# Model-Based Systems Engineering (MBSE) with SysML

*An Approach for Reducing Cost and Improving Quality*

Presenter:

Lenny Delligatti, M.S., OCSMP Advanced

Last modified: 19 JUN 2012

## Presentation Objectives:

1. Provide an overview of the MBSE approach
2. Provide an overview of the Systems Modeling Language (SysML)
3. Provide evidence of the ROI that MBSE offers over the traditional document-based approach

## *Overview of MBSE*

## Overview of MBSE:

“MBSE is 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.”

-- International Council on Systems Engineering (INCOSE),
*Systems Engineering Vision 2020,* Sep 2007

## Overview of MBSE:

MBSE is *not* a process; it does not replace our existing process sets.

MBSE can be adopted by any organization to develop and maintain systems with higher quality and reduced cost while fully complying with existing process sets (e.g. ISO, CMMI, etc.).

## Overview of MBSE:

### *The 3 Pillars of MBSE*

1) Modeling *languages*

2) Modeling *methods*

3) Modeling *tools*

# Overview of MBSE:

## *Modeling Languages*

The Object Management Group (OMG) has adopted many modeling language specifications (i.e. grammars) that have become de facto standards for the various engineering domains:

- *Unified Profile for DoDAF and MoDAF (UPDM)*
- *Systems Modeling Language (SysML)*
- *Unified Modeling Language (UML)*
- *Modeling and Analysis of Real-Time Embedded systems (MARTE)*
- *Business Process Modeling Notation (BPMN)*

...and many more.

# Overview of MBSE:

## *Modeling Methods*

Examples of well-defined modeling methods:

- INCOSE *Object-Oriented Systems Engineering Method (OOSEM)*
- *JPL State Analysis (SA)* method
- Lockheed Martin *Model-based Systems Engineering Method (MSEM)*
- IBM Telelogic *Harmony-SE* method
- IBM *Rational Unified Process for Systems Engineering (RUP SE)*
- Weilkiens' *System Modeling (SYSMOD)* method

*Caveat*: no out-of-the-box method will meet a project's specific needs and goals. Planning for a model-based project requires customization of a method.

## Overview of MBSE:

### *Modeling Tools*

Examples of some widely used commercial-grade modeling tools:

- IBM *Rhapsody*
- NoMagic *MagicDraw*
- Atego *Artisan Studio*
- Altova *Umodel*
- Sparx Systems *Enterprise Architect*

Each tool complies with OMG's language specifications to varying degrees of fidelity.  Some tools offer greater capabilities than others (e.g. autogeneration of code, reverse engineering of code into a model). Selecting a tool must be part of the project planning process.

## Overview of MBSE:
### *The System Model*

The output of MBSE activities is a coherent and consistent model of the system (i.e. *system model*) expressed in a standard modeling language (e.g. SysML).

The system model is:

1) a set of elements (which represent the system's parts, actors, behaviors, events, requirements, constraints, and test cases),

and

2) the relationships between those elements (e.g. associations, dependencies, generalizations, flows, etc.).

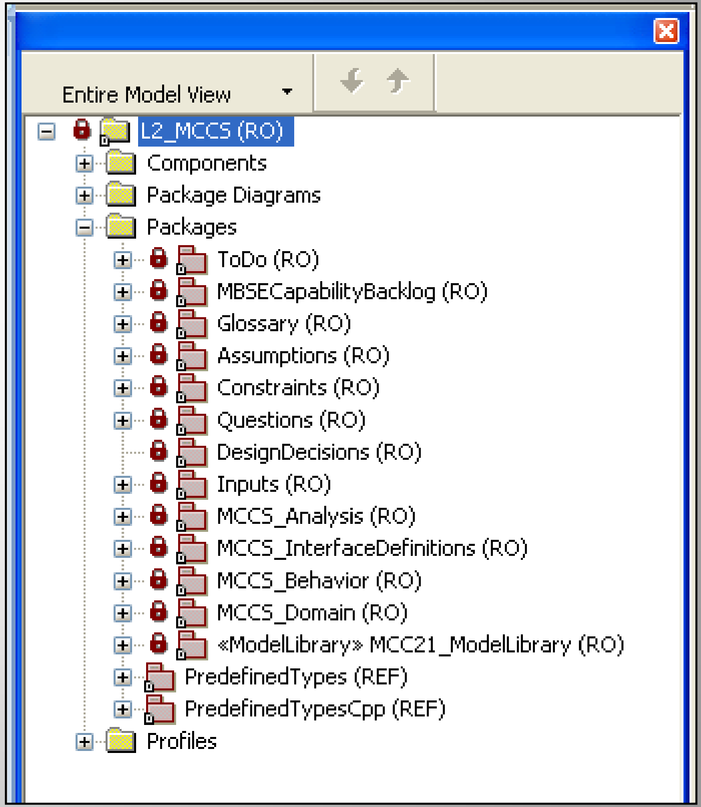
# Overview of MBSE:

## *The System Model*

The system model is created with a modeling tool, like Rhapsody, and stored in a *model repository* (i.e. a set of data files outputted from the tool).

A model user can navigate that model repository in the tool’s *model browser*.

For example, this is a view of the top-level set of packages in a system model as seen in the Rhapsody model browser.



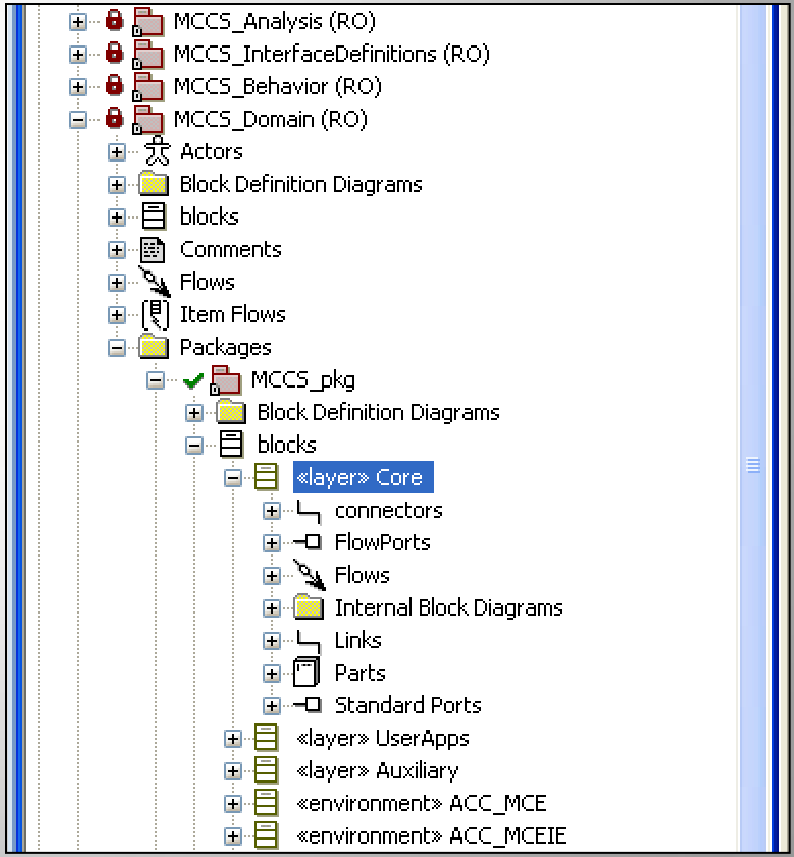
Entire Model View

- L2_MCCS (RO)
  - Components
  - Package Diagrams
  - Packages
    - ToDo (RO)
    - MBSECapabilityBacklog (RO)
    - Glossary (RO)
    - Assumptions (RO)
    - Constraints (RO)
    - Questions (RO)
    - DesignDecisions (RO)
    - Inputs (RO)
    - MCCS_Analysis (RO)
    - MCCS_InterfaceDefinitions (RO)
    - MCCS_Behavior (RO)
    - MCCS_Domain (RO)
    - «ModelLibrary» MCC21_ModelLibrary (RO)
    - PredefinedTypes (REF)
    - PredefinedTypesCpp (REF)
  - Profiles

# Overview of MBSE:

## *The System Model*

A model user can navigate the model to view a specific element (e.g. the SysML block representing the Core layer) and the relationships that it has with other model elements (e.g. connectors, flows, compositions, etc.)



```
MCCS_Analysis (RO)
MCCS_InterfaceDefinitions (RO)
MCCS_Behavior (RO)
MCCS_Domain (RO)
    Actors
    Block Definition Diagrams
    blocks
    Comments
    Flows
    Item Flows
    Packages
        MCCS_pkg
            Block Definition Diagrams
            blocks
                «layer» Core
                    connectors
                    FlowPorts
                    Flows
                    Internal Block Diagrams
                    Links
                    Parts
                    Standard Ports
                «layer» UserApps
                «layer» Auxiliary
                «environment» ACC_MCE
                «environment» ACC_MCEIE
```

# Overview of MBSE:

## *The Model vs. Diagrams of the Model*

In the *document-based* engineering approach, designers create a set of diagrams that serve as views of the system design. These diagrams are even casually referred to as "models" of the system.

