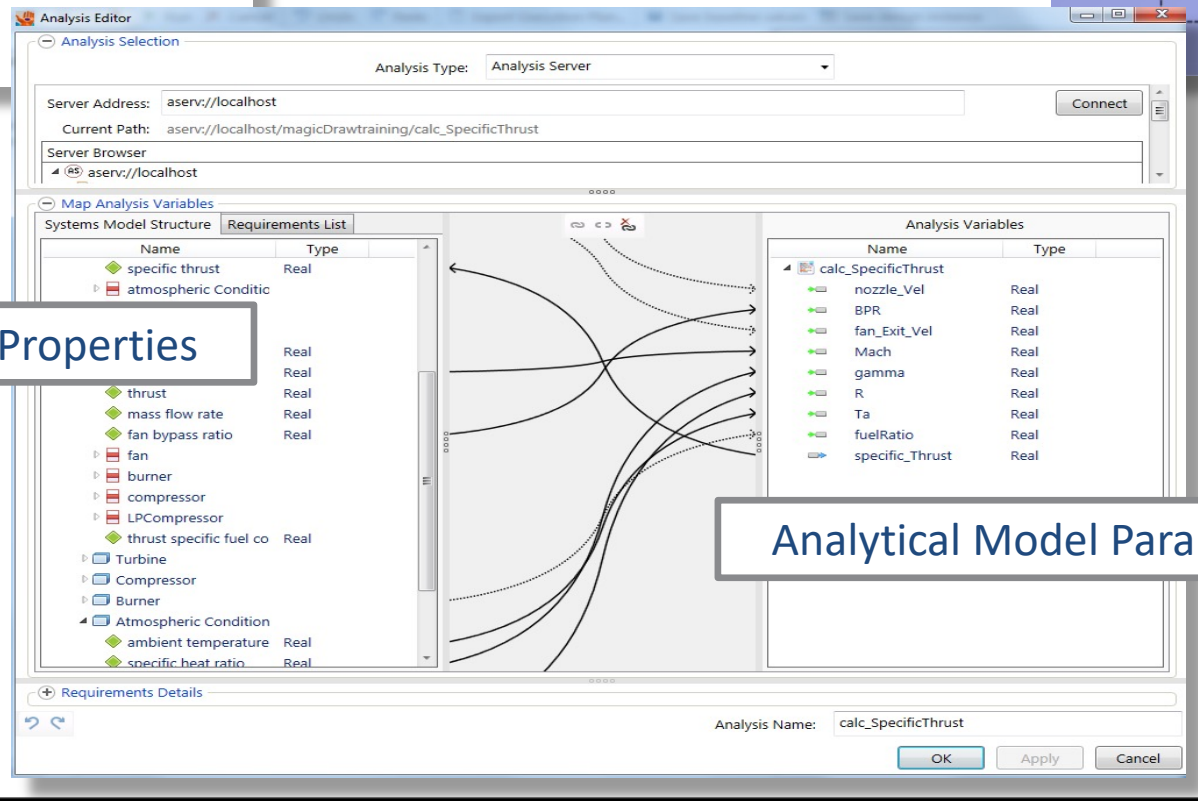
Simulation tools such as STK assess if the signalDetected condition is satisfied using precise event-based timing

Run Trade Studies with ModelCenter MBSE





MBSE Value Properties



Analytical Model Parameters

Example Use Case: Communications Satellite Design

SysML Authoring Tools (e.g., Cameo)

