



# MBSE in Aerospace Industry

Dr. Ebad Jahangir & Mr. Randy Skelding

Pratt & Whitney, United Technologies Corporation

June 5, 2014



**United Technologies**

# Today's Agenda

United Technologies Corporation

Systems Engineering in UTC

Requirements Definition and Management

Modeling and MBSE



**United Technologies**



**INCOSE**  
New England



# United Technologies

Powerful  
**Innovation**

Powerful  
**Solutions**



**United Technologies**



**INCOSE**  
New England

# Providing high-technology commercial building and aerospace systems to customers around the world



# UTC BUSINESSES



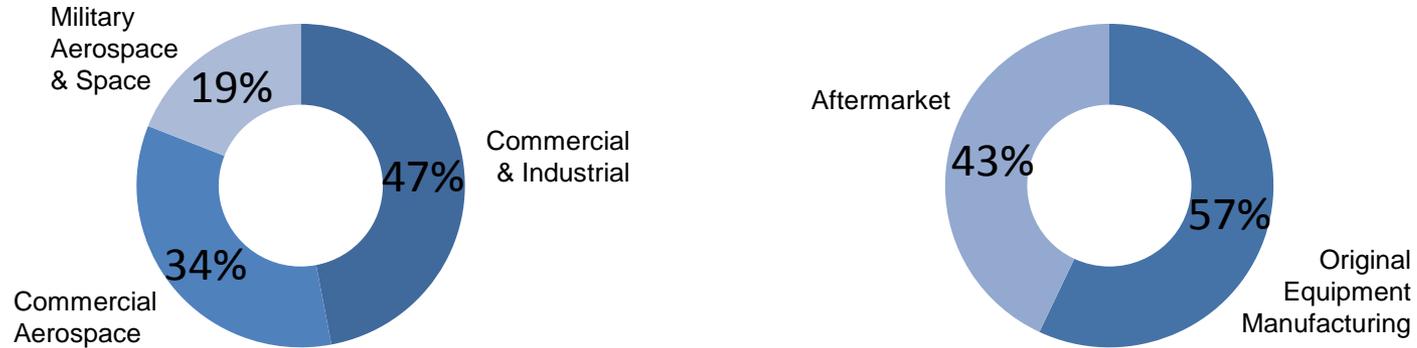
\*Otis and UTC Climate, Controls & Security each continue to report their financial and operational results as separate segments.



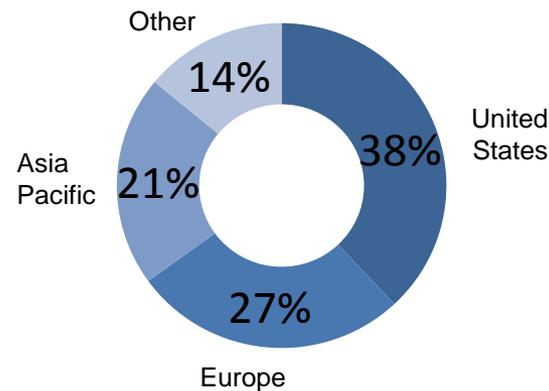
# PERFORMANCE

## 2013 net sales \$62.6 billion

### TYPE



### GEOGRAPHY



# OUR PEOPLE

More than 200K people in ~70 countries



# OUR COMMITMENTS



## INNOVATION

We are a company of ideas that are nurtured by a commitment to research and development. The achievements of our founders inspire us to reach always for the next innovative and powerful and marketable idea. We seek and share ideas openly and encourage diversity of experience and opinion.

## RESPONSIBILITY

Successful businesses improve the human condition. We maintain the highest ethical, environmental and safety standards everywhere, and we encourage and celebrate our employees' active roles in their communities.

## PERFORMANCE

Our customers have a choice, and how we perform determines whether they choose us. We aim high, set ambitious goals and deliver results, and we use customer feedback to recalibrate when necessary. We move quickly and make timely, well-reasoned decisions because our future depends on them. We invest authority where it needs to be, in the hands of the people closest to the customer and the work.

## OPPORTUNITY

Our employees' ideas and inspiration create opportunities constantly, and without limits. We improve continuously everything we do, as a company and as individuals. We support and pursue lifelong learning to expand our knowledge and capabilities and to engage with the world outside UTC. Confidence spurs us to take prudent risks, to experiment, to cooperate with each other and, always, to learn from the consequences of our actions.

## RESULTS

We are a preferred investment because we meet aggressive targets whatever the economic environment. We communicate honestly and forthrightly to investors, and deliver consistently what we promise. We are a company of realists and optimists, and we project these values in everything we do.