In the *model-based* engineering approach, the system model is itself an engineering artifact that has an existence independent of any diagrams that a designer may create to provide views of the model for various stakeholders.

This distinction leads to the Fundamental Precept of MBSE:

***A diagram of the model is never the model itself. It is merely one view of the model.***

# Overview of MBSE:

## *The Model vs. Diagrams of the Model*

As an MBSE practitioner builds the system model (in the model repository), he or she will *also* create diagrams to present specific views of the model.

Each diagram is created for a specific *purpose* to address the specific *concerns* of a specific set of *stakeholders*. Each diagram presents at best a subset of the features specified in the model. Features that are not relevant to the stated diagram purpose should be elided.

This leads to the corollary to the Fundamental Precept of MBSE:

***You cannot conclude that a feature doesn't exist by its absence on a diagram. It may be shown in another view of the model…or in no view at all.***

# Overview of MBSE:

## *The Model vs. Diagrams of the Model*

A quick metaphor to internalize this idea:

*The model is a mountain.*

*A diagram is a picture of the mountain.*

# Overview of MBSE:

## *The Power of a Single Model Repository*

This distinction between *model* and *diagram* is the root of the extraordinary ROI that the model-based approach offers over the document-based approach.

*In the model-based approach:* if a requirement changes, the designer goes to a single location—the model repository—to make the necessary changes to an element. And the modeling tool automatically updates *all* diagrams where that element appears. There is no opportunity for incorrect data entry. Consistency across all views is ensured.

*In the document-based approach*, the designer has to know a priori which diagrams are impacted by the change and where those diagrams are located. He or she then needs to open each one sequentially, and then manually make the same change repeatedly in multiple locations—a process which is time-consuming and error-prone.

# Overview of MBSE:

## *System Model as Primary Artifact*

Some customers—in particular, government agencies—are often constrained by legislation or regulation to follow a document-based engineering approach.  Documents must be formally reviewed, approved, and then archived.

In the MBSE approach, the system model serves as the primary artifact from which all other required engineering artifacts can be autogenerated via the modeling tool and its accessories:

- Requirements Specs.
- Interface Definition Documents (IDDs)
- System and Software Design Specs.
- Engineering Analysis Specifications and Results
- Test Case Specifications and As-Run Results

# Overview of MBSE:

## *Language Transformations: Model to Source Code*

The ultimate capability for a model-based engineering organization to achieve is the automated transformation of a SysML or UML model into production-quality source code.

This capability allows a design team to use SysML and UML as a higher-level programming language while simultaneously building and evolving a system model, which provides system maintainers:

- a visualization of system structure and behavior, allowing for faster understanding and lower maintenance costs
- a mechanism for performing automated downstream impact analysis when the customer submits change requests throughout the maintenance stage of the lifecycle
- automated, consistent updates of all autogenerated artifacts when changes are made in the model repository

## *Overview of SysML*

# Overview of SysML:

## *What it is…*

SysML is a graphical modeling language for the systems engineering domain.

The key word is "language." SysML has a grammar and vocabulary just like any of the natural languages we speak in this world (e.g. Hindi, Japanese, English, etc.).

SysML is the language "spoken" by systems engineers to visualize, communicate, and document ideas about a system's requirements, structure, behavior, and constraints.

# Overview of SysML:

## *What it isn't…*

SysML is *not* a methodology.

In *Survey of Model-Based Systems Engineering (MBSE) Methodologies*, Jeff Estefan defines *methodology* as:
"a collection of related processes, methods, and tools."
A *process* defines *what* tasks must be performed. A *method*—used here as a synonym for *procedure*—specifies *how* those tasks should be performed.

With that in mind, I repeat: SysML is *not* a methodology. The language can be used to support many different defined methodologies.

## Overview of SysML:

### *The Origin Story*

The Unified Modeling Language (UML) was adopted by the Object Management Group (OMG) in 1997 as a standard modeling language for the *software* domain.

In 2001 INCOSE decided to adopt UML as a standard modeling language for systems engineering…despite the fact that UML's constructs did not sufficiently capture all of the language concepts that are meaningful in the *systems* domain.

In 2003 the OMG published the *UML for Systems Engineering RFP,* jointly drafted by the OMG and INCOSE, detailing the requirements for a *systems* modeling language.

## Overview of SysML:

### *The Origin Story*

A SysML development team formed to create the initial draft of the SysML specification.  That team was led by Alan Moore, the language architect.

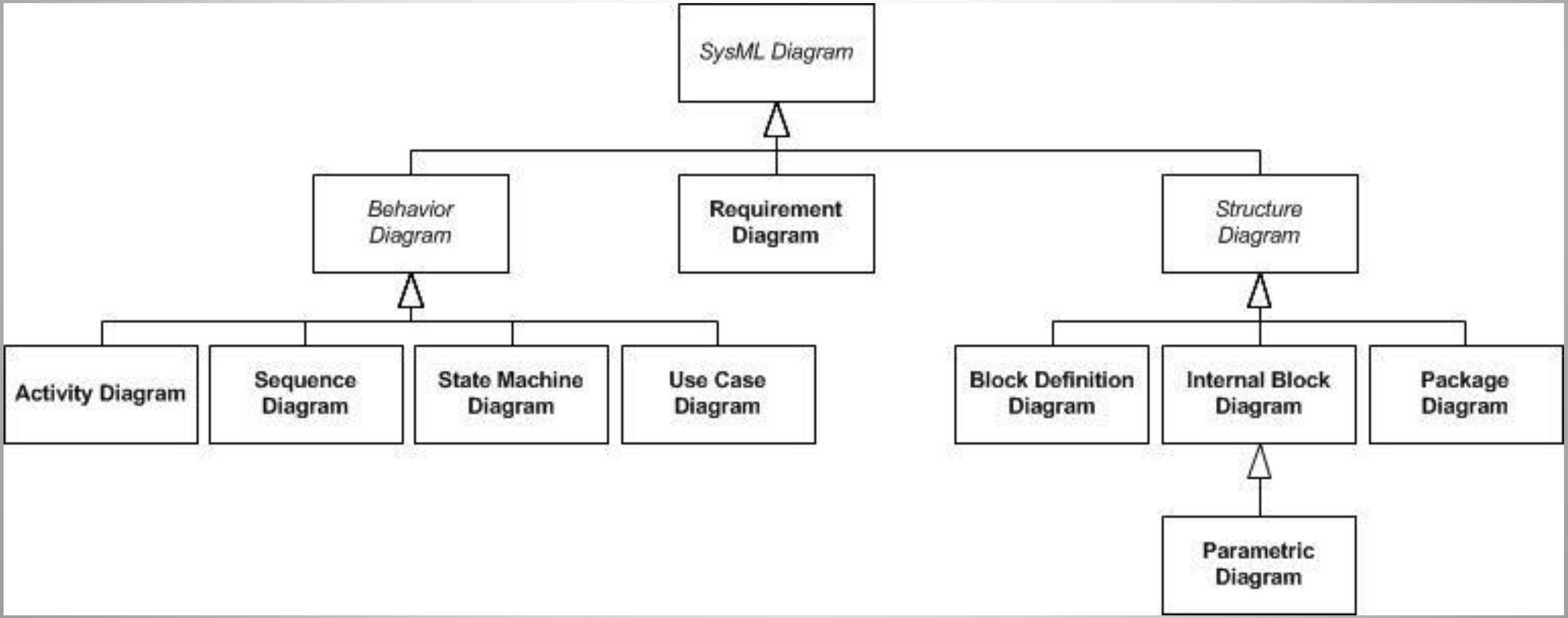
The draft specification was adopted by the OMG in April 2006.

The OMG SysML Finalization Task Force (FTF) then refined the document and published the SysML specification v1.0 in September 2007 for public use.

Today the OMG SysML Revision Task Force (RTF) is currently working on SysML v1.4.