4 pillars of SysML

- **Behavior**
- **Structure**
- **Parametrics**
- **Requirements**

Moxie
Behavioral simulation

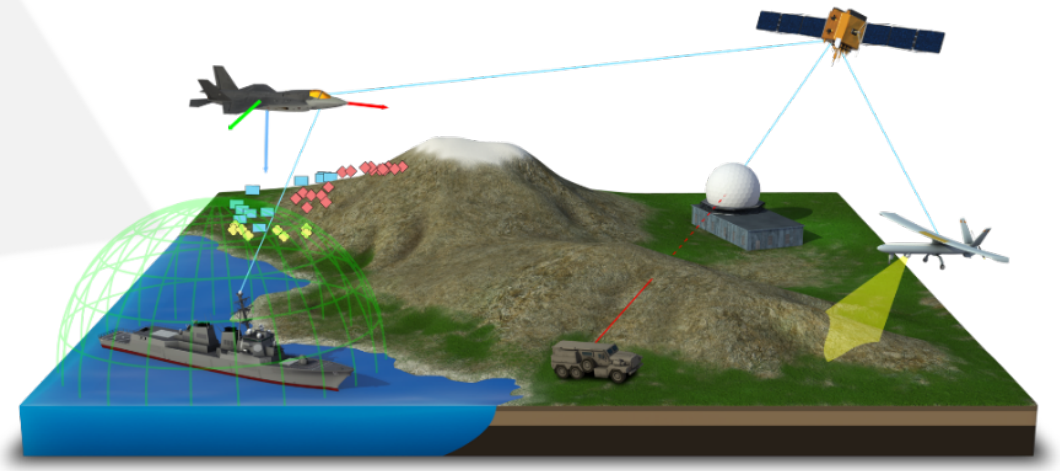
Moxie facilitates simulation of the SysML-defined *behavioral* characteristics of the satellite system relative to the Design Reference Mission requirements modeled in STK and other tools.

For example, are the satellite's data caching behaviors adequate to handle communication degrade scenarios that evolve during the course of a particular mission scenario?

ModelCenter MBSE
Trade studies

ModelCenter facilitates trade studies on the SysML-defined *physical* characteristics of the satellite system relative to Design Reference Mission requirements modeled in STK and other tools.

For example, is the gain pattern provided by a particular antenna design adequate to satisfy communication relay requirements in particular mission scenarios?

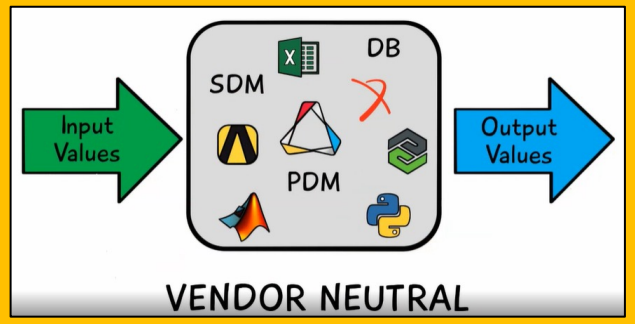


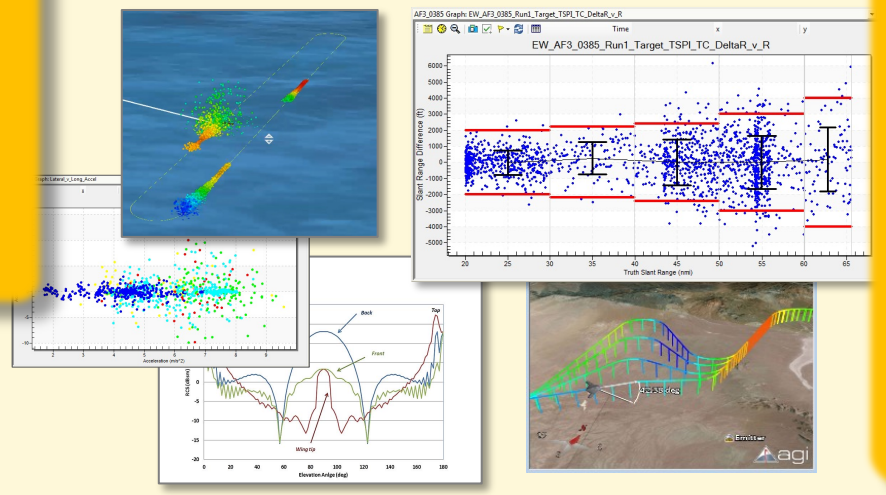
Design Reference Mission (DRM) modeled in tools such as STK