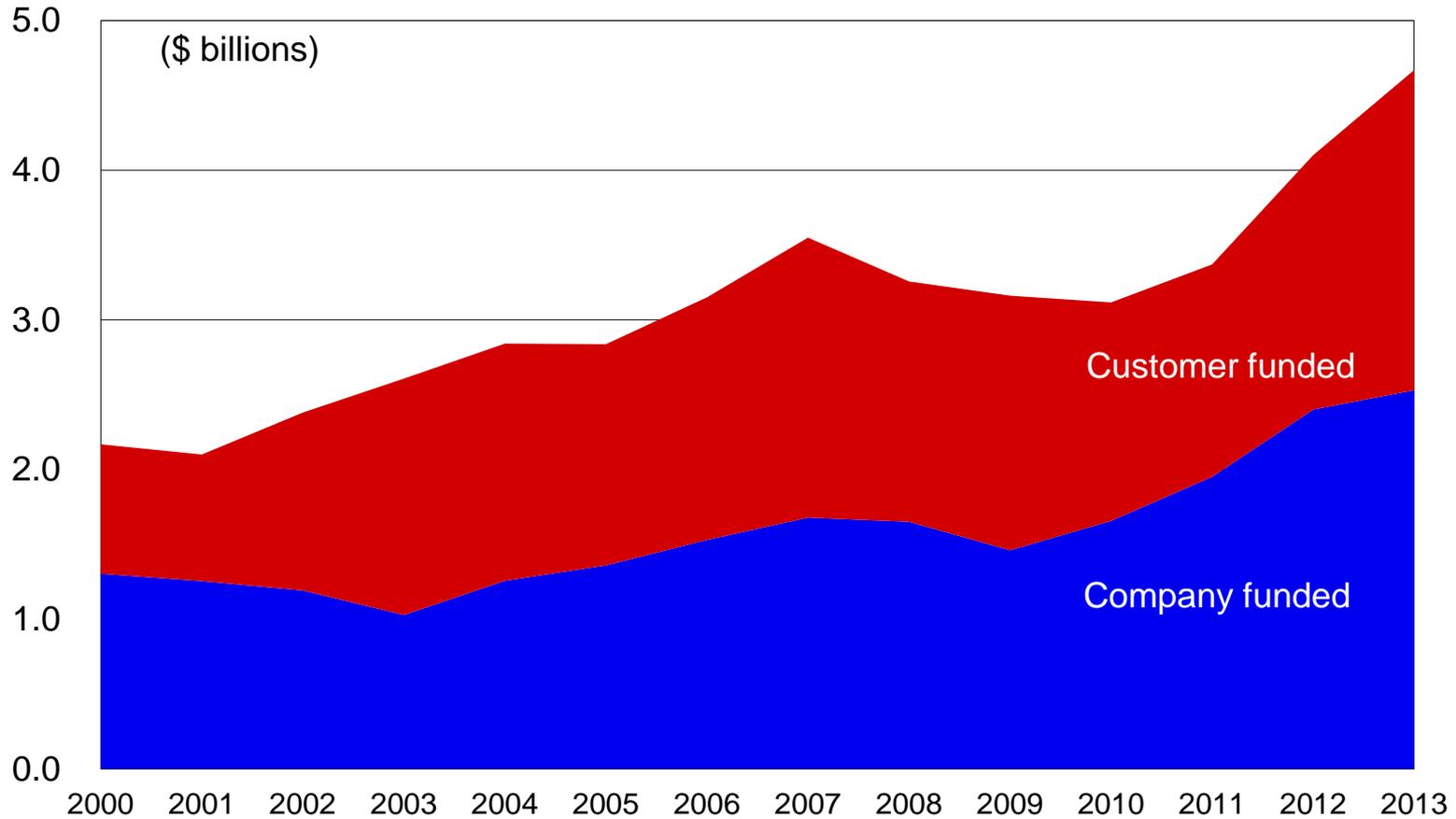


**United Technologies**



# COMMITMENT TO R&D

## Investing in the future



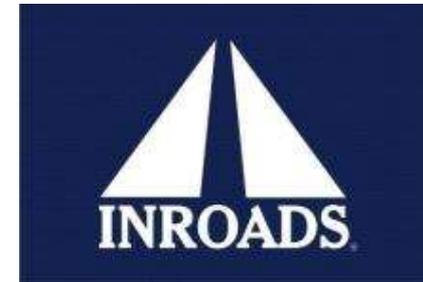
\$4.7 billion invested in 2013

2000 - 2007 includes discontinued operations

# RECOGNITION



Distinguished Service Award  
from the United Service Organizations



Top Corporate Partner,  
7<sup>th</sup> consecutive year





火龙果·整理  
uml.org.cn



# United Technologies



## OTIS

United Technologies



## IAE International Aero Engines



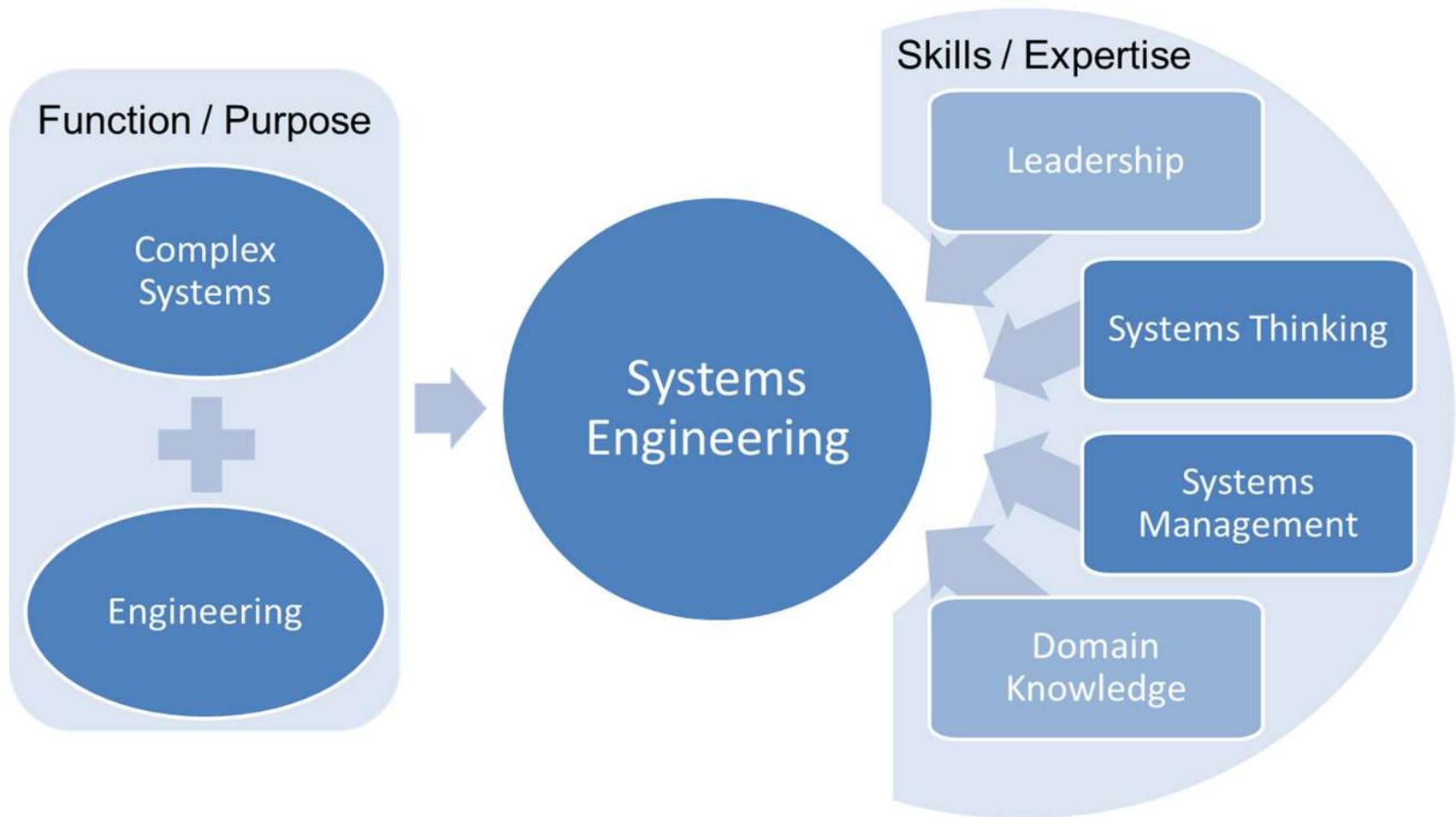
## UTC Aerospace Systems



Systems Engineering is defined within United Technologies as the process which rigorously translates customer needs into a structured set of specific requirements, synthesizes a system architecture that satisfies those requirements, and allocates them in a physical system, meeting cost, schedule and performance objectives throughout the life cycle

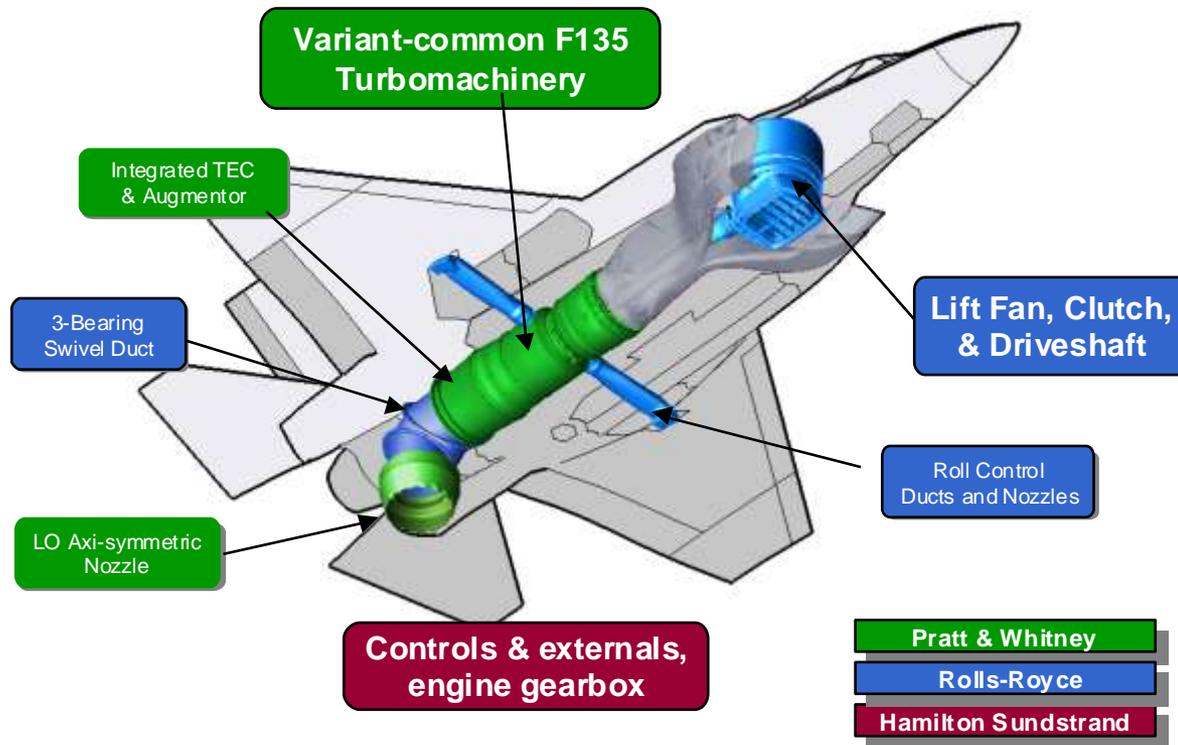


# This is What SE is About



Systems Engineers *lead* the realization or modification of complex systems in a *Systematic* manner through a *Systemic* understanding of the *issues*

## Propulsion System Complexity Driving Need for More Robust Systems Engineering Process and Tools



# System Requirements

System requirements are all of the requirements at the system level that describe the functions which the system as a whole should fulfill to satisfy the stakeholder needs and requirements,

- and is expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary

A requirement is a statement that identifies a product or process operational, functional, or design characteristic or constraint,

- which is unambiguous, testable, or measurable and necessary for product or process acceptability (ISO/IEC 2007).



# Classification of Requirements

Types	Description
Functional Requirements	Describe qualitatively the system functions or tasks to be performed in operation.
Performance Requirements	Define quantitatively the extent, or how well, and under what conditions a function or task is to be performed (e.g. rates, velocities). These are quantitative requirements of system performance and are verifiable individually. Note that there may be more than one performance requirement associated with a single function, functional requirement, or task.
Usability Requirements	Define the quality of system use (e.g. measurable effectiveness, efficiency, and satisfaction criteria).
Interface Requirements	Define how the system is required to interact or to exchange material, energy, or information with external systems (external interface), or how system elements within the system, including human elements, interact with each other (internal interface). Interface requirements include physical connections (physical interfaces) with external systems or internal system elements supporting interactions or exchanges.
Operational Requirements	Define the operational conditions or properties that are required for the system to operate or exist. This type of requirement includes: human factors, ergonomics, availability, maintainability, reliability, and security.
Modes and/or States Requirements	Define the various operational modes of the system in use and events conducting to transitions of modes.
Adaptability Requirements	Define potential extension, growth, or scalability during the life of the system.
Physical Constraints	Define constraints on weight, volume, and dimension applicable to the system elements that compose the system.
Design Constraints	Define the limits on the options that are available to a designer of a solution by imposing immovable boundaries and limits (e.g., the system shall incorporate a legacy or provided system element, or certain data shall be maintained in an online repository).
Environmental Conditions	Define the environmental conditions to be encountered by the system in its different operational modes. This should address the natural environment (e.g. wind, rain, temperature, fauna, salt, dust, radiation, etc.), induced and/or self-induced environmental effects (e.g. motion, shock, noise, electromagnetism, thermal, etc.), and threats to societal environment (e.g. legal, political, economic, social, business, etc.).
Logistical Requirements	Define the logistical conditions needed by the continuous utilization of the system. These requirements include sustainment (provision of facilities, level support, support personnel, spare parts, training, technical documentation, etc.), packaging, handling, shipping, transportation.
Policies and Regulations	Define relevant and applicable organizational policies or regulatory requirements that could affect the operation or performance of the system (e.g. labor policies, reports to regulatory agency, health or safety criteria, etc.).
Cost and Schedule Constraints	Define, for example, the cost of a single exemplar of the system, the expected delivery date of the first exemplar, etc.