# Overview of SysML:

## *SysML Diagram Taxonomy*



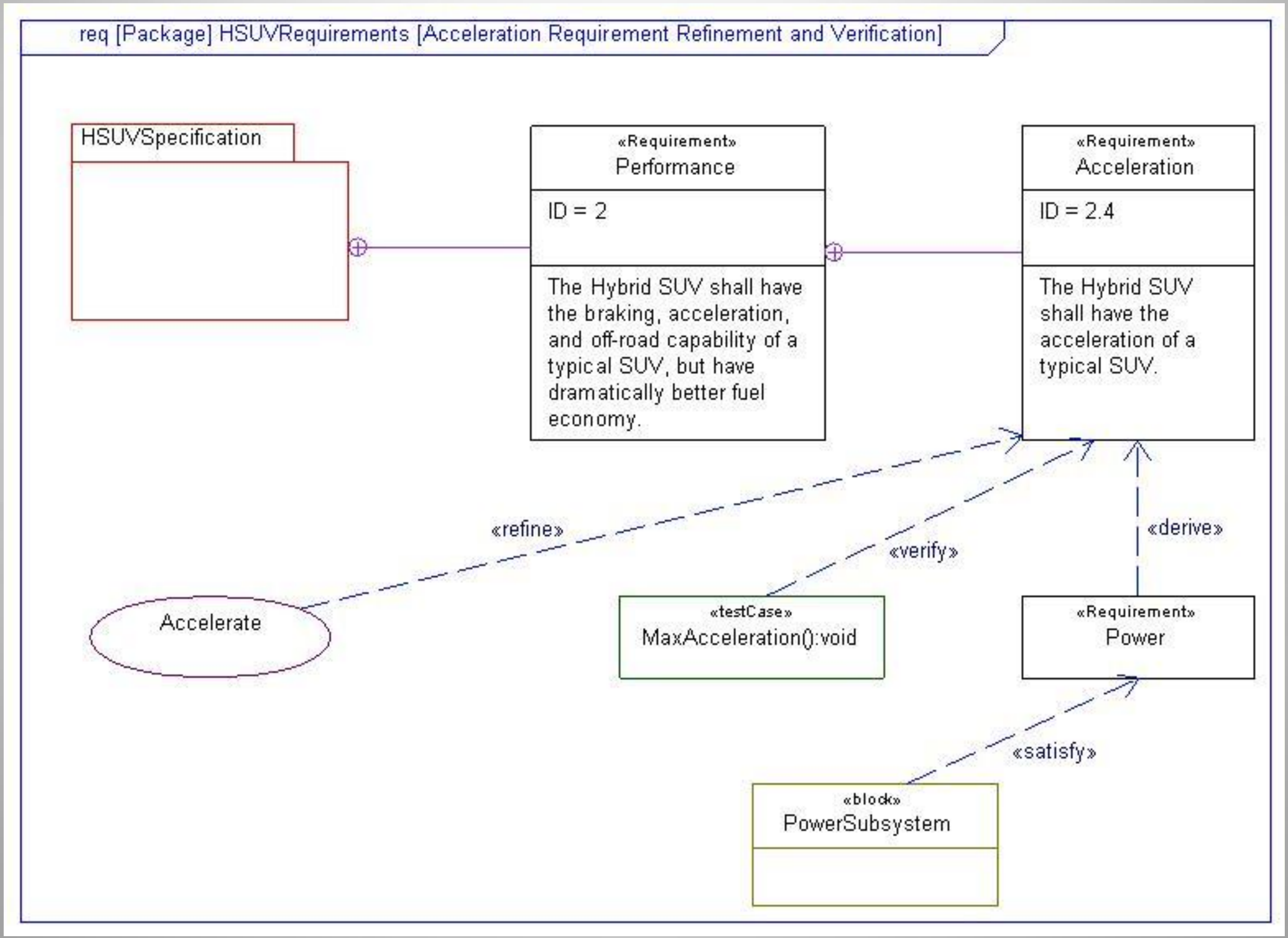
# Overview of SysML:
## *Requirement Diagram*

A *Requirement Diagram* is used to display:

- text-based requirements,
- the relationships between requirements (e.g. *containment*, *derivation*, and *copy* relationships), and
- the relationships between requirements and other model elements (e.g. *satisfaction, verification,* and *refinement* relationships).

# Overview of SysML:

## *A Sample Requirement Diagram*


req [Package] HSUVRequirements [Acceleration Requirement Refinement and Verification]
HSUVSpecification
«Requirement»
Performance
ID = 2
The Hybrid SUV shall have the braking, acceleration, and off-road capability of a typical SUV, but have dramatically better fuel economy.
«Requirement»
Acceleration
ID = 2.4
The Hybrid SUV shall have the acceleration of a typical SUV.
«refine»
«verify»
«derive»
Accelerate
«testCase»
MaxAcceleration():void
«Requirement»
Power
«satisfy»
«block»
PowerSubsystem

# Overview of SysML:

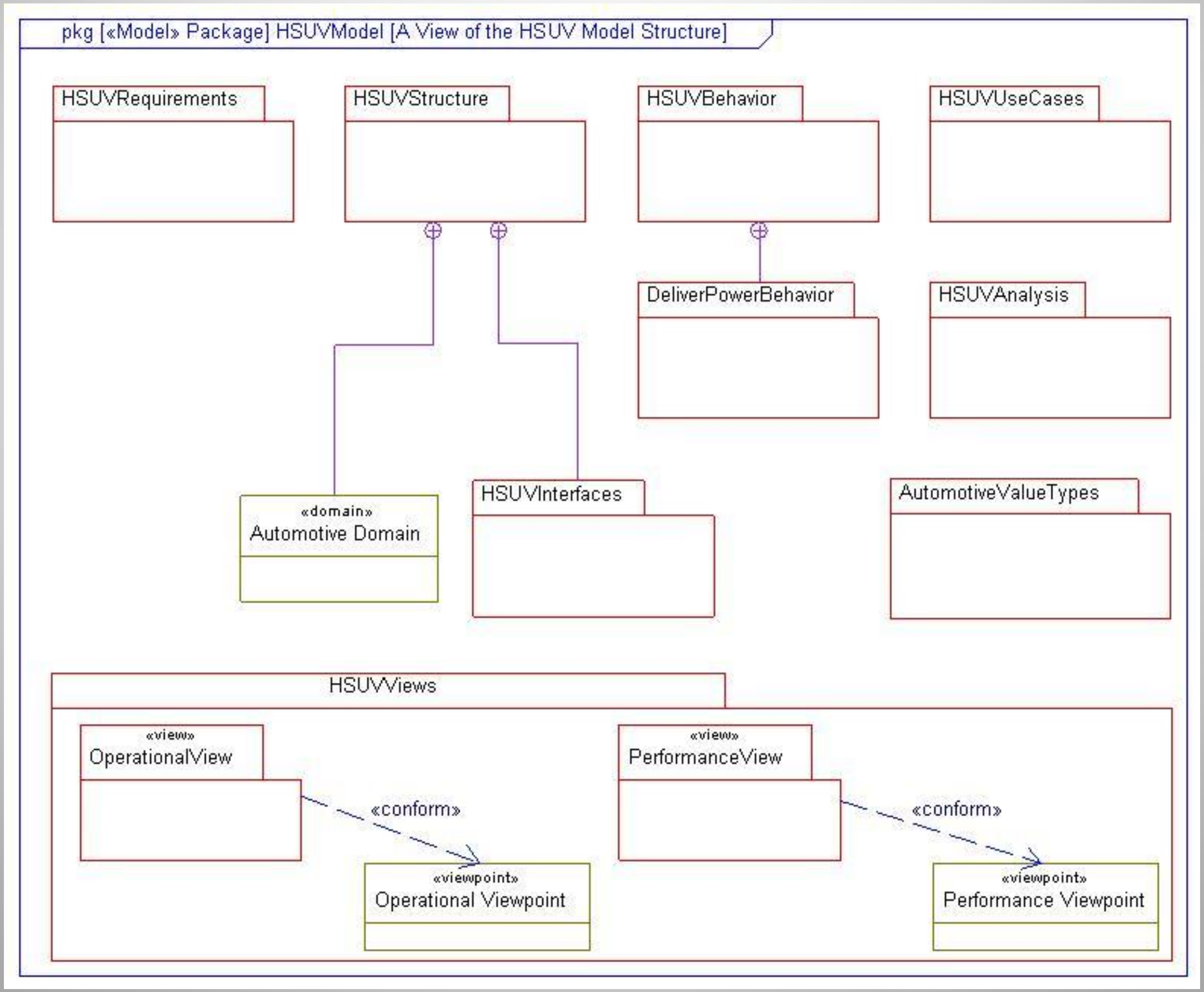
## *Package Diagram*

A *Package Diagram* is used to display how a model is organized in the form of a package containment hierarchy.

A package diagram may also show the model elements that packages contain and the dependencies between packages and their contained model elements.