Integrate Ansys MBSE Tools with Other Data, Tools, and Workflows

- Multiple integration approaches exist to support any digital engineering environment

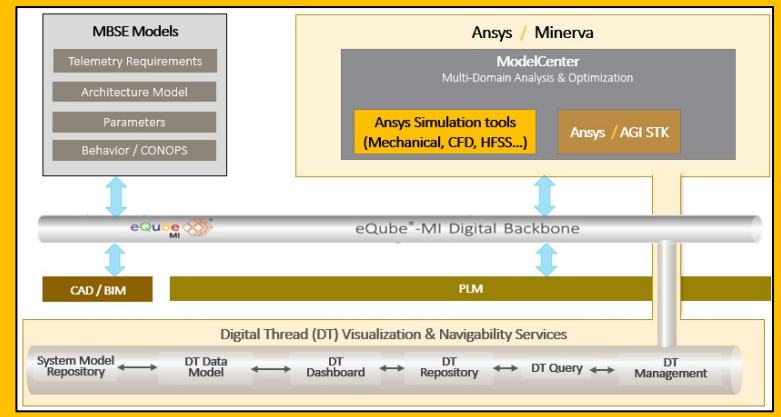
ModelCenter's black box approach enables integration of disparate tools into common workflows (e.g., costing, meshes, PLM/PDM)





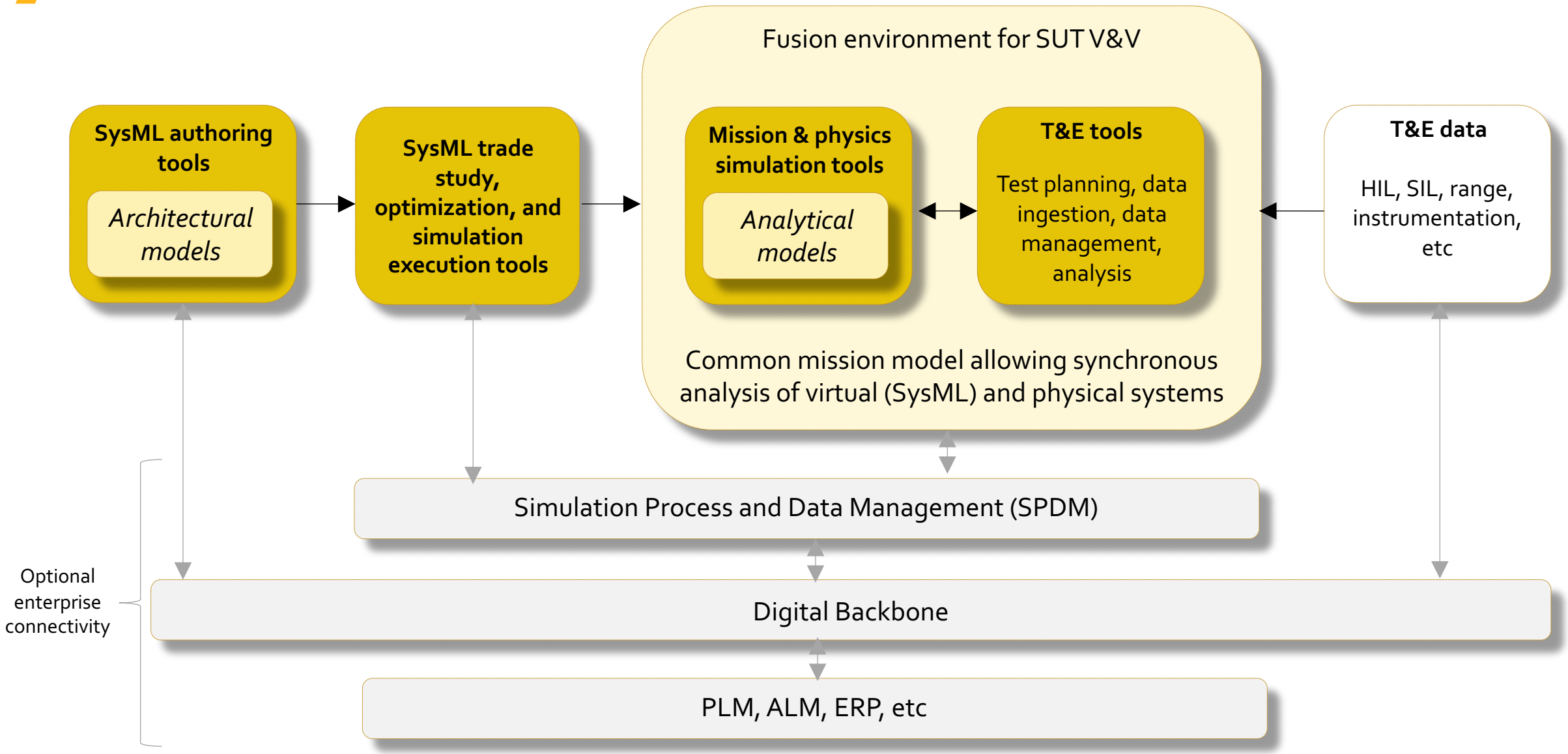
TETK automates import of data from external sources, allowing your STK-based MBSE/DME analysis to be enhanced with data generated by your broader Test & Evaluation environment (e.g., SIL, HIL, range data)