# Modern Gas Turbine Optimization an Exercise in Managing Complexity

~ 80,000 PARTS

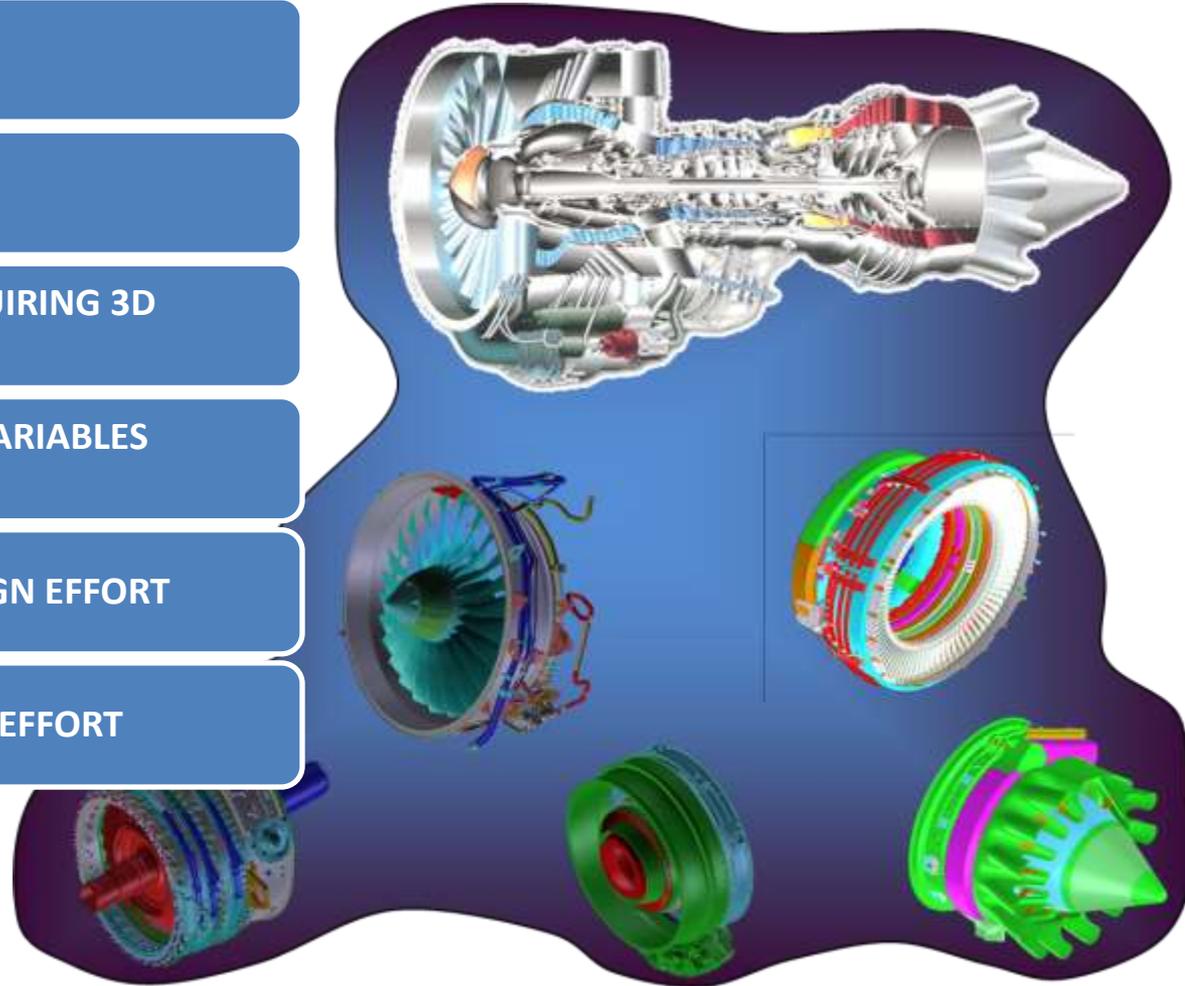
~ 5,000 PART NUMBERS

~ 200 MAJOR PART NUMBERS REQUIRING 3D  
FEA/CFD ANALYSIS

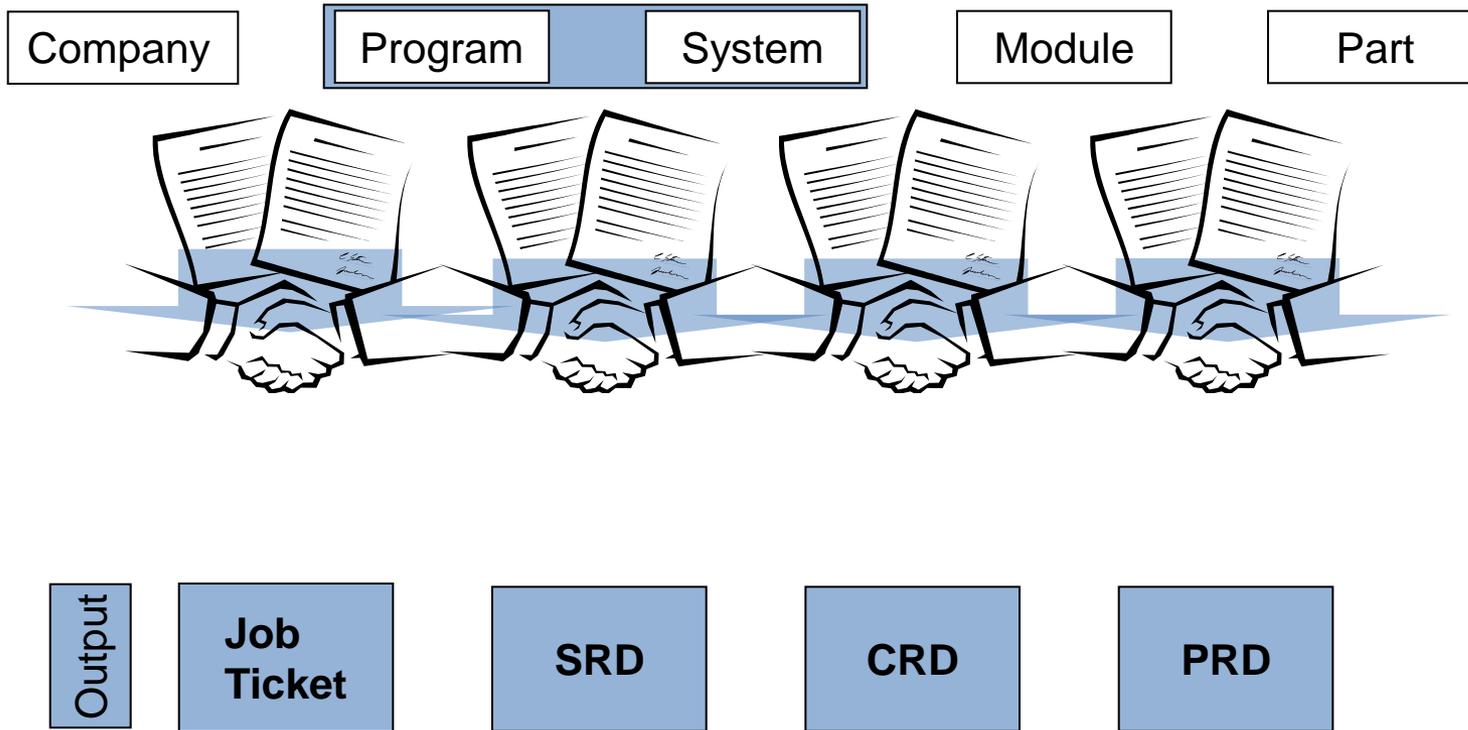
~ 5,000-10,000 PARAMETRIC CAD VARIABLES  
DEFINE MAJOR PART NUMBERS

~ 200 MAN-YEAR ANALYTICAL DESIGN EFFORT

~ 200 MAN-YEARS DRAFTING / ME EFFORT



# Requirements Management

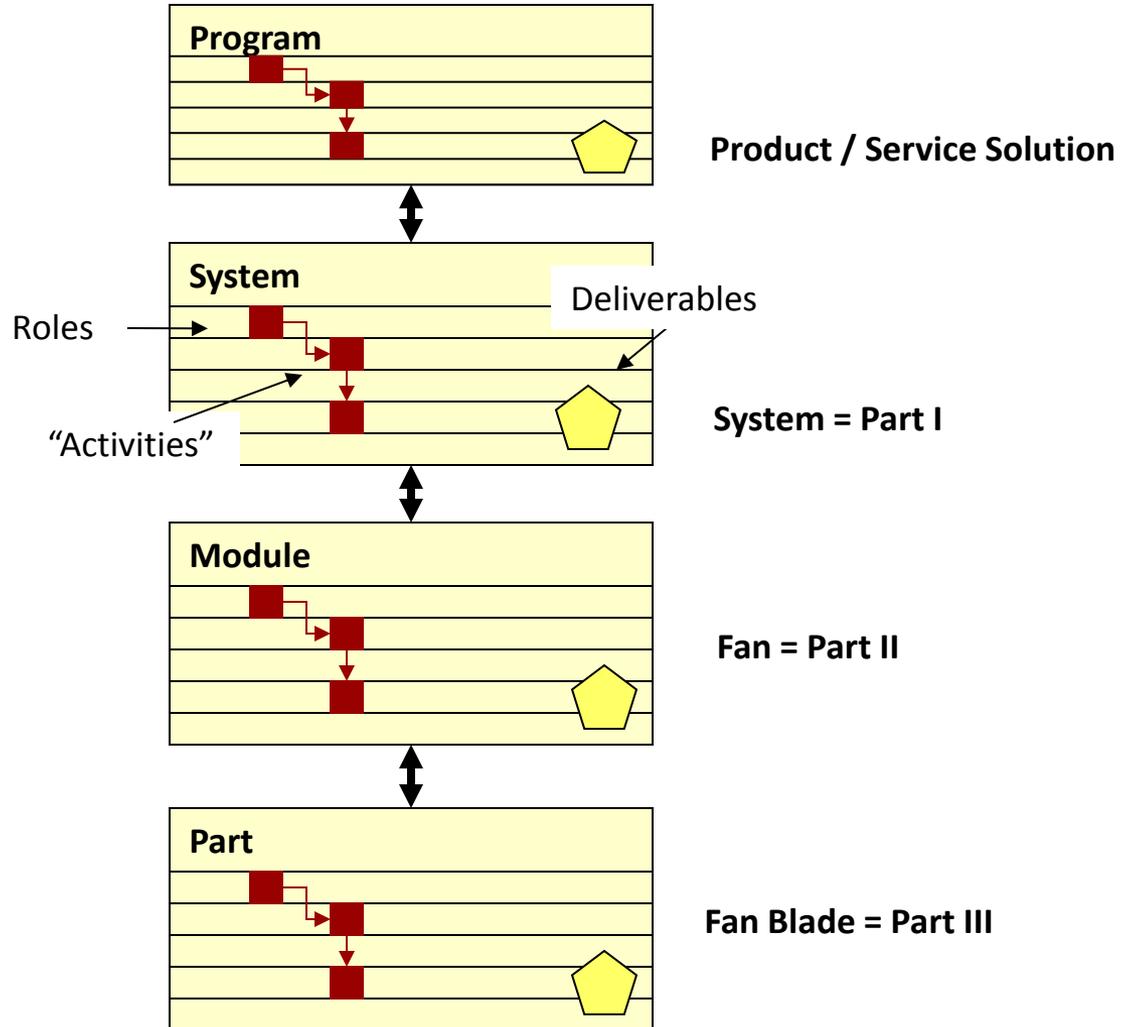


Requirements Flow to 3 Levels

# Job Ticket or Contract => Require 火龙果 · 整理 uml.org.cn

Job Ticket measures compliance to requirements

Job Ticket System Parameter
Performance
Weight
Efficiency
Reliability
Operability
Augmentor
Observability
Cost
Maintenance Cost
Durability



# Typical Requirements Flaws

## Ambiguity

- The natural language is not clear and it has to be “interpretation” is required.

## Non-Determinism

- The requirements allow to have choices at implementation level This does not mean that implementation must be non-deterministic.

## Inconsistency

- Some requirements are inconsistent to each other if they do not allow a solution that satisfies all of them.

## Vacuity

- A requirement is vacuous if by satisfying the other requirements it is implicitly satisfied.

## Realizability

- The requirement is not capable of being physically implemented

## Completeness

- All possible conditions would be covered.

## Extraneous

- The requirement does not belong to function being specified

## Negative

- Requirements makes verification difficult

## General

- Requirements makes verification difficult (always, under all conditions)



# The classical “paper” based method for Systems Requirements



Picture Source: Dr Peter Hoffman – IBM / Rational

# The classical “paper” based method for Systems Requirements



Picture Source: Dr Peter Hoffman – IBM / Rational



**United Technologies**



**INCOSE**  
New England

## Dilemma

- Complex systems leads to complex requirements; complex behavior is difficult to capture in any natural language like English

## Questions

- How can consequences of early commitments related to functional or nonfunctional requirements be understood at the earliest possible time during development?
- How can we make “requirements” more flexible over a greater portion of the system life cycle?





Need to manage complexity growth in requirements with cost/schedule effective methods

Models at different abstraction layers allow early and consistent guidance and enable formal analysis to design and verify correct behavior at different layers



# What is MBSE?

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 life cycle phases.