# Overview of SysML:

## *A Sample Package Diagram*



# Overview of SysML:

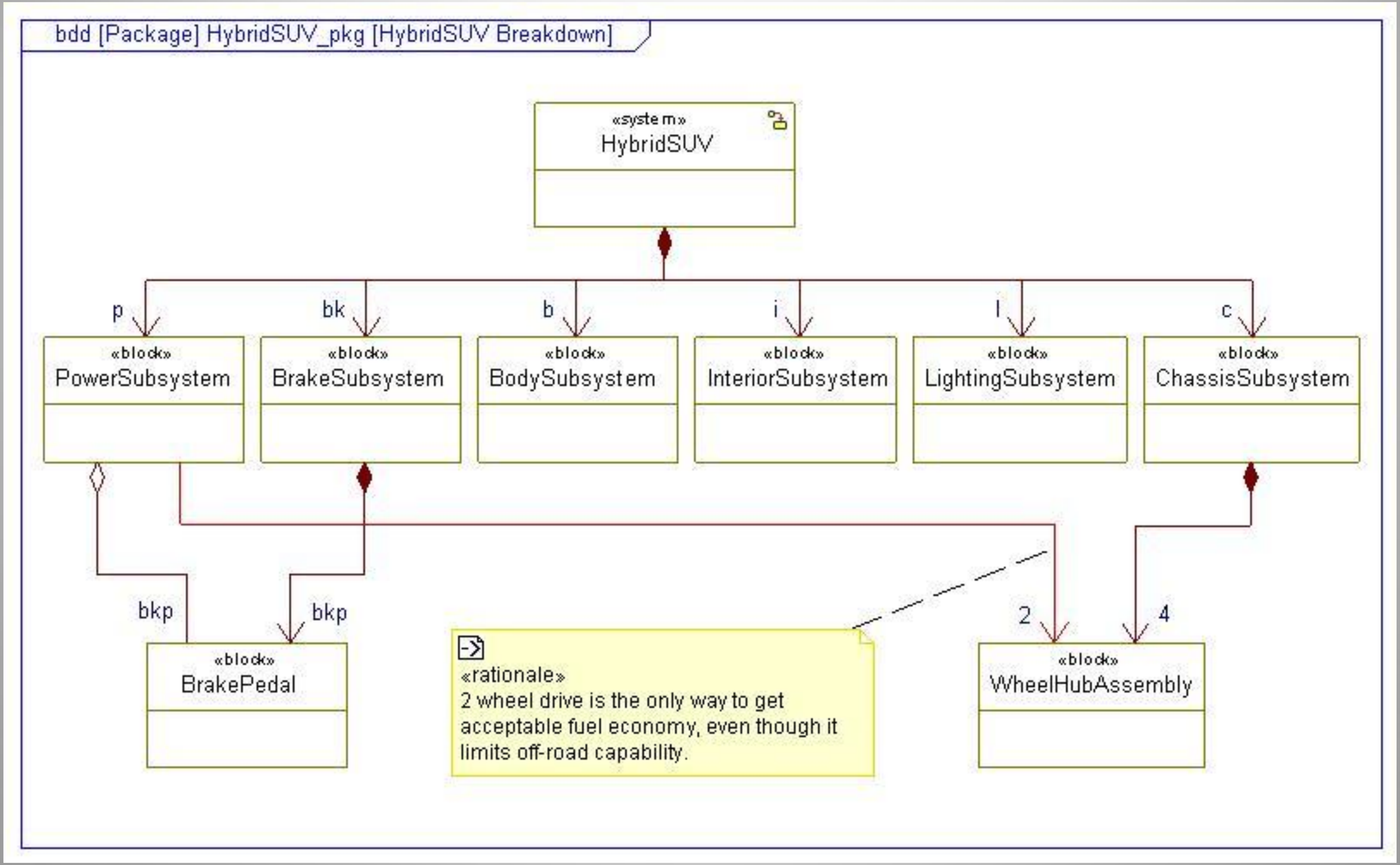
## *Block Definition Diagram (BDD)*

A *Block Definition Diagram (BDD)* is used to display the properties and operations of blocks and the relationships between blocks (i.e. associations, generalizations, and dependencies).

Common uses for a BDD include displaying system hierarchy trees and classification trees.

# Overview of SysML:

## *A Sample BDD*



## Overview of SysML:

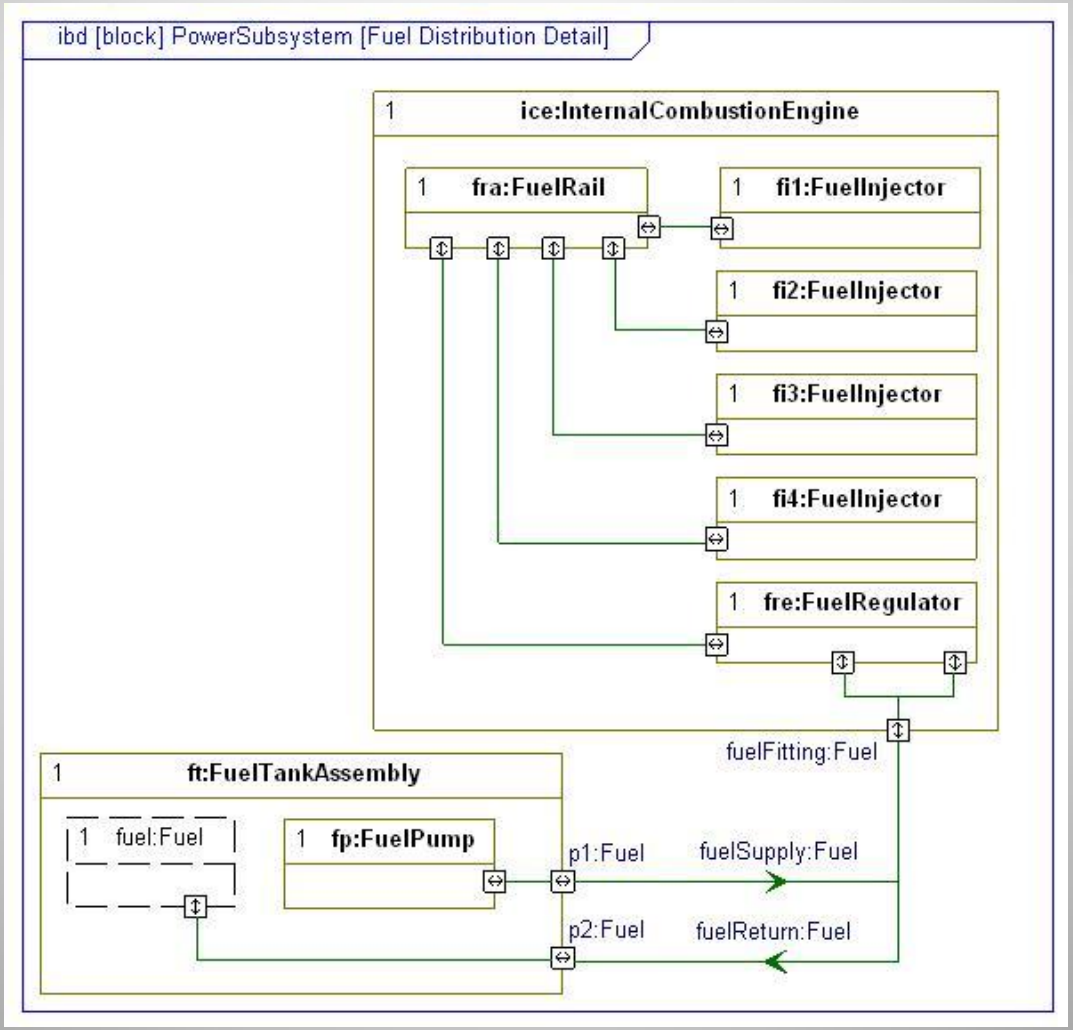
### *Internal Block Diagram (IBD)*

An *Internal Block Diagram (IBD)* is used to capture the internal structure of a block—specifically its part properties and reference properties—and the connectors between those properties.

An IBD always represents the internal structure of a *single* block.

