Use of MATLAB & Simulink tools for Model-Based System Engineering (MBSE)

November, 2014

Ariel Rubanenko, Systematics

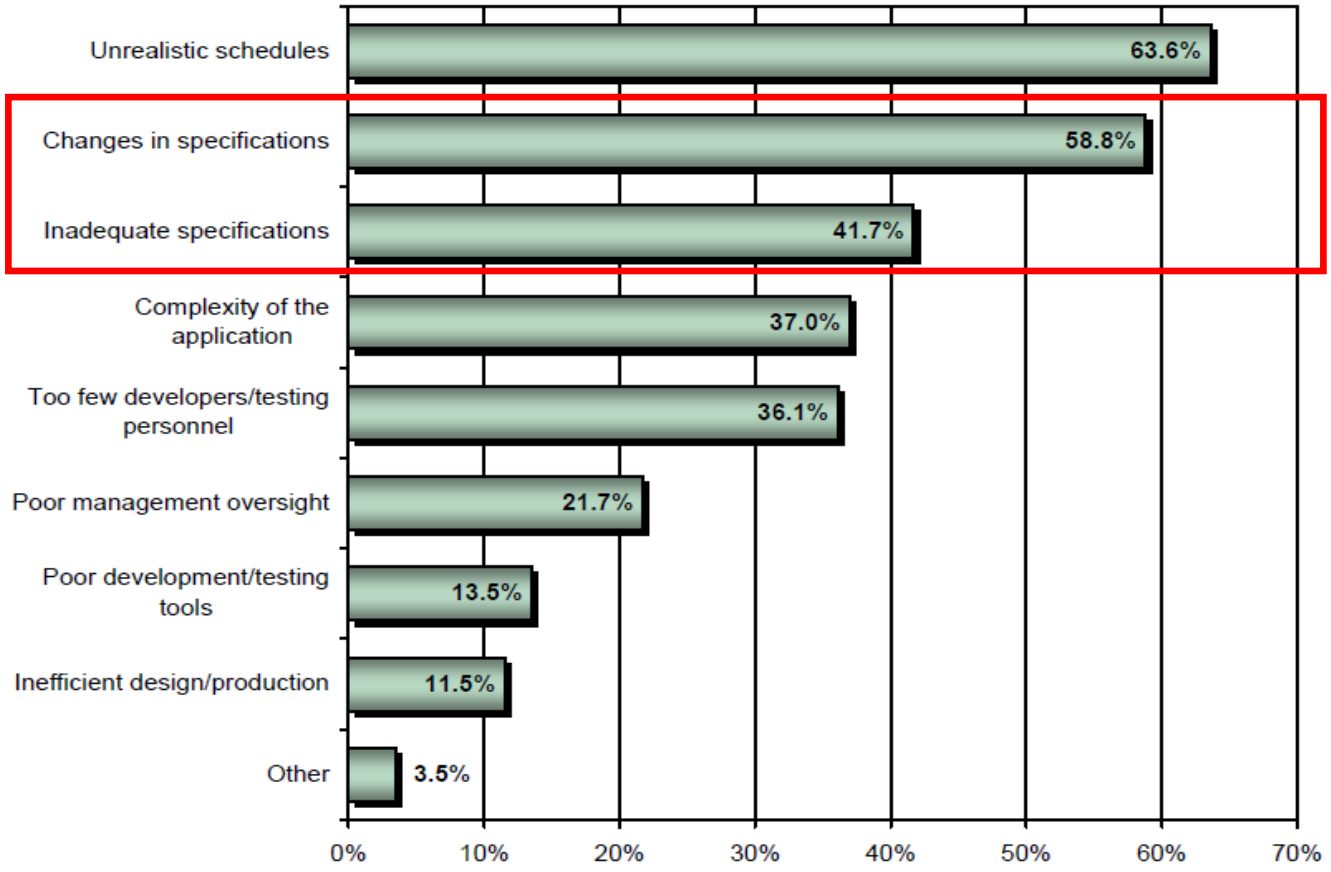
Application, Consulting & Tool Implementation for