The **Minerva** SPDM tool provides a single ASoT for mission engineering data, optimizes simulation workflows, and connects to the broader digital engineering ecosystem

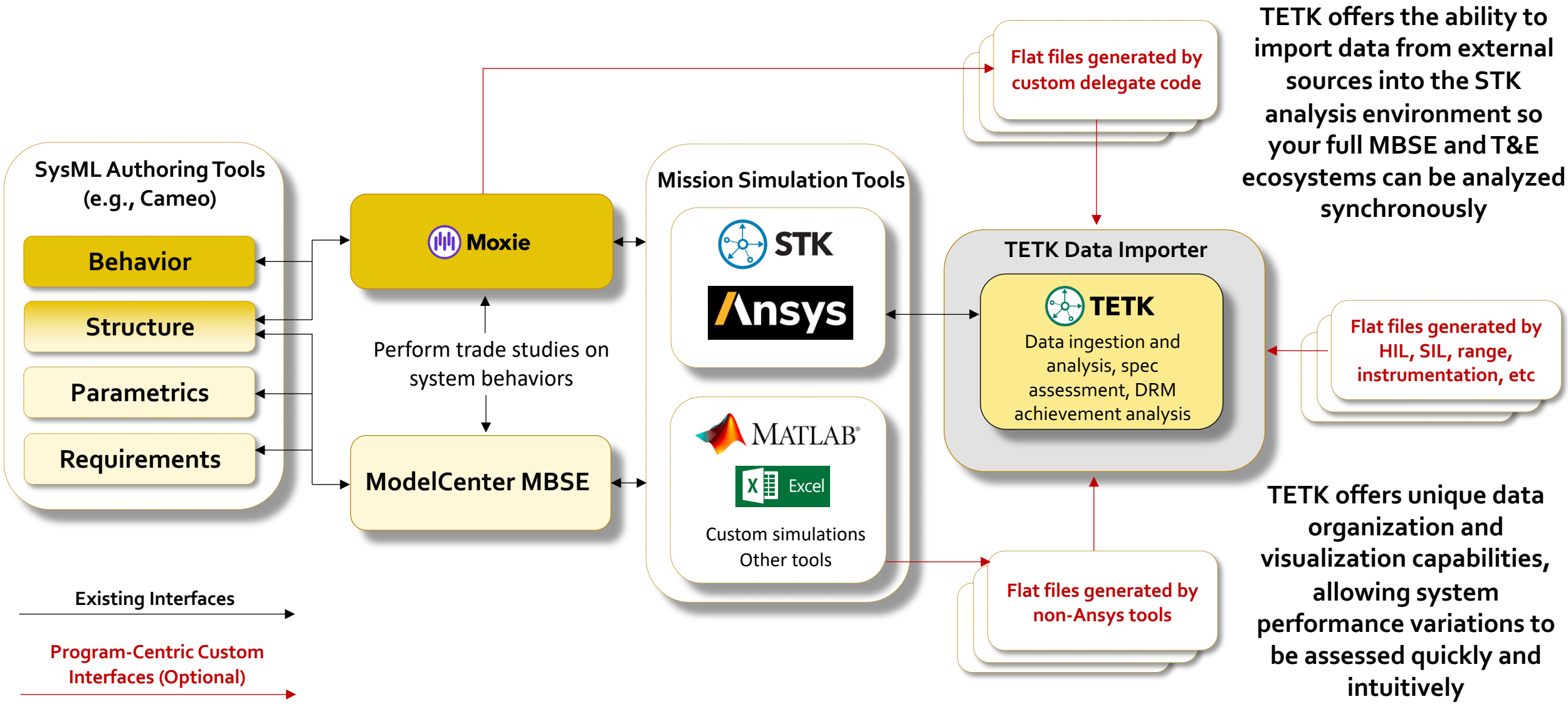


Additional Information

MBT&E Architecture



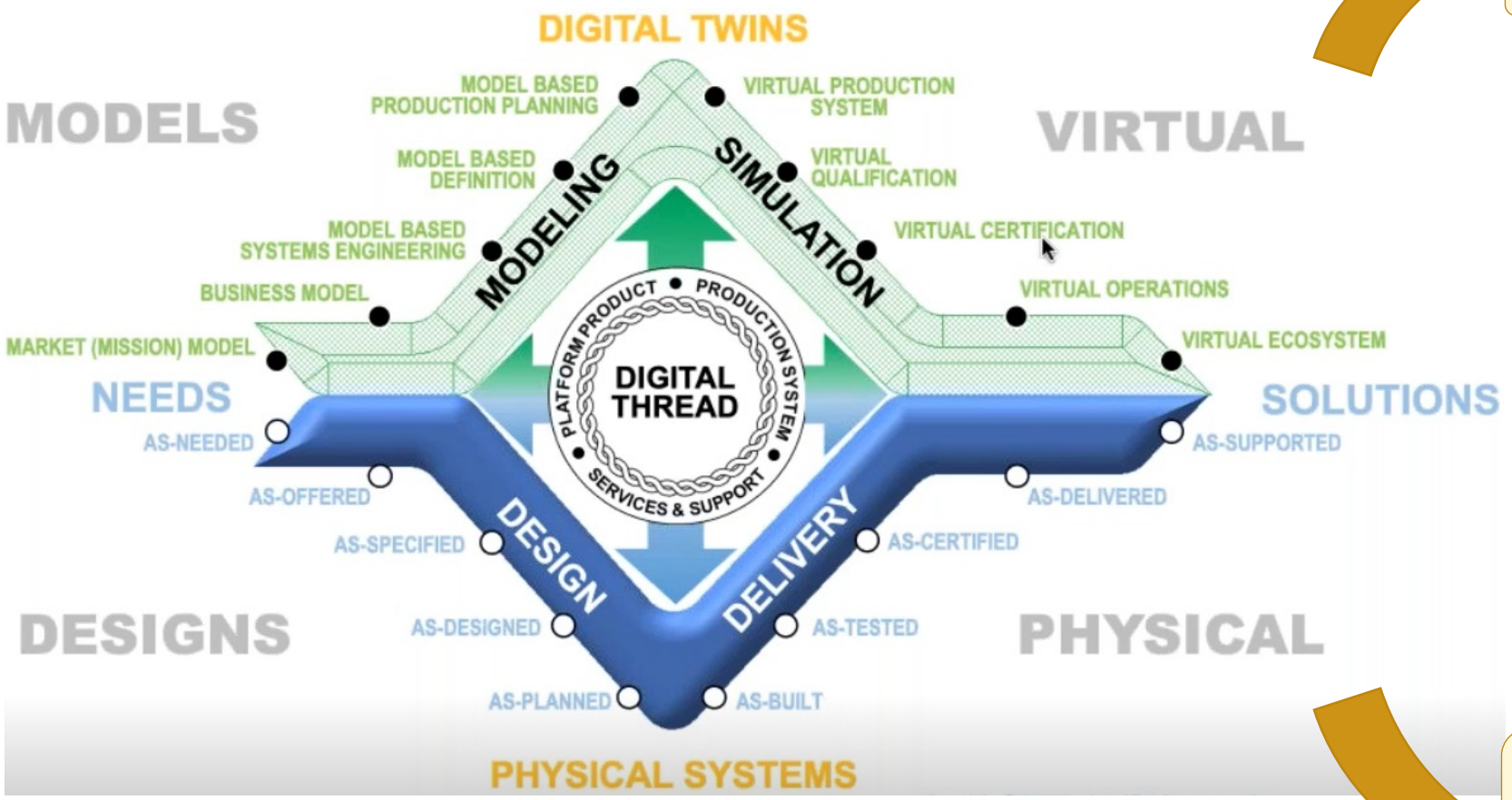
MBT&E Tools



TETK offers the ability to import data from external sources into the STK analysis environment so your full MBSE and T&E ecosystems can be analyzed synchronously

TETK offers unique data organization and visualization capabilities, allowing system performance variations to be assessed quickly and intuitively

Tune Tour Digital Twin with Physical System Data



https://www.boeing.com/features/innovation-quarterly/may2017/feature-technical-model-based-engineering.page