# Overview of SysML:

## *A Sample IBD*



## Overview of SysML:

### *Parametric Diagram*

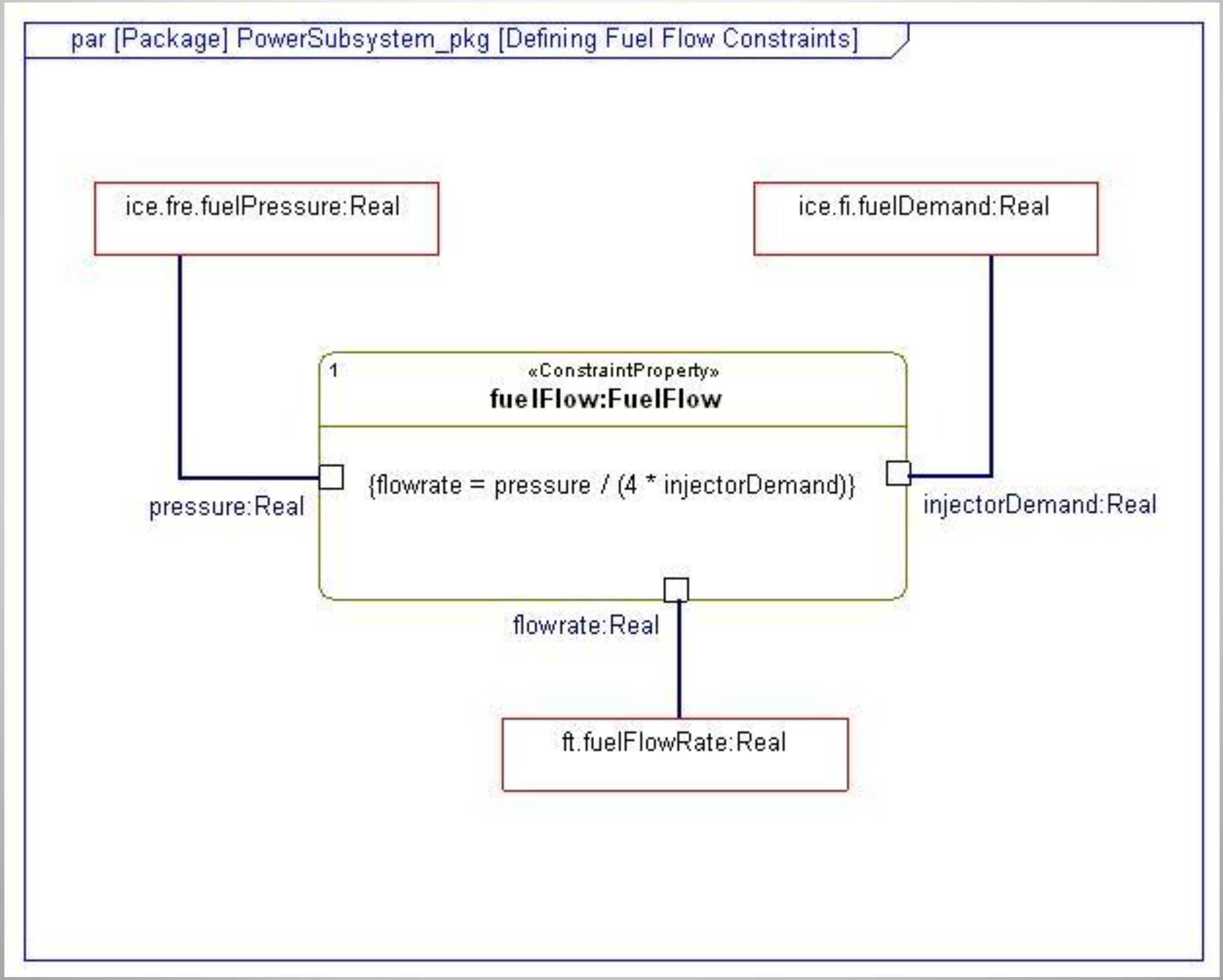
A *Parametric Diagram* is used to express how one or more constraints—expressed as equations and inequalities—are bound to the value properties of a system.

Parametric diagrams support engineering analyses, to include: performance, reliability, availability, power, mass, and cost.

Parametric diagrams can also be used to support trade studies of alternative candidate physical architectures.

# Overview of SysML:

## *A Sample Parametric Diagram*


par [Package] PowerSubsystem_pkg [Defining Fuel Flow Constraints]
ice.fre.fuelPressure:Real
ice.fi.fuelDemand:Real
1
«ConstraintProperty»
fuelFlow:FuelFlow
{flowrate = pressure / (4 * injectorDemand)}
pressure:Real
injectorDemand:Real
flowrate:Real
ft.fuelFlowRate:Real

## Overview of SysML:

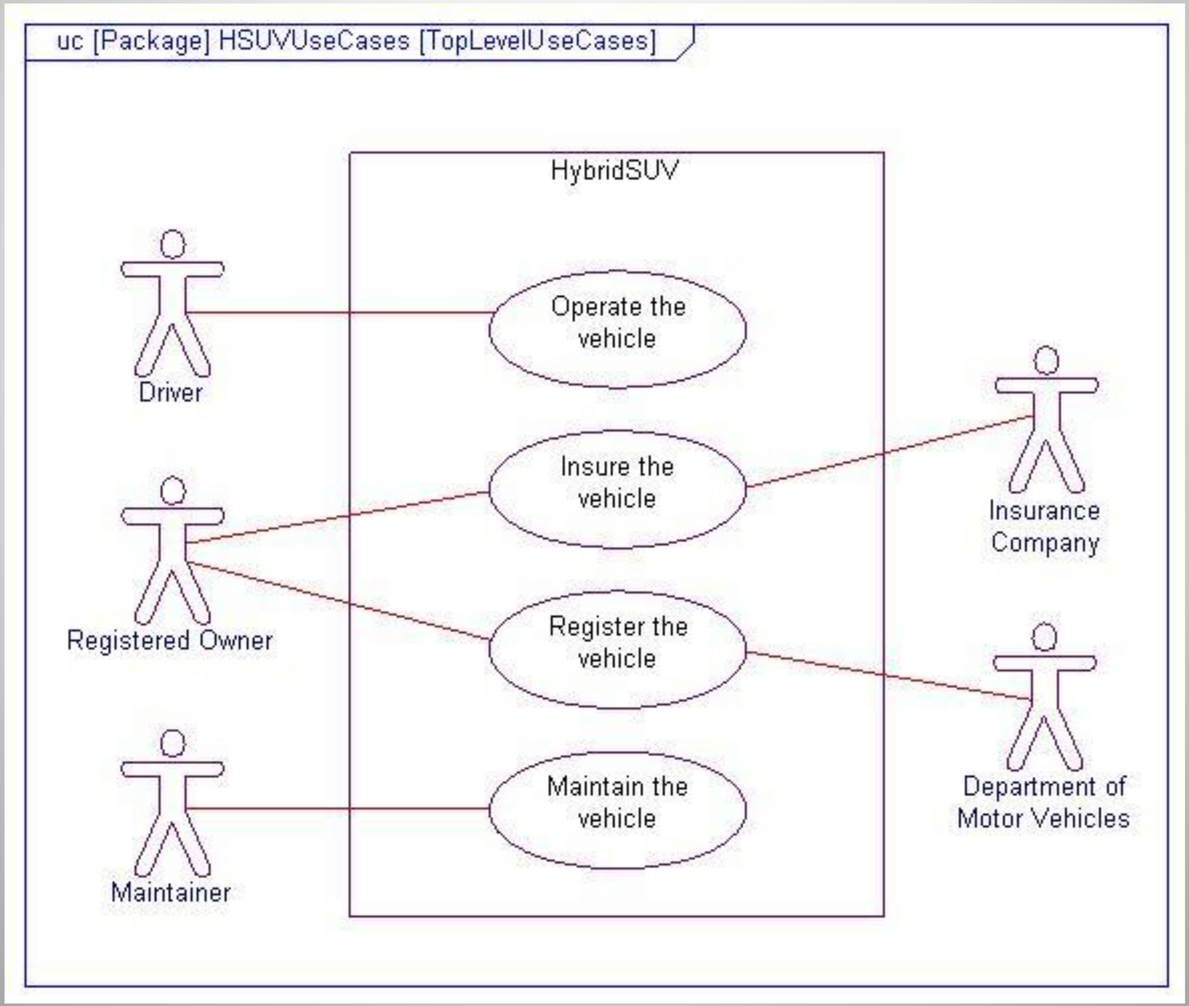
### *Use Case Diagram*

A *Use Case Diagram* is used to convey the use cases that a system performs and the actors that invoke and participate in those use cases.

A use case diagram is a black-box view of the services that a system performs in collaboration with its actors.