Large-scale simulations, Guidance, Navigation & Control

Today's Agenda

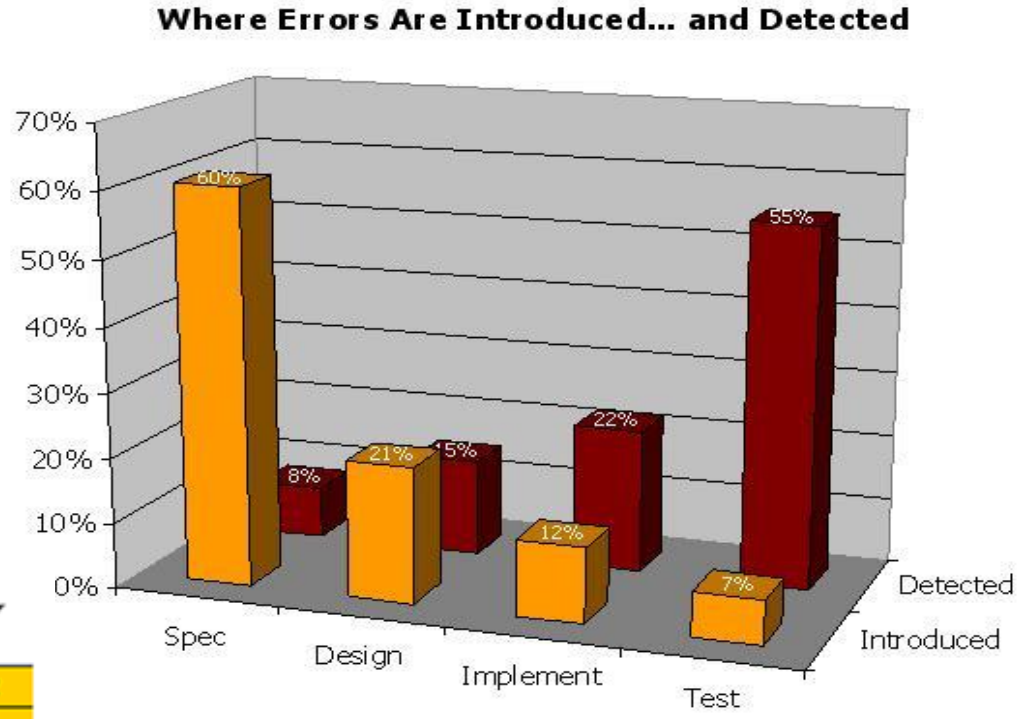
- Introduction to Model-Based System Engineering
- Introduction for MathWorks tools
- Example for system engineering workflow using Mathworks tools

Why did we miss our deadline?

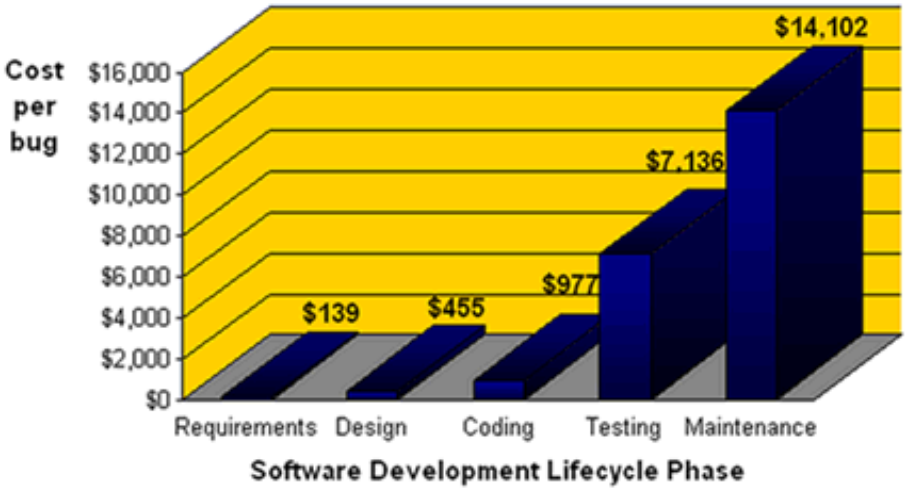


Reasons for late projects, as reported by Venture Development Corporation.
Source: Embedded Software Strategic Market Intelligence report, Volume 4, December 2007, VDC.
Note: Percentages sum to over 100% due to multiple responses.

Early Verification - WHY?