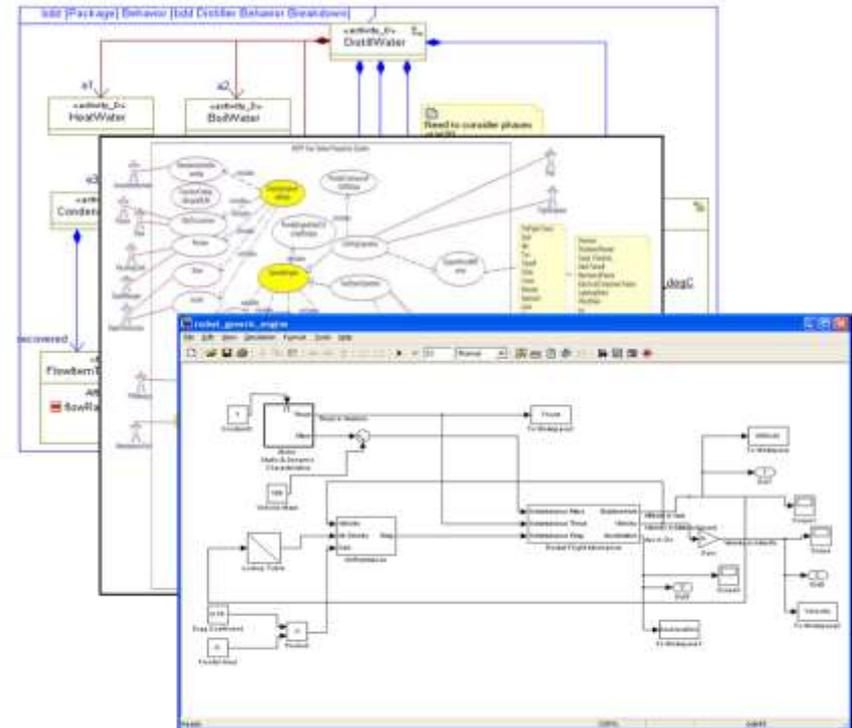
# MBSE Essential Elements

Models replace or enhance documents for specifying essential system behavior

Models capture multiple viewpoints about the system and have formal visual and internal representations

Models drive an analysis with tool / computer support (a drawing is not a model until it supports an analysis method)

Integrated models of various kinds form explicit design contracts for all aspects of the system



Source – see reference [ 4 ] PW Presentation at Technion July 2011

Model Based System Engineering integrates various engineering disciplines with executable models which enable the engineers to optimize entire system design for the product life cycle

# Challenge: Integrating Traditional Requirements with MBSE

## Traditional Requirements

- Customer-Supplier Contracts
- Certification

## MBSE

- Software development
- Integrated modeling maturity

You think because you understand 'one' you must also understand 'two', because one and one make two. But you must also understand 'and'.  
Rumi



# Functional Modeling with SysML

## “How to Start a Jet Engine”



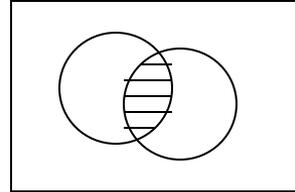
# Models have different purposes

- Functional Modeling
  - Logical relationships between activities and sequences in time
- Parametric Modeling
  - Extends Functional Modeling to include equations or models of constraints on physical and functional elements. Data/Results may be collected by repeated computer runs in time-domain or thru a separate Monte Carlo analysis.
- Dynamic Modeling
  - Focus is on mathematical representations of physical behavior of system or subsystem components. This may or may not be time domain.
- Business / Economic Modeling
  - Focus is on cost and schedule

*Connect the network of models together*



- Functional Modeling focuses on elucidating *functional* product requirements and the relationships between functional requirements.
- It is designed to catch situations like the following
  - Page 257 states “The valve shall be on” when yyy.
  - Page 5205 states “The valve shall be off” when zzz
  - **But the yyy and zzz conditions overlap, so the valve has to be both on and off at the same time.**
- It is designed to clearly show the states and modes of the system based on an analysis of the system requirements.

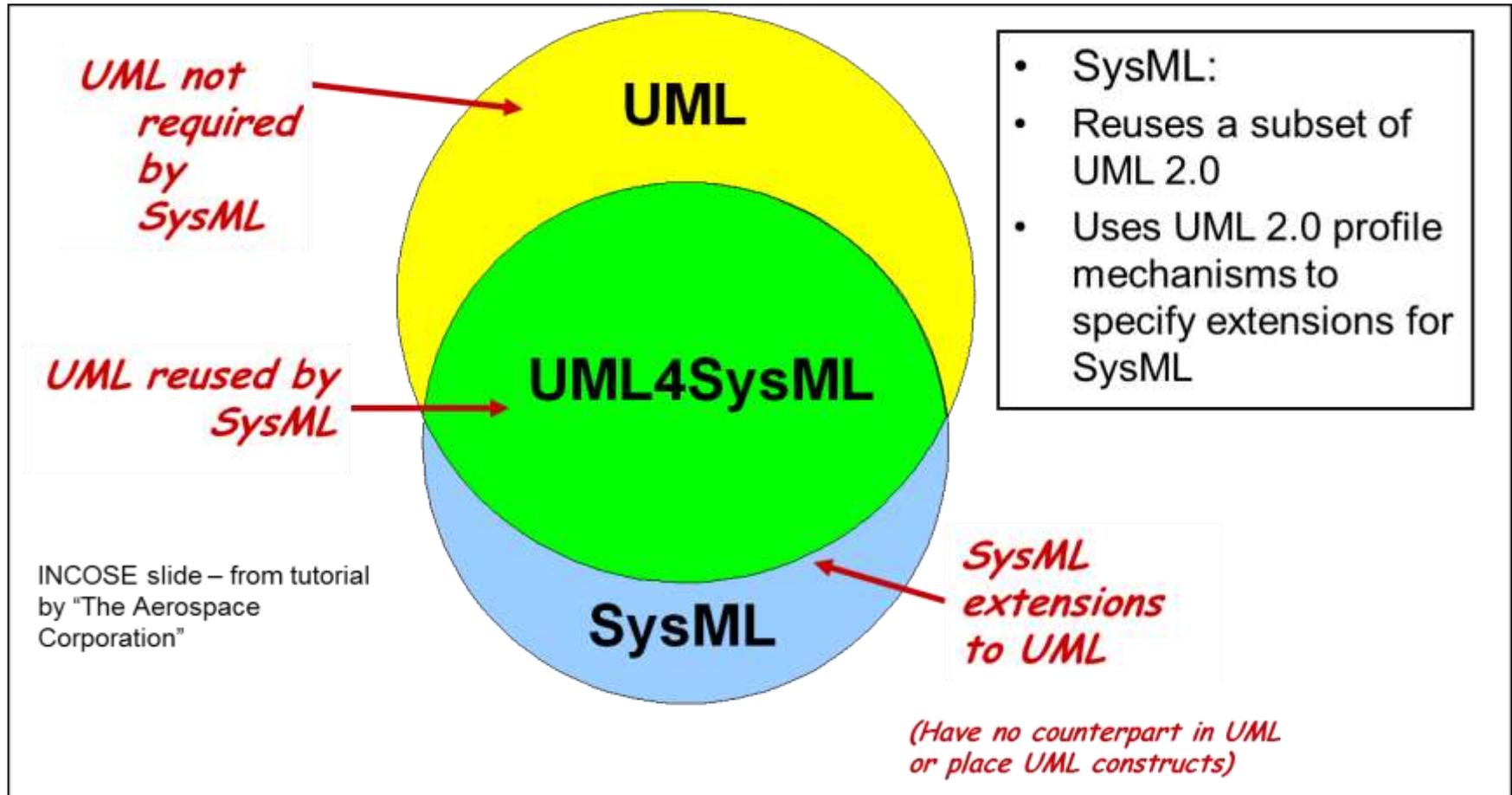


# Benefits of Functional Mode

- Exploration of design risks - moved “to the left”
- Details of test sequences uncovered.
- Better quality specifications.
- Fewer turnbacks during system testing.



# What is SysML?



***SysML tailored for Systems Engineering***



- Concept of Operations - multi view analysis, Use Case Analysis (**Top down - start with customer needs!**)
- Functional Modeling - SYSML, activity diagrams, state and sequence diagrams, **executable models - used for system V&V**
- Design Space Exploration / Trades - **SYSML Parametric Diagrams Glue Models Together**
- MDO expanded role, Abstraction layer based design, **Model libraries, Design Contracts**
- Simulation - Abstraction layers, Model libraries, Design Contracts.
- Formal methods - tie models together by analysis of design contracts. Create “**Probabilistic Certificates of Correctness**”.
- Model Integration Framework