# Overview of SysML:

## *A Sample Use Case Diagram*



## Overview of SysML:

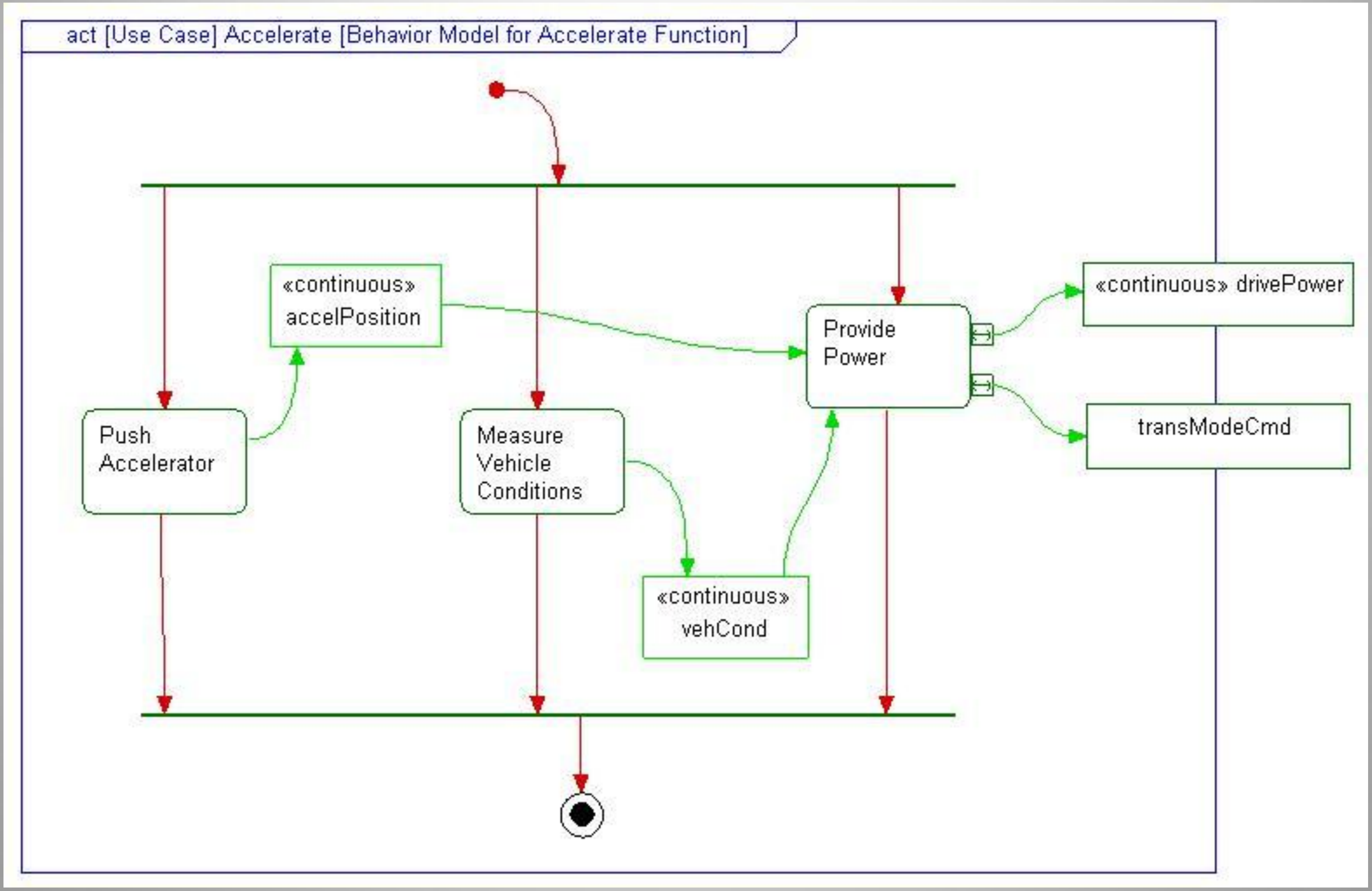
### *Activity Diagram*

An *Activity Diagram* is used to specify a behavior with a focus on the flow of control and the transformation of inputs into outputs through a controlled sequence of actions.

Activity diagrams are commonly used as an *analysis* tool to better understand the problem space and to express the desired behavior of the system-under-design (SuD) within that context.  It is not intended to be a precise specification of a behavior.

# Overview of SysML:

## *A Sample Activity Diagram*


act [Use Case] Accelerate [Behavior Model for Accelerate Function]
«continuous» accelPosition
Push Accelerator
Measure Vehicle Conditions
«continuous» vehCond
Provide Power
«continuous» drivePower
transModeCmd

# Overview of SysML:

## *Sequence Diagram*

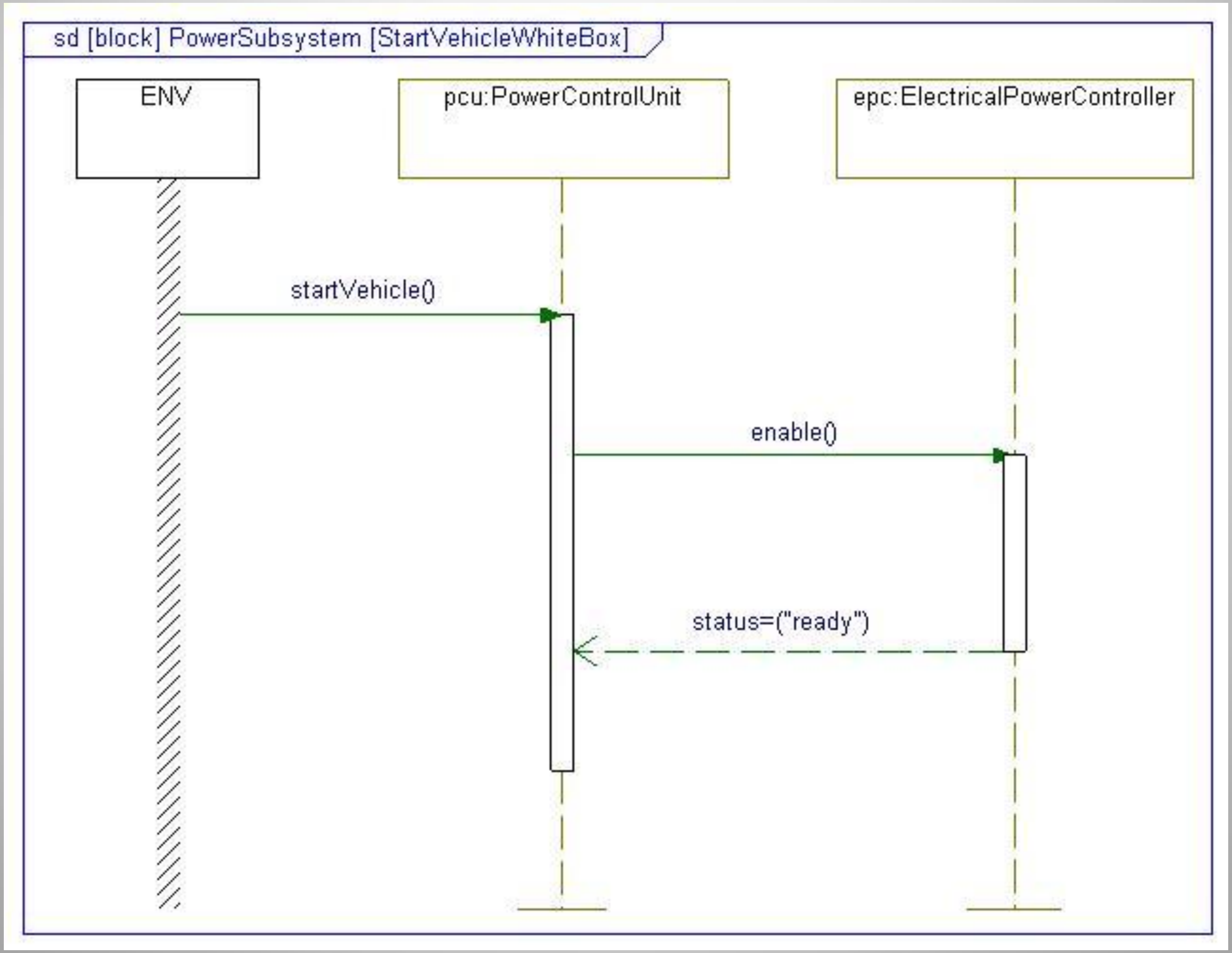
A *Sequence Diagram* is used to specify a behavior with a focus on how the parts of a block interact with one another via operation calls and asynchronous signals.

Sequence diagrams are commonly used as a detailed design tool to precisely specify a behavior as an input to the development stage of the lifecycle.

Sequence diagrams are also an excellent mechanism for specifying test cases.

# Overview of SysML:

## *A Sample Sequence Diagram*


sd [block] PowerSubsystem [StartVehicleWhiteBox]
ENV
pcu:PowerControlUnit
epc:ElectricalPowerController
startVehicle()
enable()
status=("ready")