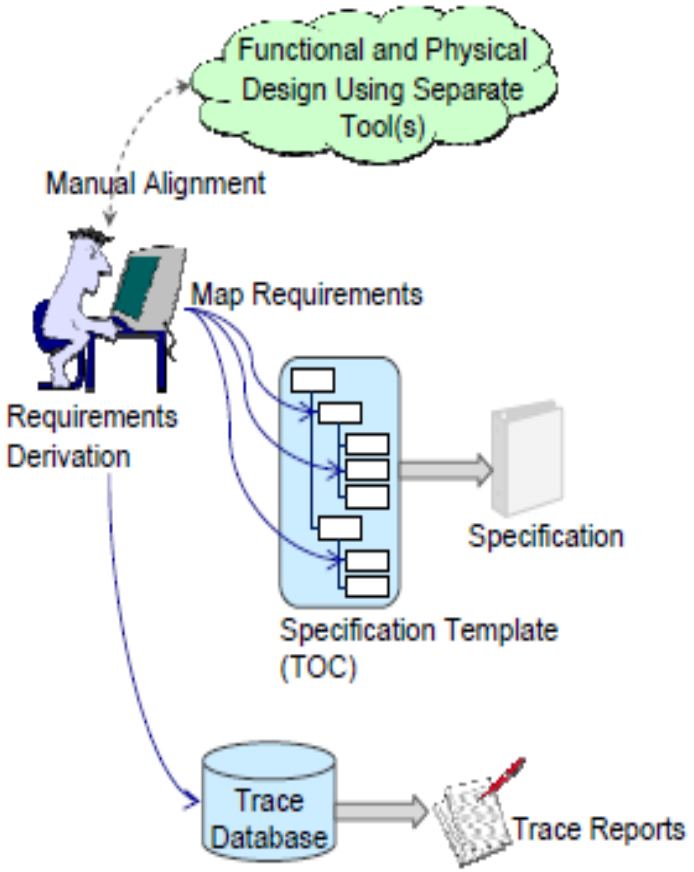


Source: B. Boehms and V. Basilli, "Software Defect Reduction Top 10 List", IEEE Computer



Document Centric .vs. MBSE

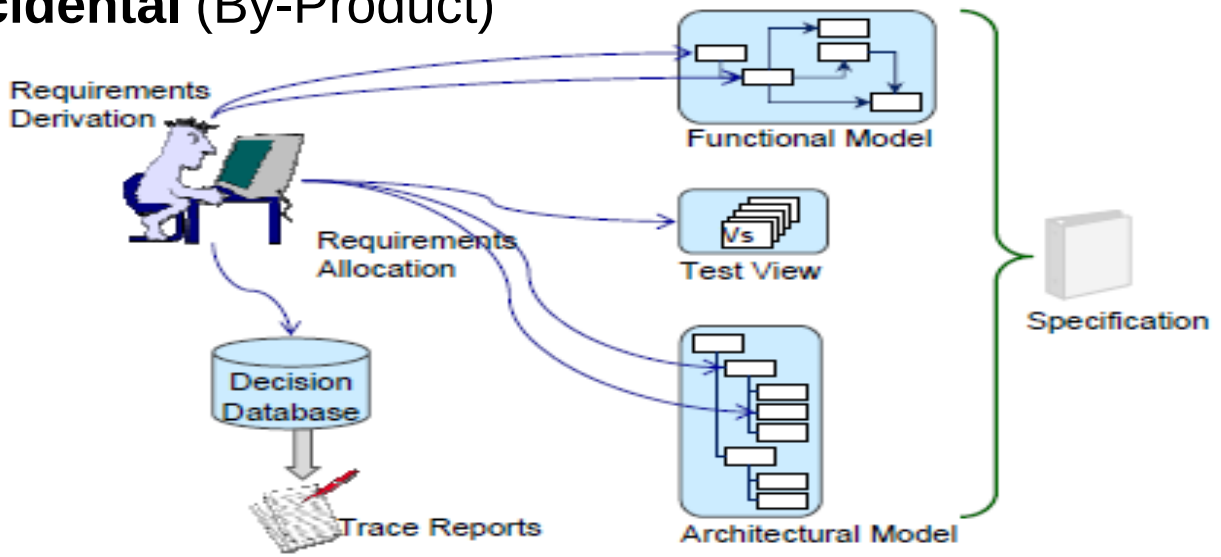
- Document Centric System Engineering
  - Focus is specification
  - Tool of Choice – Requirement Management Tools
  - Quality – traceability Reports?
  - Measure of completeness – Page count?
  - Correctness – Specification Structure is a static functional model