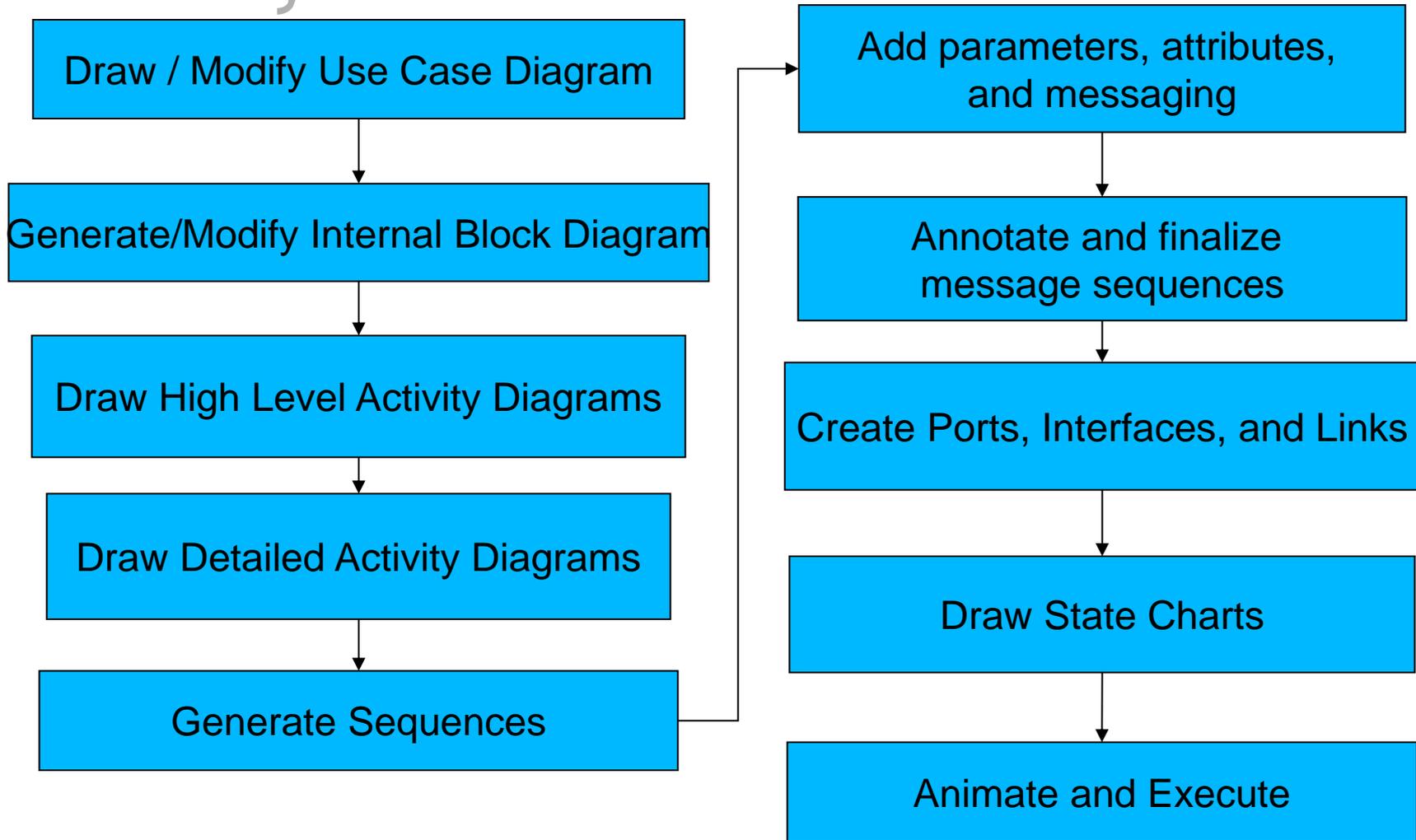


# Example Drawings for Functional Modeling using SYSML

- Use Case Diagram - captures system or subsystem scope
- Activity Diagram - captures tasks and control flow
- Internal Block Diagram - captures system structure and interfaces
- Sequence Diagram - captures details of interactions between system and external actors
- State Diagram - details states and modes of system