# Overview of SysML:

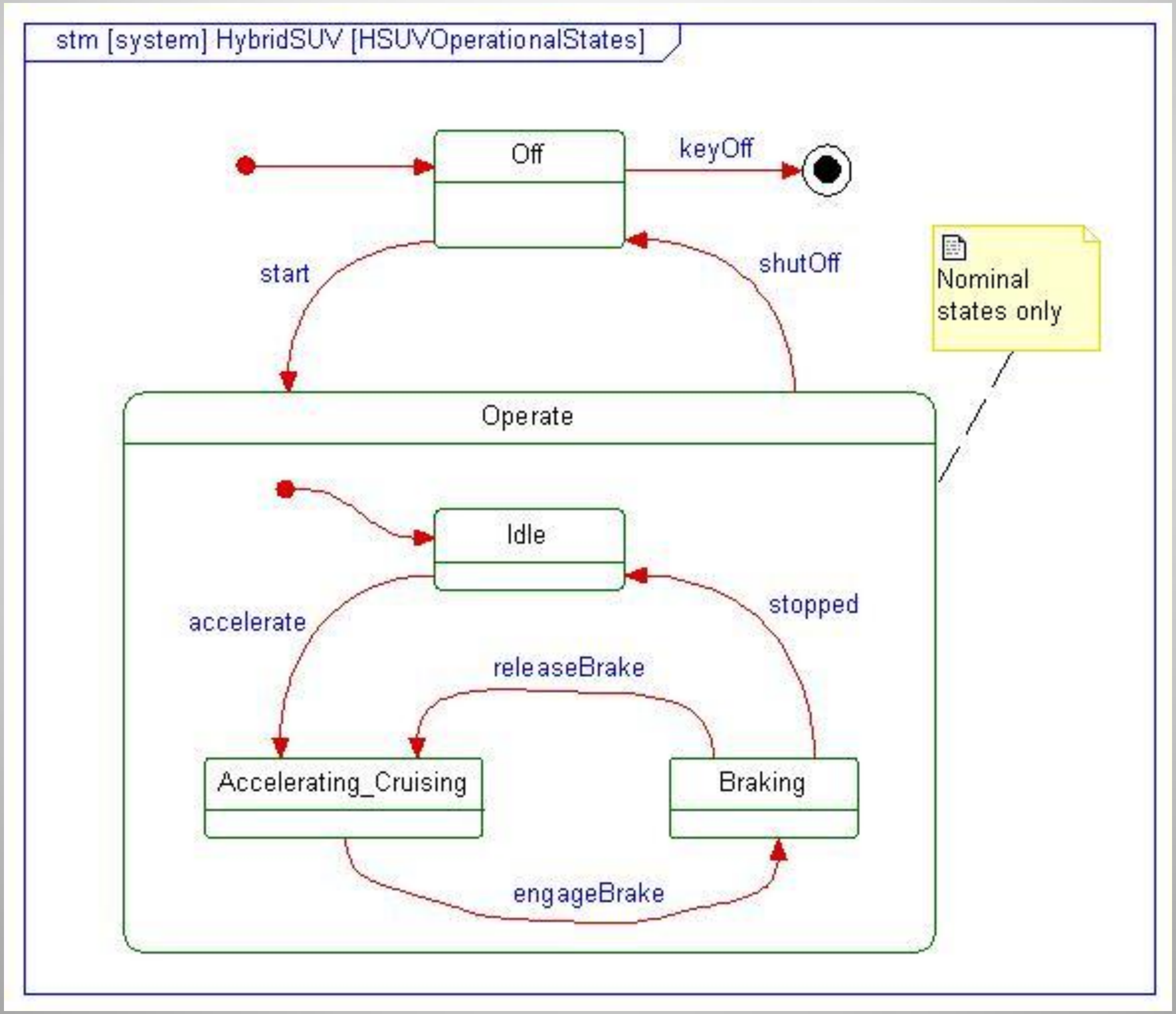
## *State Machine Diagram*

A *State Machine Diagram* is used to specify a behavior with a focus on the set of states of a block and the possible transitions between those states in response to system events.

A state machine diagram, like a sequence diagram, is a precise specification of a block’s behavior that can serve as an input to the development stage of the lifecycle.

# Overview of SysML:

## *A Sample State Machine Diagram*



## *MBSE*
## *Return on Investment*

## MBSE Return on Investment:

### *A Case Study from LM Aeronautics*

The set of slides that follow show the data collected from four real aerospace and defense programs at Lockheed Martin.

Three of the programs used the traditional document-based engineering approach.  One program used the model-based engineering approach.

The actual names of the programs have been omitted to protect the integrity of these ongoing programs.

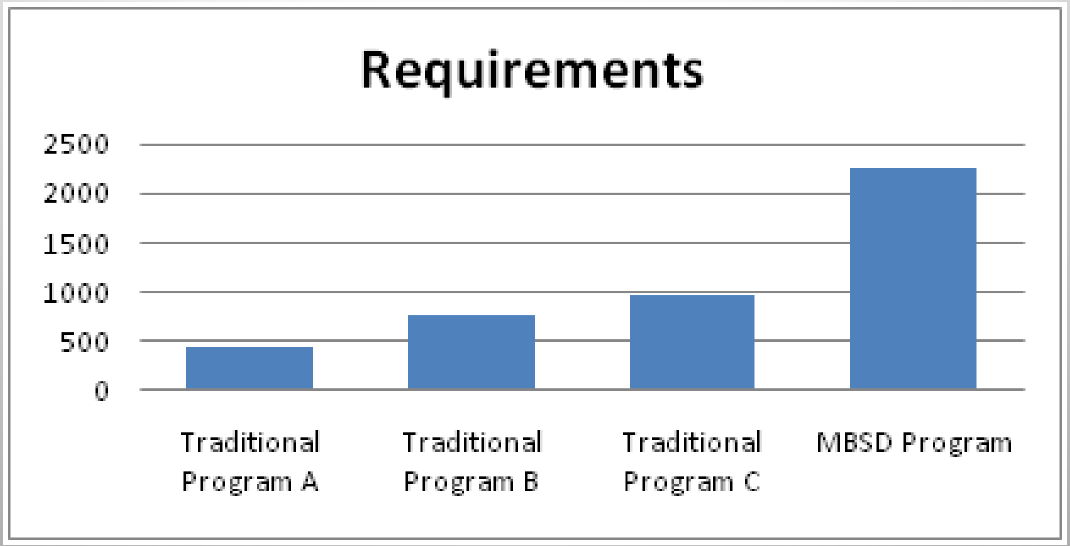
*Note:  these slides use the acronym "MBSD" (Model-Based Systems Development).  Though the industry has not come to consensus on a set of acronyms, "MBSD" is meant to convey a set of model-based activities that span all engineering domains, while "MBSE" is generally used to refer to model-based activities within the systems engineering domain only.*

# MBSE Return on Investment:

## *Requirements Comparison*

This figure compares the number of requirements for these four aeronautics programs.

The program that used the model-based approach had more than twice as many requirements as the next largest program.

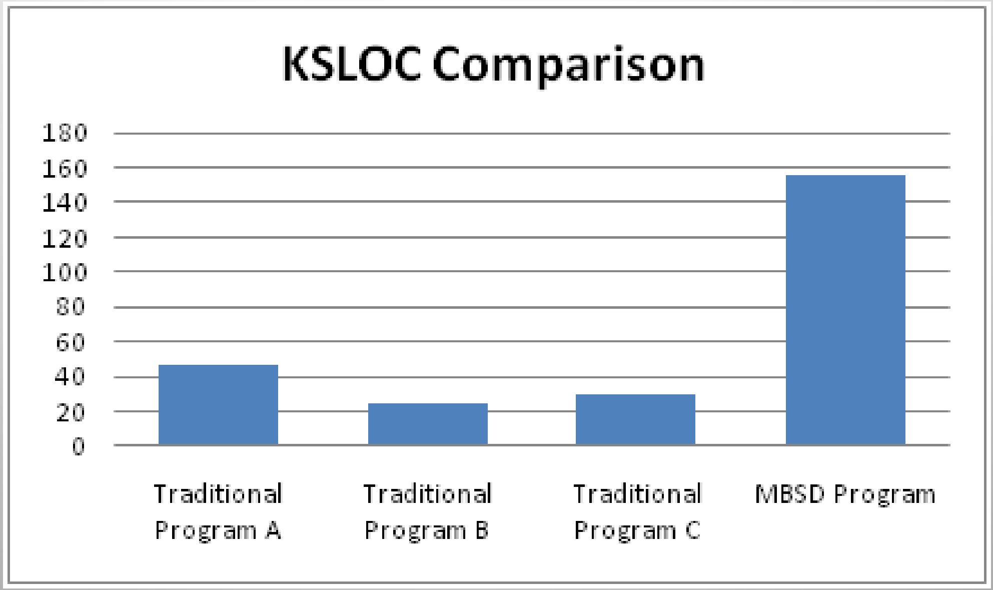


# MBSE Return on Investment:

## *KSLOC Comparison*

This figure compares KSLOC for these four programs as another metric to contrast system size.

The program that used the model-based approach developed a significantly larger system than the other three programs.



# MBSE Return on Investment:

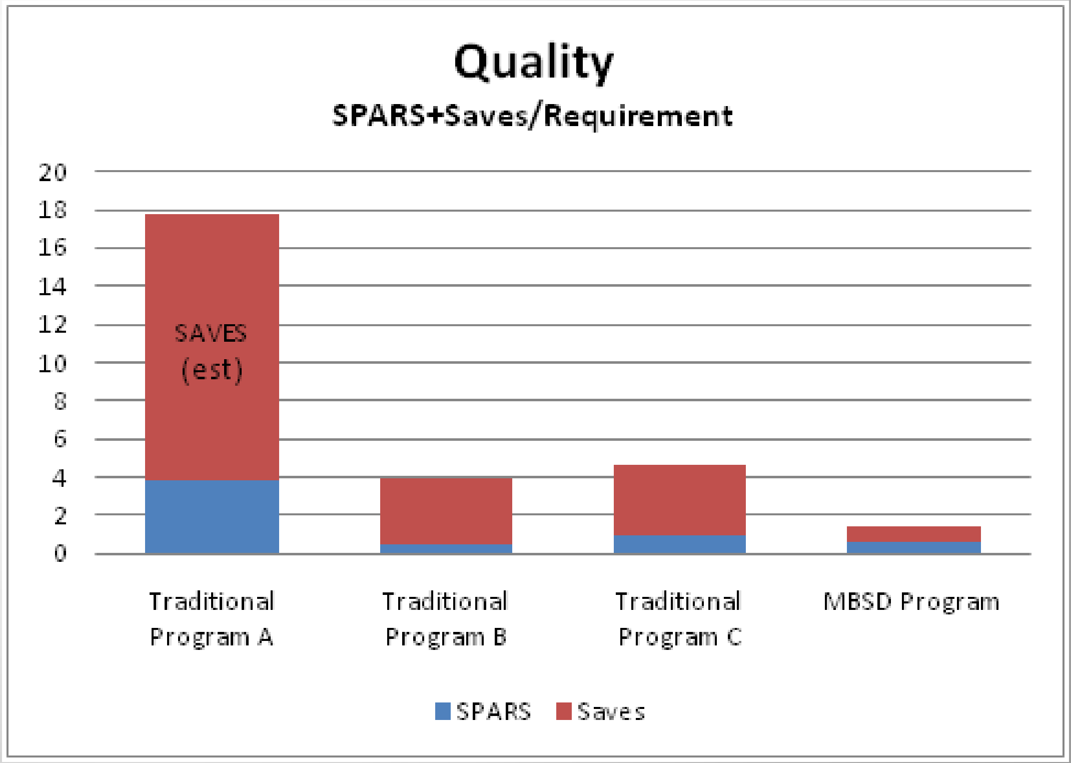
## *Quality Comparison*

This figure compares the number of *defects per requirement* for these four programs.