SysML

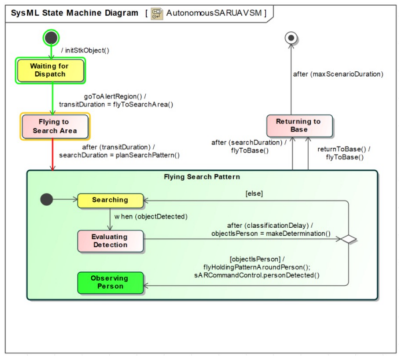
ModelCenter & Moxie

STK
Common mission model allowing detailed comparison of virtual vs. physical performance

TETK

Data from SIL, HIL, instrumentation, range, etc

Steps to Integrate SysML Behaviors with STK via Moxie



Model your System with SysML

Use your SysML modeling tool (e.g., Cameo) to create structures and behaviors that model the system you want to simulate

```
@DelegateFor("SARStructure::AutonomousSARUAV")
@AutoDelegateImplementation
public interface AutonomousSARUAV extends SpatialEntity {
  Property<SARCommandControl> sARCommandControlProperty();
  Property<TimeDuration> transitDurationProperty();
  Property<TimeDuration> searchDurationProperty();
  Property<TimeDuration> maxScenarioDurationProperty();
  Property<BooleanValue> objectDetectedProperty();
  Property<TimeDuration> classificationDelayProperty();
  Property<Boolean> objectIsPersonProperty();
  Property<String> stkObjectPathProperty();
  Property<String> platformTypeProperty();
  Property<Double> searchAltitudeProperty();
  Property<Double> searchPatternWidthProperty();
  Property<Double> searchVolumeFOVProperty();
  Property<BooleanValue> fuelStateLowProperty();
  Property<Double> initialFuelPercentProperty();

  void goToAlertRegion();
  void returnToBase();
  TimeDuration flyToSearchArea();
  TimeDuration planSearchPattern();
  boolean makeDetermination();
  void flyHoldingPatternAroundPerson();
  void initStkObject();
  void flyToBase();
}
```