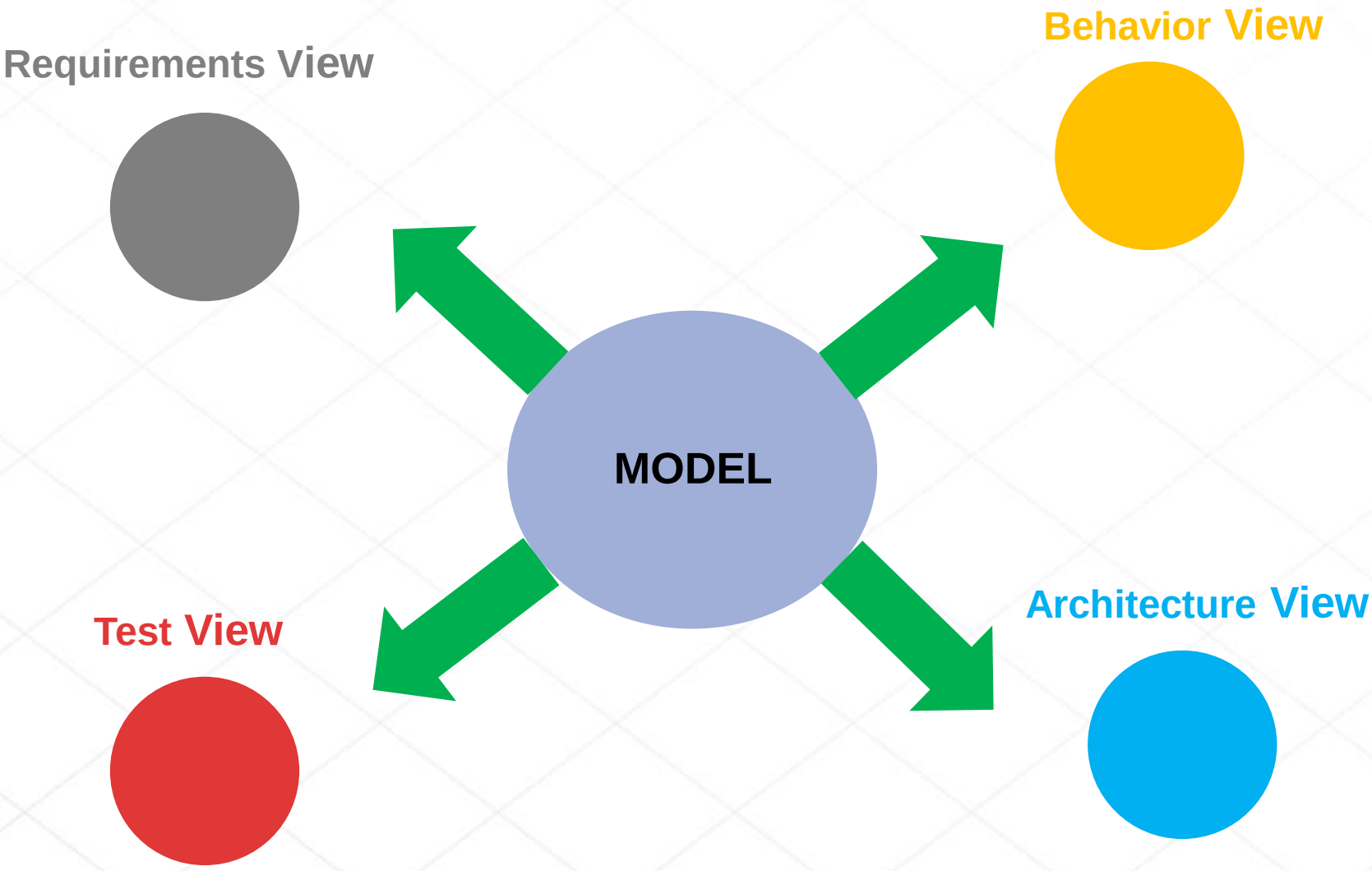
Requirements Specification is the Focus

Document Centric .vs. MBSE

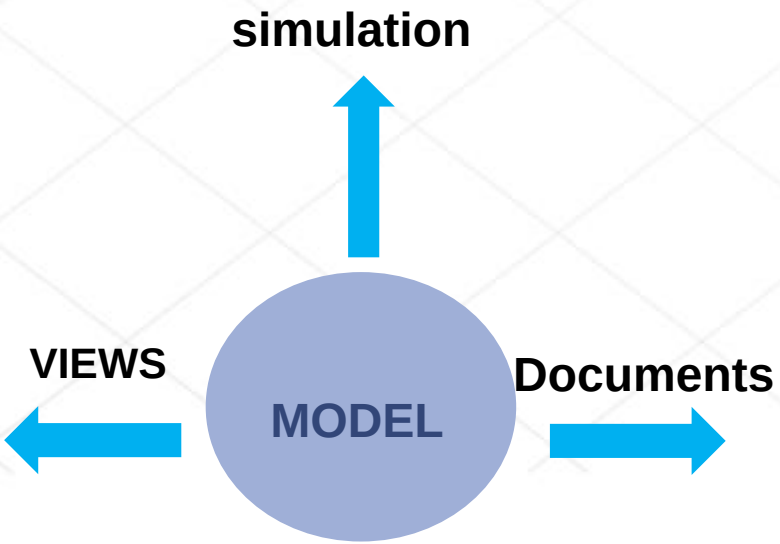
- Model Based System Engineering
  - Understanding the system behavior
  - Relating Requirements to Functions
  - Quality – traceability Reports?
  - Complete all Views of the Model
  - Specification is **incidental** (By-Product)