*Note:*

*"SPAR" refers to a defect caught after release.*

*"Save" refers to a defect caught before release.*

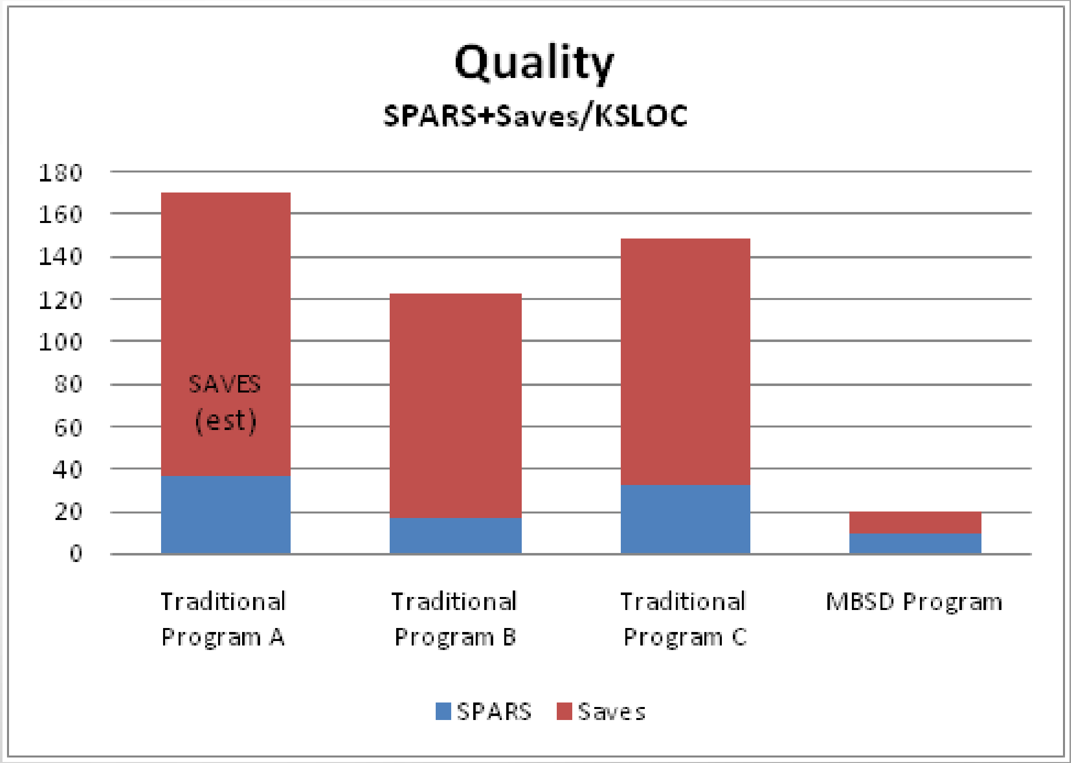


## MBSE Return on Investment:

### *Quality Comparison*

This figure compares the number of *defects per KSLOC* for these four programs.

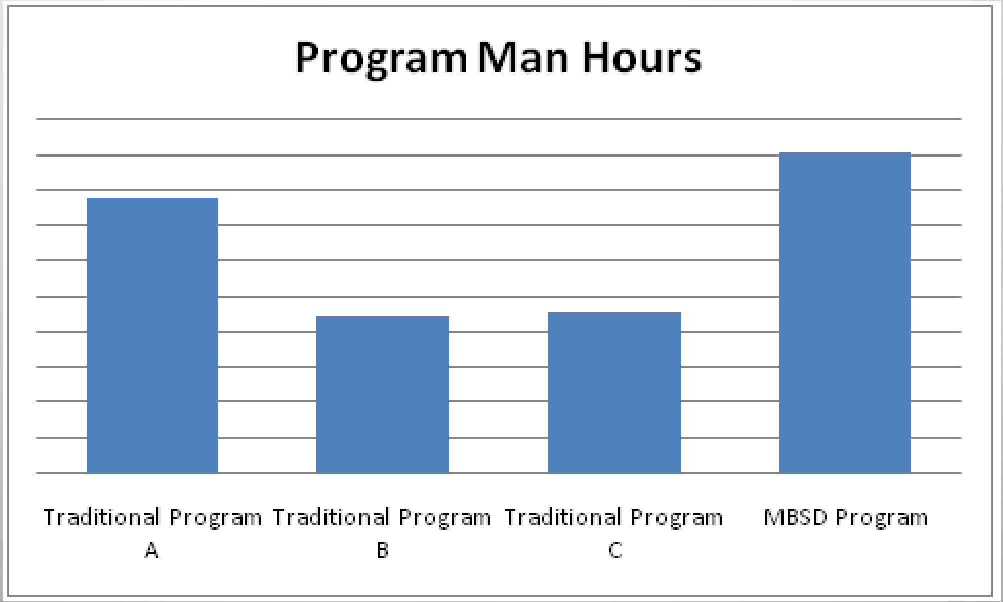
Though the MBSD-produced system was significantly larger than the other three, its defect density was greatly reduced.



# MBSE Return on Investment:
## *Program Cost in Man-Hours*

This figure compares the relative cost of these four programs in man-hours.  (Actual values have been omitted for proprietary reasons.)

As expected, the *total* number of man-hours for the MBSD-produced system was greater than the other three, correlated to system size.

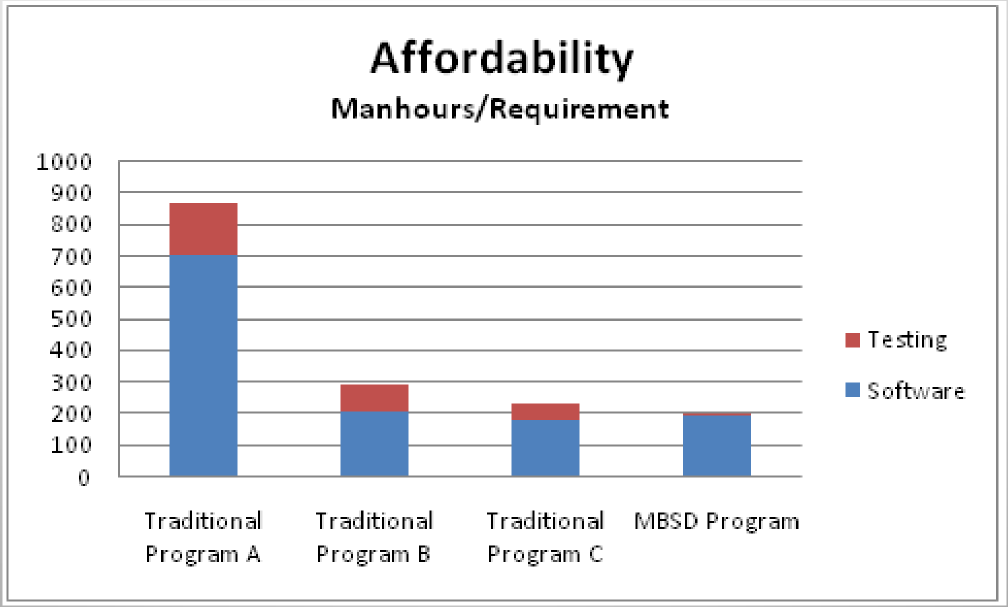


# MBSE Return on Investment:

## *Affordability*

When normalized for system size, however, the MBSD-produced system was developed at a significantly reduced cost.

Program C—the next closest in affordability—was 10% more costly than the MBSD-produced system.



# Contact Information

Lenny Delligatti, M.S., OCSMP Advanced

Cell phone: 757-630-2976

Work phone: 281-336-5095

E-mail: lenny.delligatti@lmco.com

lennydelligatti@gmail.com