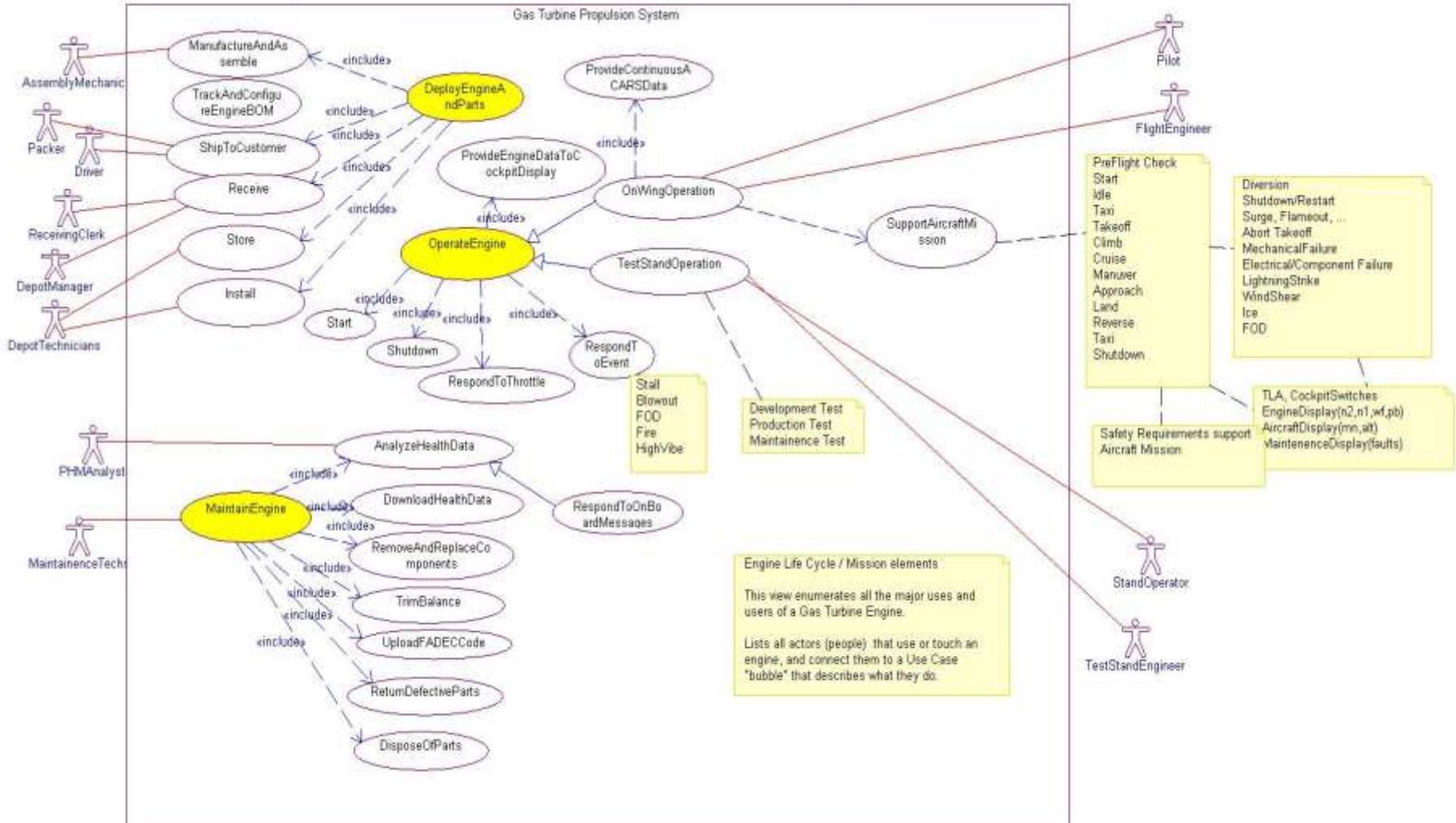


# The Harmony or “MBSE Best Practice Mini Cycle - as we use it

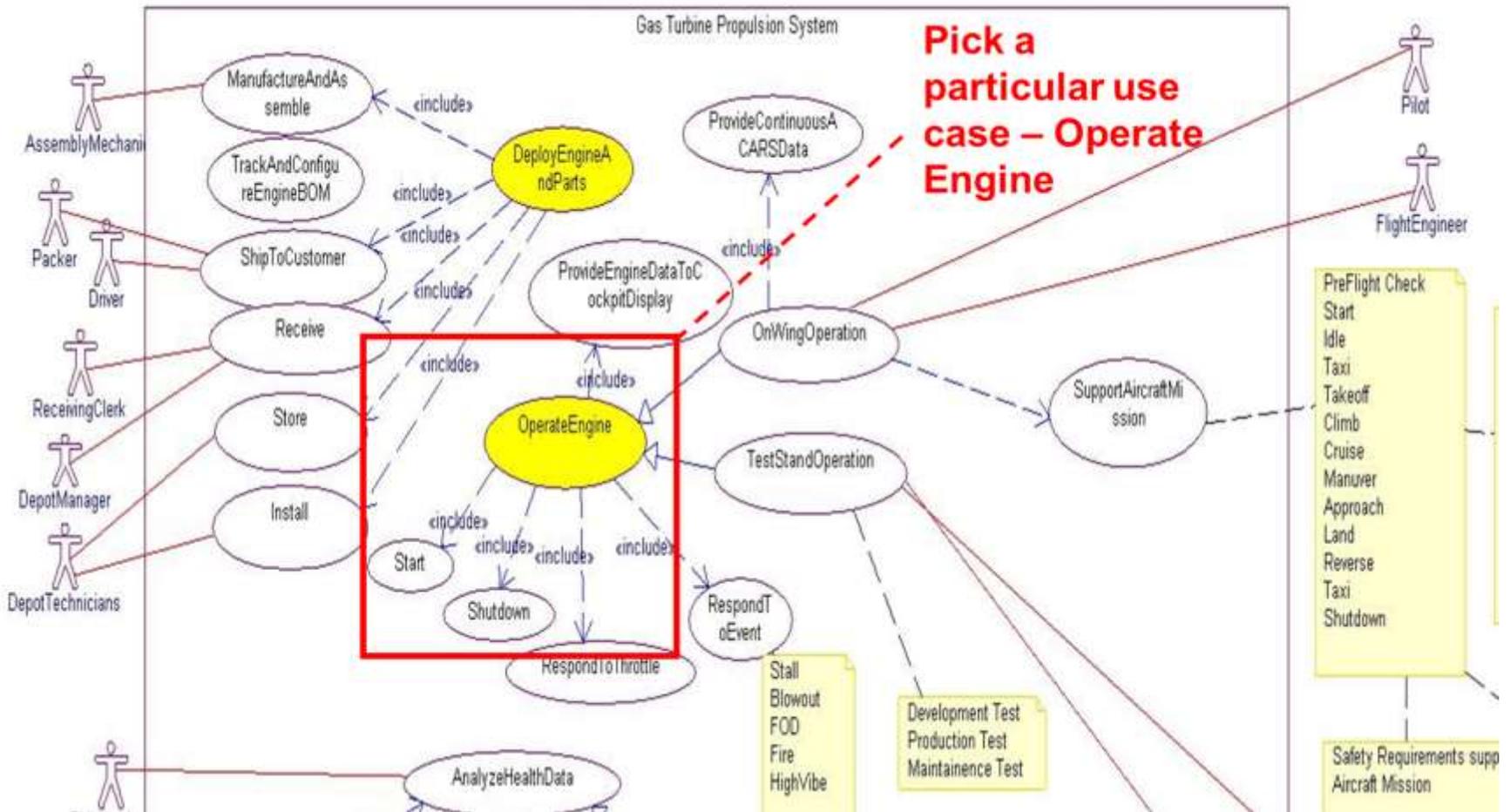


# Family of use cases - highest level system description

## Use Case Diagram – Gas Turbine Engine – family of use cases



# Lets zoom in so we can read the c



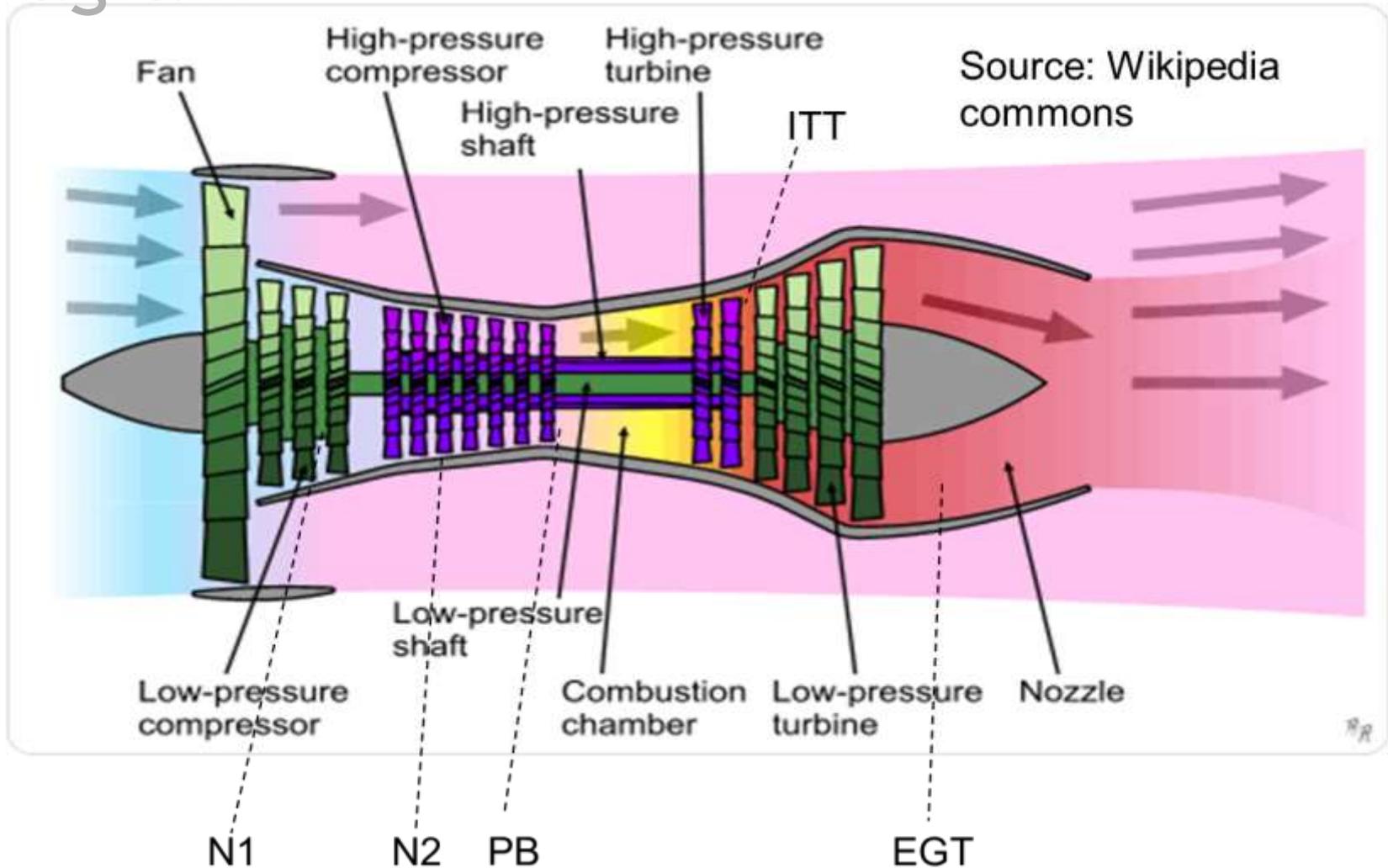
## Overview

- Specific sequence of events are needed to start a gas turbine engine.
  - Turbine Rotation established by Air Starter Subsystem - driving generator for power and pressure for pumps
  - Fuel flow is enabled.
  - Proper Fuel / Air mix is established in combustion chamber.
  - Electrical spark from Ignition Subsystem starts combustion.
  - Conditions monitored for automatic restart if necessary.
  - Controlled ramp increases fuel flow per schedule to achieve stable idle
- Air framer specified switch semantics need to be rationalized with standard signals and start sequence for product family components.