Requirements Specification is a By-Product

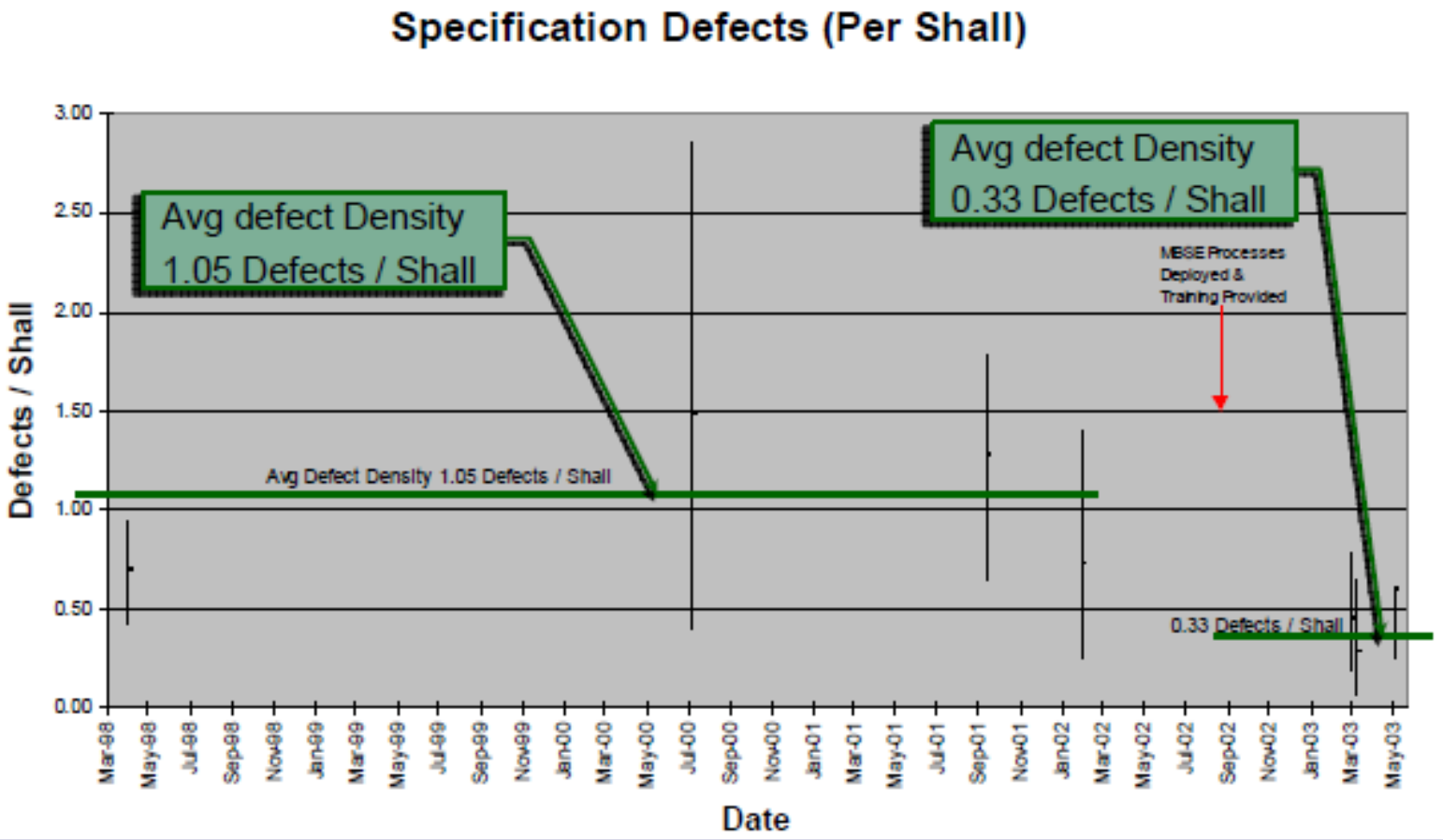

Requirements View
Behavior View
MODEL
Test View
Architecture View

Document Driven	Model-Based SE (Improved Practice)
Gathered from Documents at hand	Derived from model
Independent drawings	Consistent views
Static diagrams	Executable behavior
Data storage	Linked repository
Stored views	Dynamic view generation
Ad hoc process (inconsistent results)	Repeatable process (consistent results)
Manual change propagation across all affected products (by the systems engineer)	Automatic change propagation across current and future products