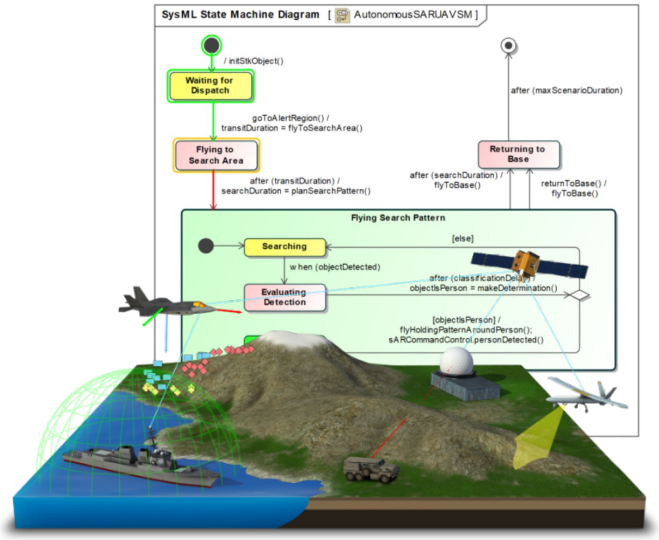
Autogenerate Moxie Code

Use Moxie's Java code generator to export interfaces and/or classes representing the SysML blocks in your system. Each interface or class includes stubs for properties and operations, along with a mapping to its corresponding SysML block.

```
@Override
public TimeDuration planSearchPattern()
{

    try {
        double currentTime = m_toolBox.getTimeHelper().moxieTimeToStkEpochSeconds(m_timeProvider.getCurrentTime());
        MoxieTime interruptTime = m_timeProvider.getCurrentTime();
        //create procedure to get from initial location to middle of search area
        boolean interrupted = false;

        IAgAvtrProcedure interruptProcedure = m_uavProcedures.get(m_uavProcedures.getCount()-1);
        interruptProcedure.getTimeOptions().setUseInterruptTime(true);
        interruptProcedure.getTimeOptions().setInterruptTime(currentTime+1);

        //create holding procedure to start
        IAgAvtrProcedureAreaTargetSearch atSearch = (IAgAvtrProcedureAreaTargetSearch) m_uavProcedures.add(AgEAvtrS
        IAgAvtrSiteSTKAreaTarget uavSite = (IAgAvtrSiteSTKAreaTarget) ((IAgAvtrProcedure) atSearch).getSite();
        //set values
        uavSite.setObjectName("AreaTarget/" + ((stkSARCommandControl)sARCommandControlProperty().getValue()).getAle
        atSearch.getAltitudeOptions().setUseDefaultCruiseAltitude(false);
        atSearch.getAltitudeOptions().setAltitude(searchAltitudeProperty().getValue());
        atSearch.setMaxSeparation(searchPatternWidthProperty().getValue());
```

Connect to STK

Populate the Java method stubs with STK Object Model code to specify which STK algorithms will execute on which STK objects during your SysML state machine transitions. Your Object Model code can connect to an existing STK scenario or generate a new scenario.