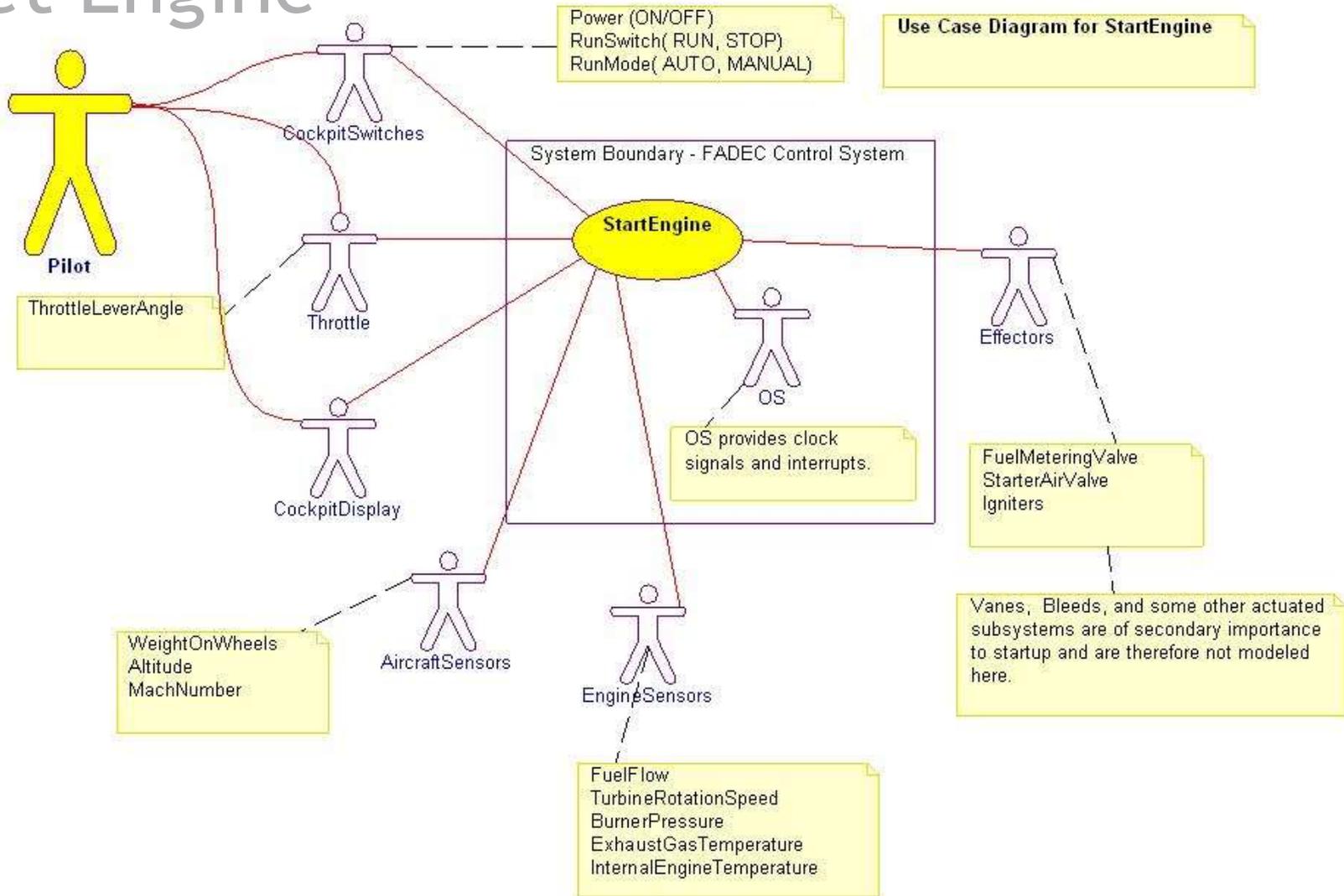


# Generic 2 Spool Gas Turbine Engine

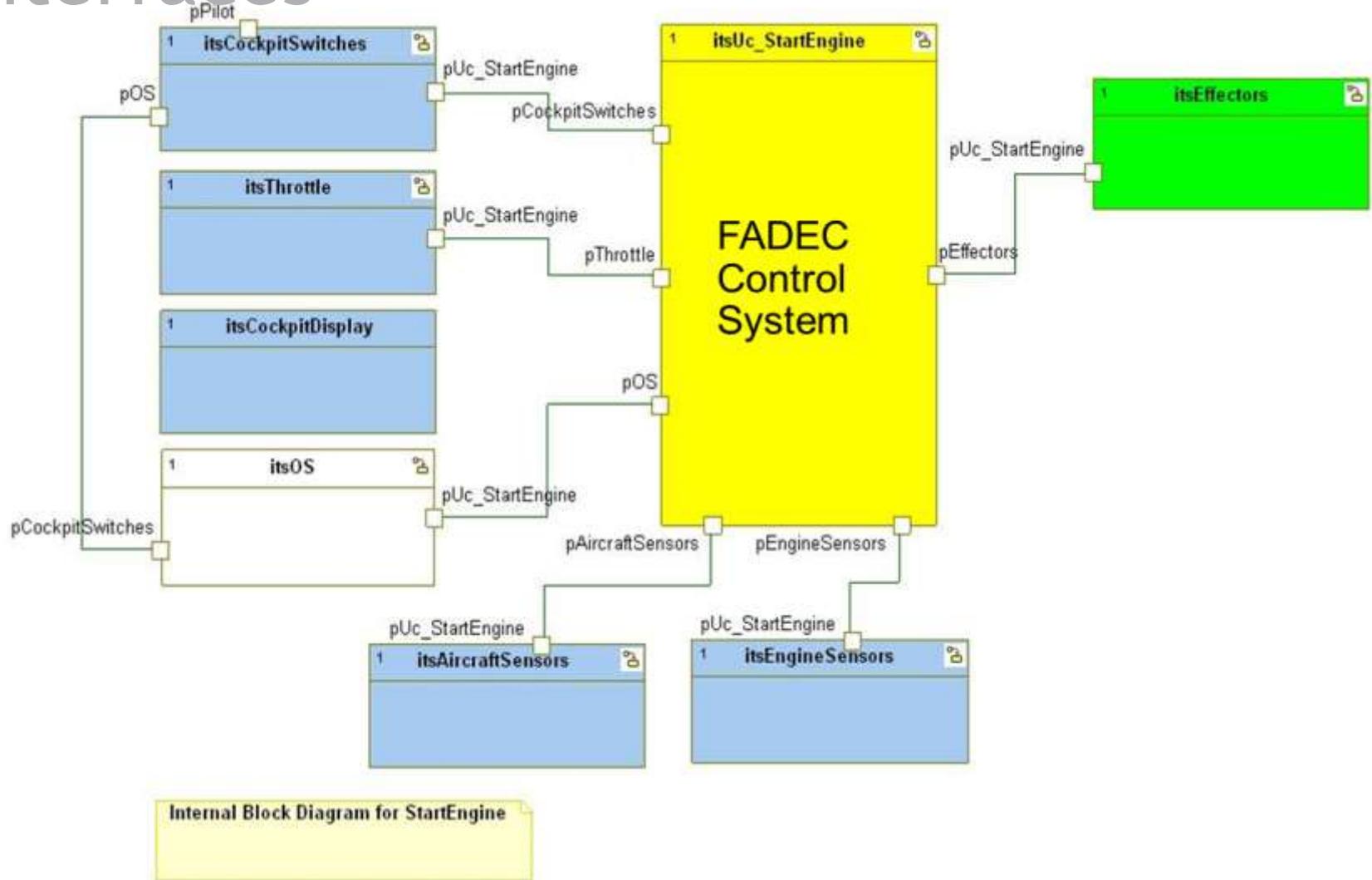
## Diagram



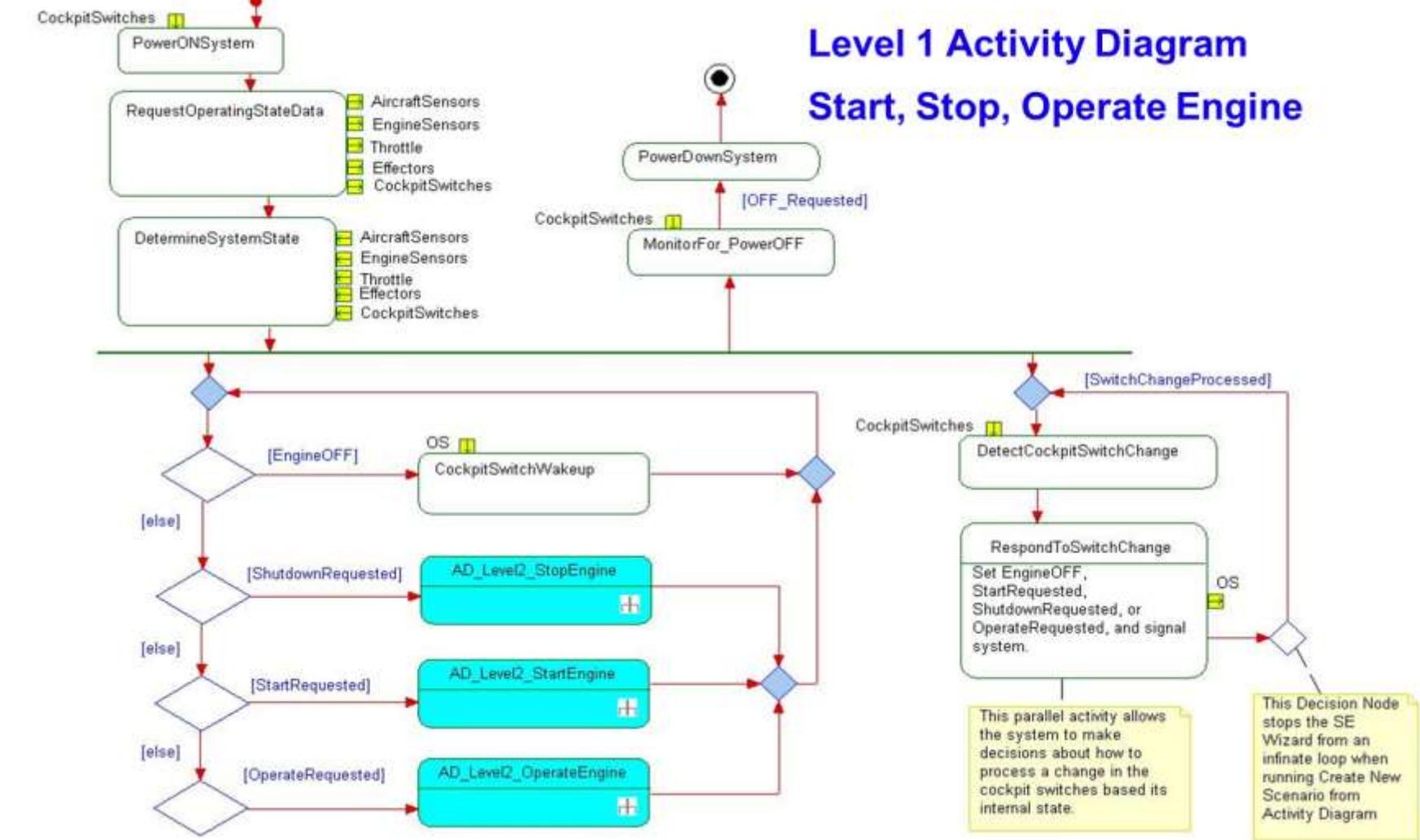
# Use Case Diagram for “How to Start Jet Engine”



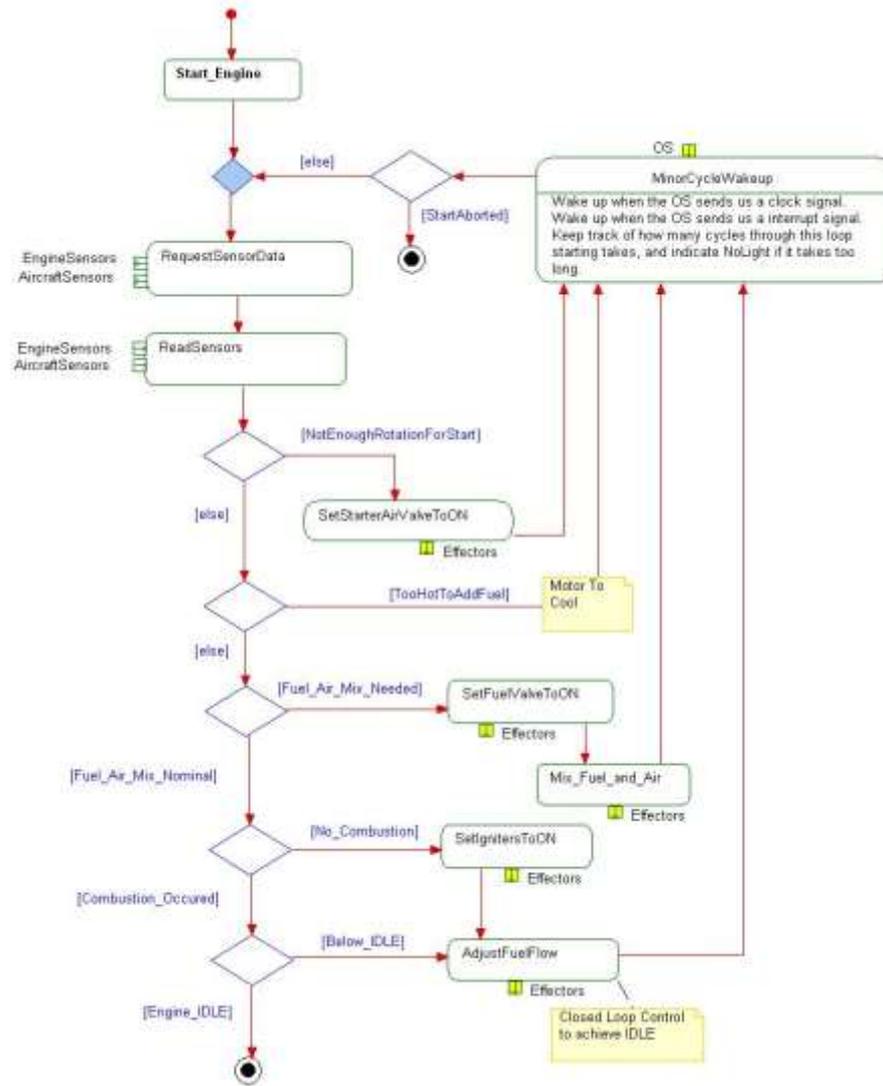
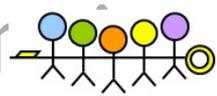
# Internal Block Diagram - formal system interfaces



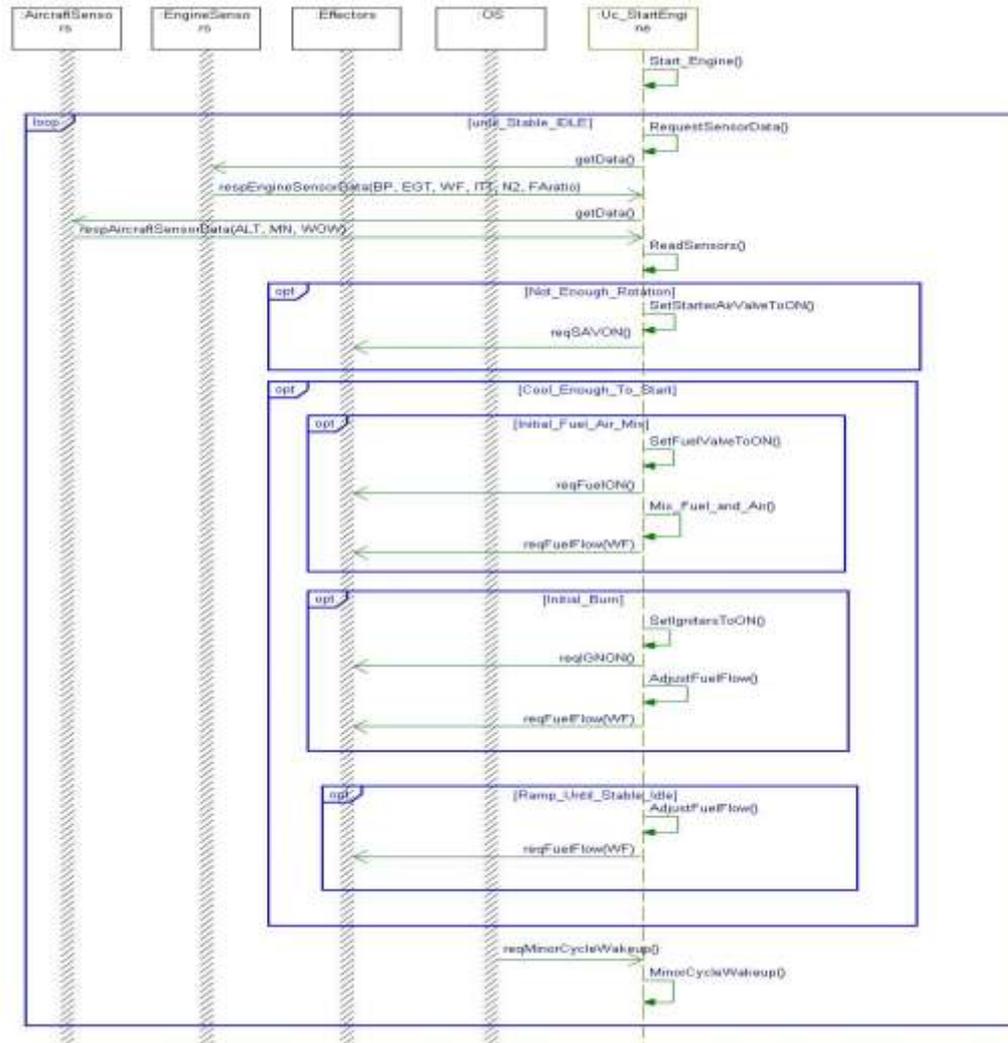
# Functional Modeling - Activity Diagram