Deploying MBSE on a Program – Lessons Learnt

Raytheon



68% Reduction in Specification Defects since MBSE Practices Introduced

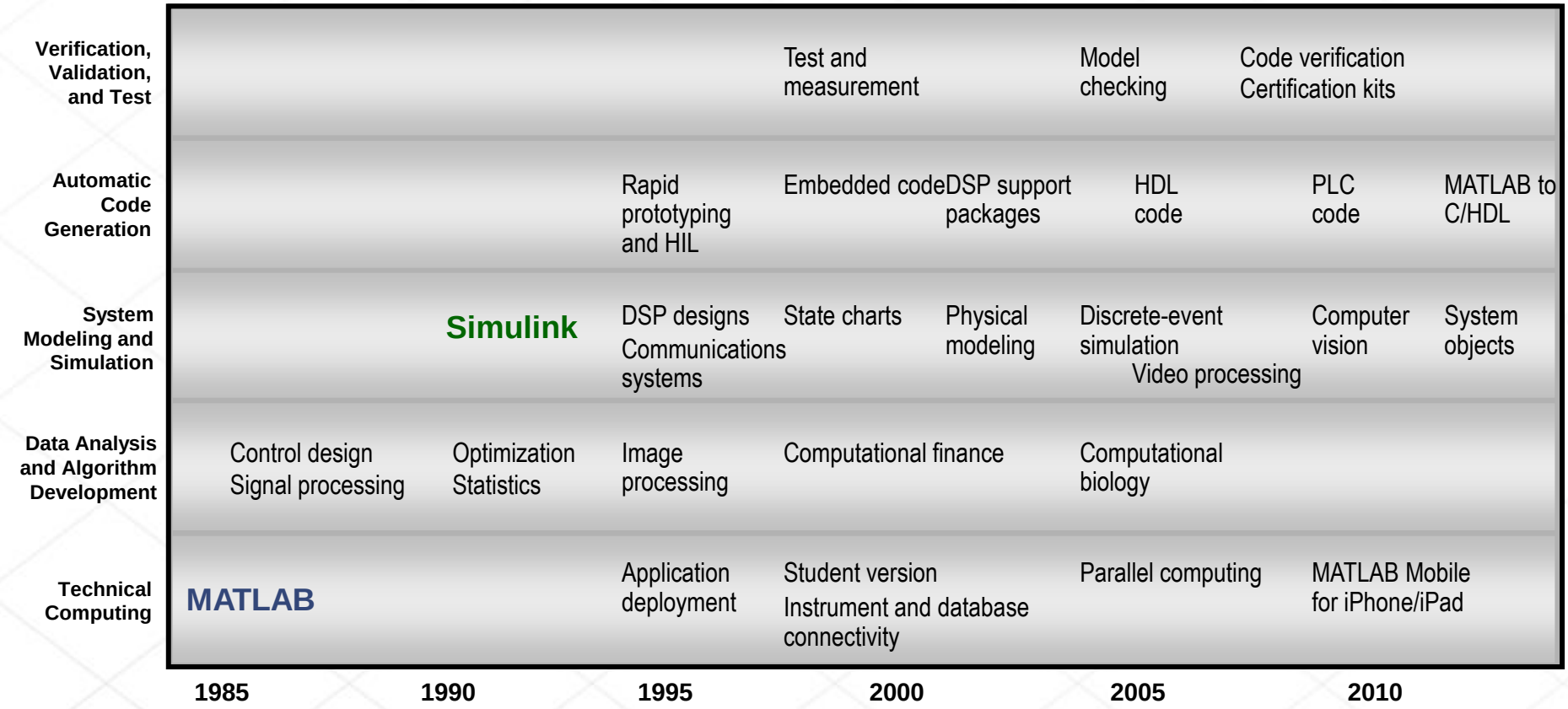
Overview of MathWorks

MathWorks at a Glance

Earth's topography on a Miller cylindrical projection, created with MATLAB and Mapping Toolbox.

- **Headquarters:** Natick, Massachusetts U.S.