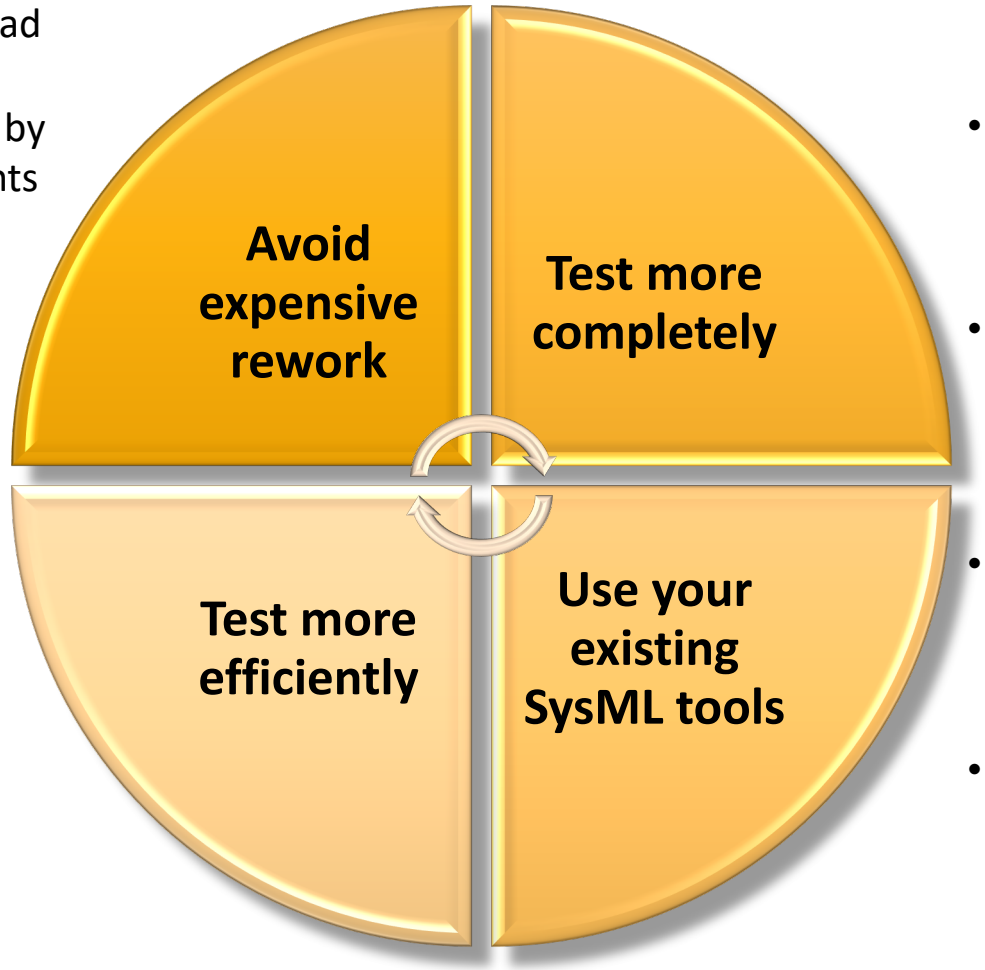
Simulate and Analyze

Use your SysML modeling tool's interface to start, pause, and debug Moxie's execution of your state machines in the STK mission environment. Analyze and validate your system's performance using STK's capabilities synchronized with state machine visuals and Moxie logs.

Iterate to increase the fidelity of your simulation

Key Value Points for Moxie

- Mature and validate system behaviors using your SysML architectures instead of a physical system
- **Reduce cost and development time** by achieving system performance insights earlier in the life cycle



- **Execute** your SysML *behavioral* models in a physics-based mission environment
- Connect your authoritative source of design truth (MBSE architectures) to your system's targeted operational environment
- Explore system performance in new and emerging mission scenarios

- Improve analysis speed and accuracy, since Moxie coordinates time across all objects in the mission simulation using **precise simulation tool timing**
- Inject custom code into your SysML state machine transitions to **customize analysis fidelity** based on your objectives

- Systems engineers can execute implemented Moxie workflows using SysML tools (e.g., Cameo) that they're already familiar with.
- Your team can **focus on system modeling** rather than creating physics algorithms and numerical integration schemes to represent the mission environment


Ansys