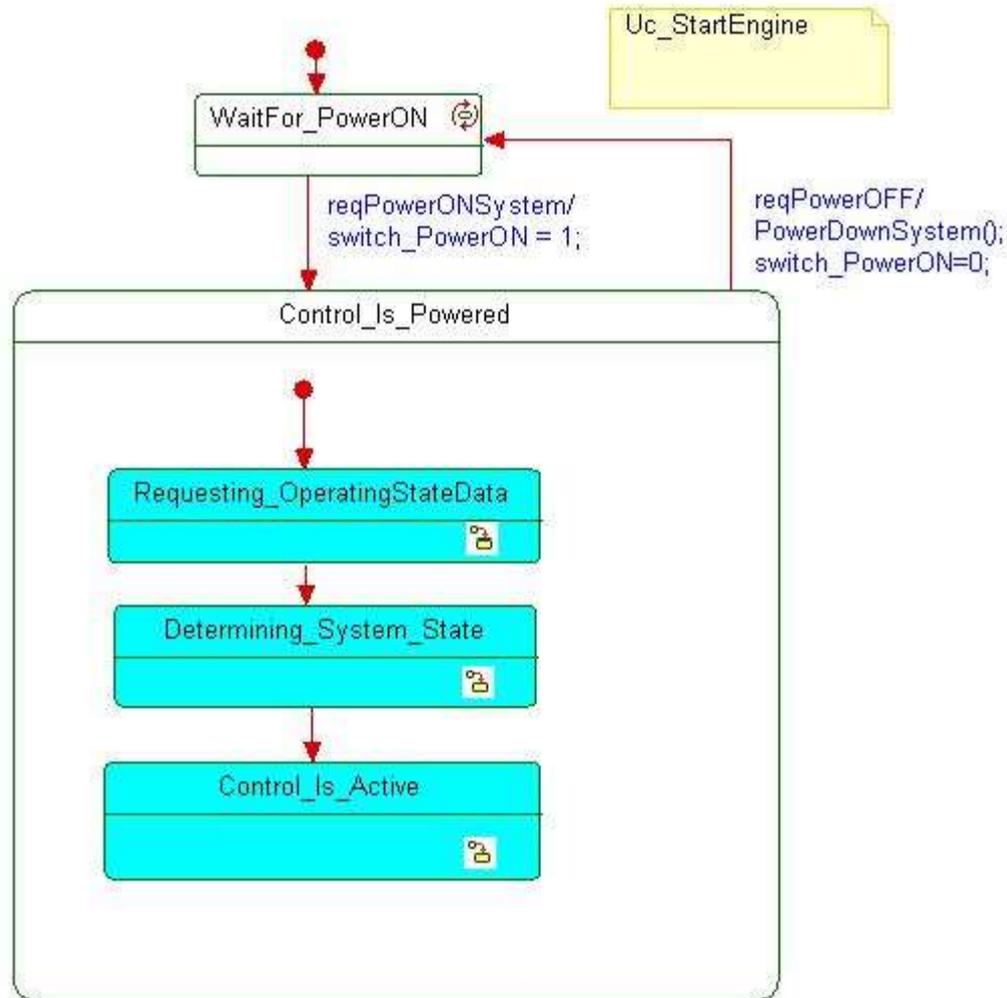
# Level 2 Activity Diagram - Start Engine



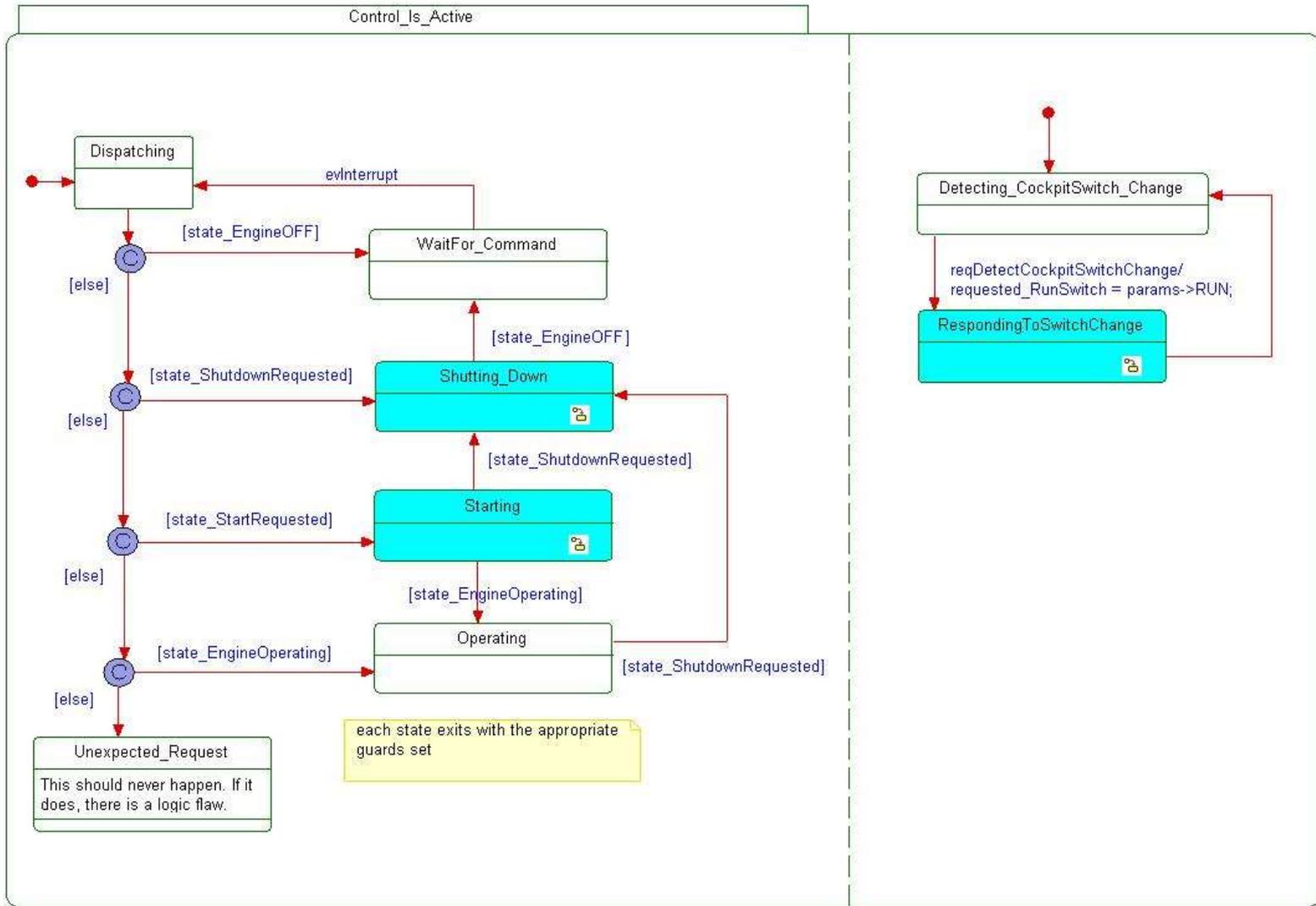
# Sequence Diagram - sequence and content of interactions



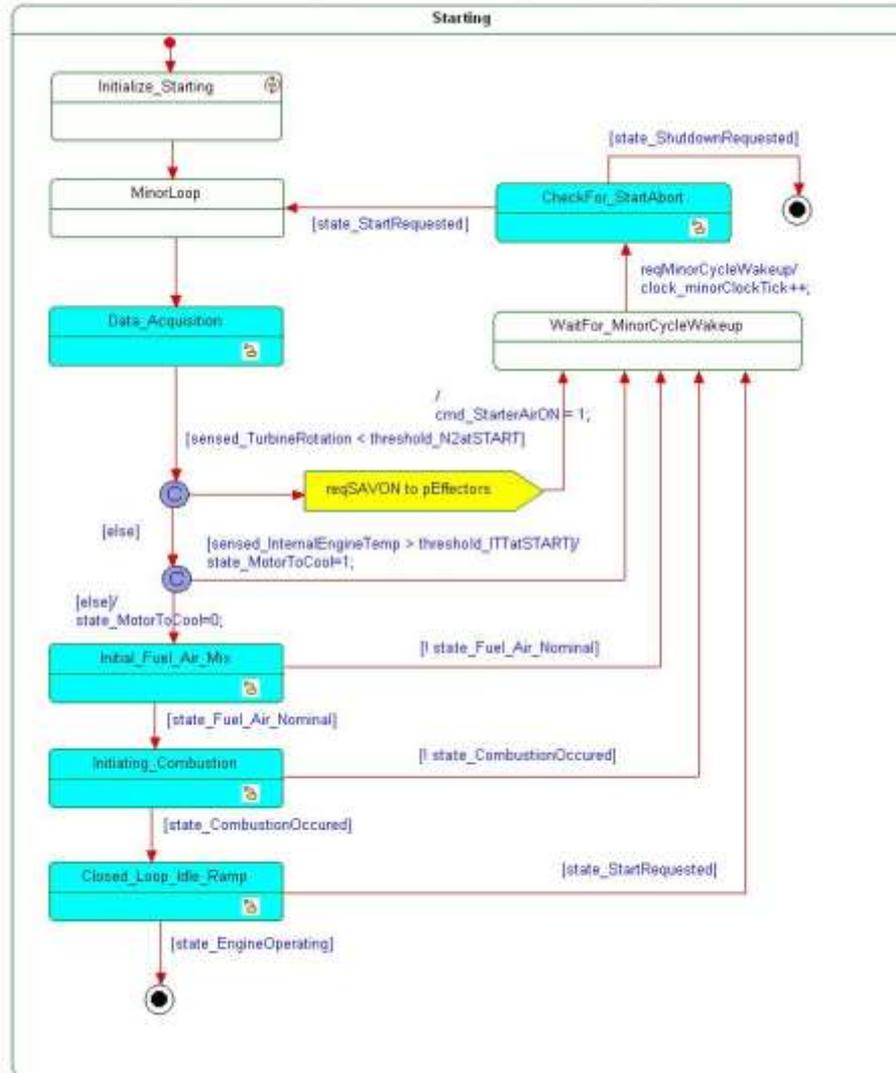
# State Diagram - states, modes, de



# Details of “Control\_Is\_Active”



# Details of Engine Startup



# So... Where are the Requirements?

- In this example, we are starting to model first from concept-of-operations information.
- So... the model becomes the requirements!
- The model then guides the writing of the requirements document.
- Key model elements - activities, dialogs, states, will trace to explicit requirements paragraphs.



# Next Steps

- After creating an executable model, and writing requirements based on the model, the next step is to create formal test sequences from the model.
- One way to do this is to create another actor, a “test sequencer” actor, and connect this actor to the external actors of the model. The test sequencer drives particular tests by setting actor states.
- The Rhapsody animated sequence diagram feature then captures the results of the test, and can be compared against a baseline.
- Book keep your work, linking requirement to test. Rhapsody has features for this.



- We talked a little about the nature and importance of system requirements.
- We showed a workflow that systematically creates the necessary content for an executable model.
- We showed how an executable model can be used to quickly validate requirements, quickly demonstrating a working system that represents the requirements.
- The model and its pictures provide a system scope and framework that makes it easier to write the formal requirements.
- The model provides a way to quickly generate formal test sequences. These provide the essential content for detailed testing in other system testing environments.