- **Other U.S. Locations:** California; Michigan; Texas; Washington, D.C.
- **Europe:** France, Germany, Italy, Netherlands, Spain, Sweden, Switzerland, United Kingdom
- **Asia-Pacific:** Australia, China, India, Japan, Korea
- Worldwide training and consulting
- Distributors serving more than 20 countries

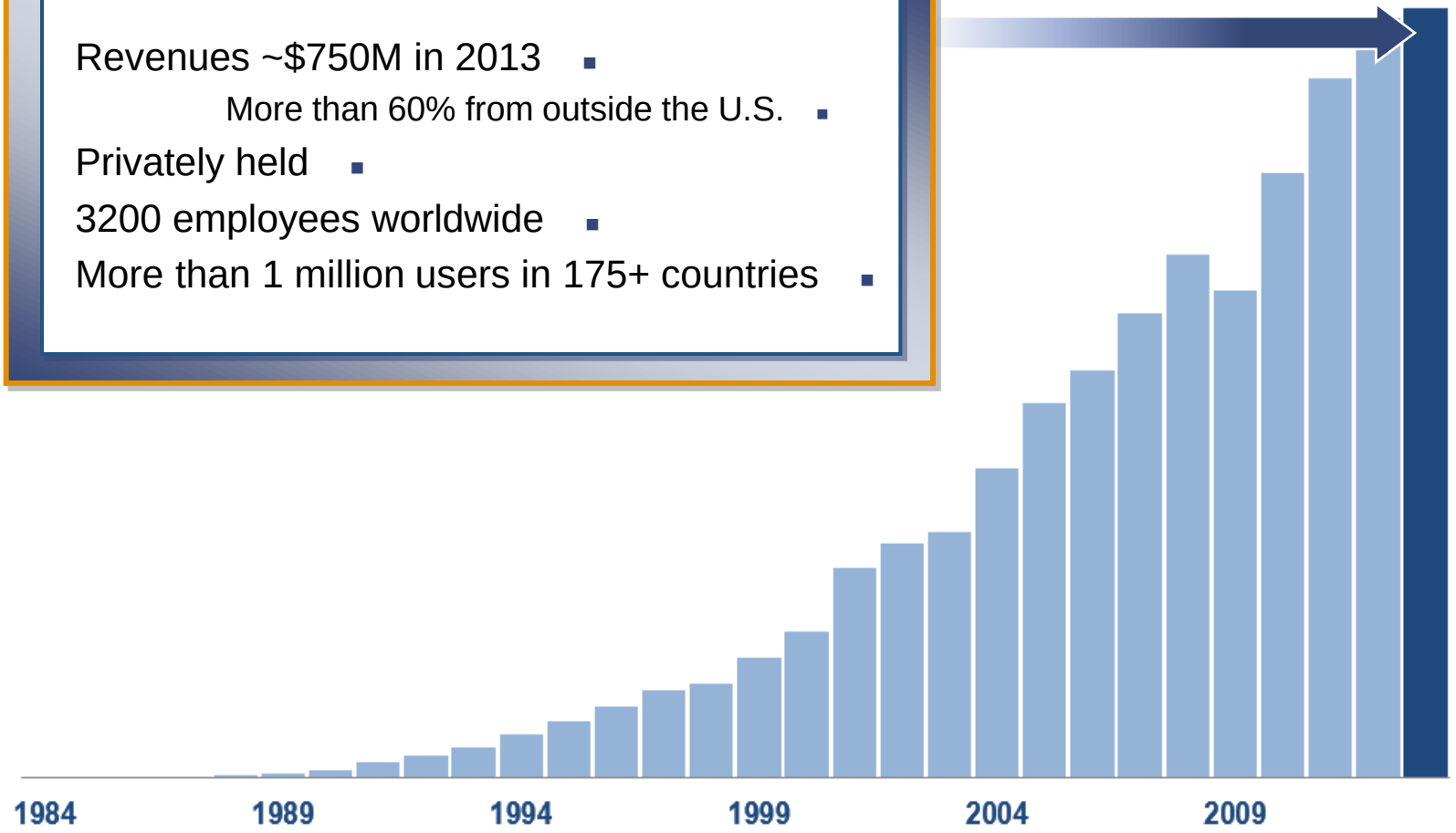
Key capabilities drive MathWorks business



Founded in 1984

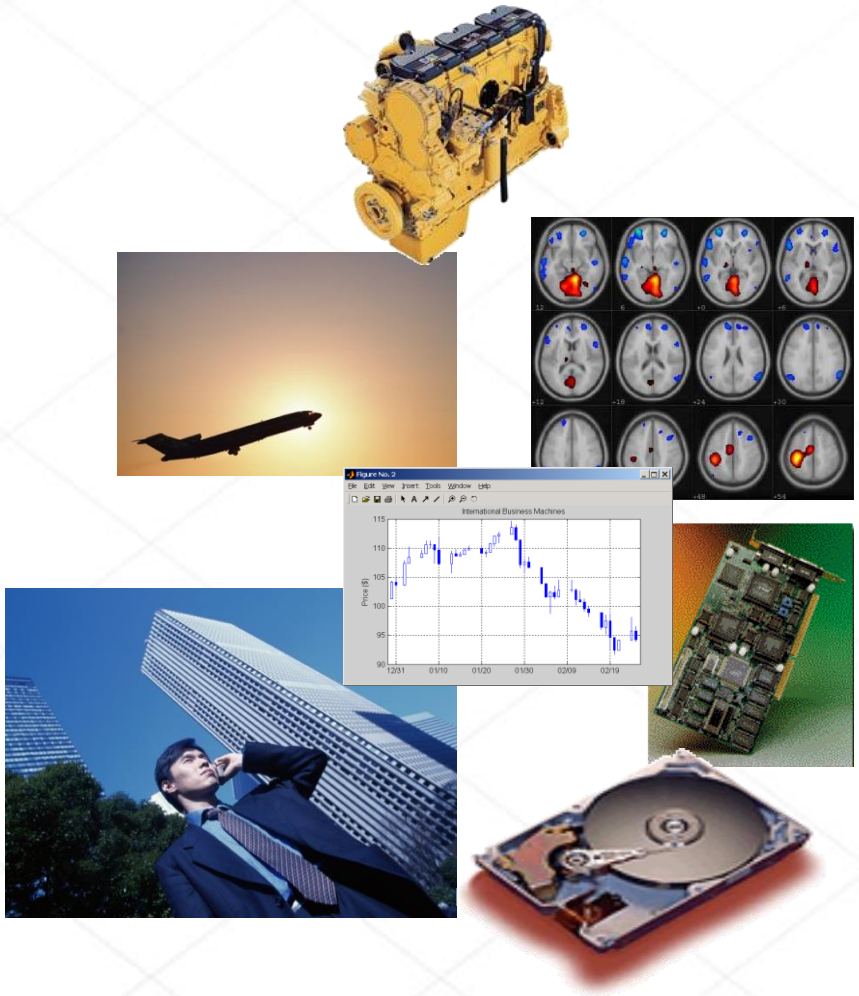
MathWorks Today

- Revenues ~$750M in 2013
  - More than 60% from outside the U.S.
- Privately held
- 3200 employees worldwide
- More than 1 million users in 175+ countries



Key Industries

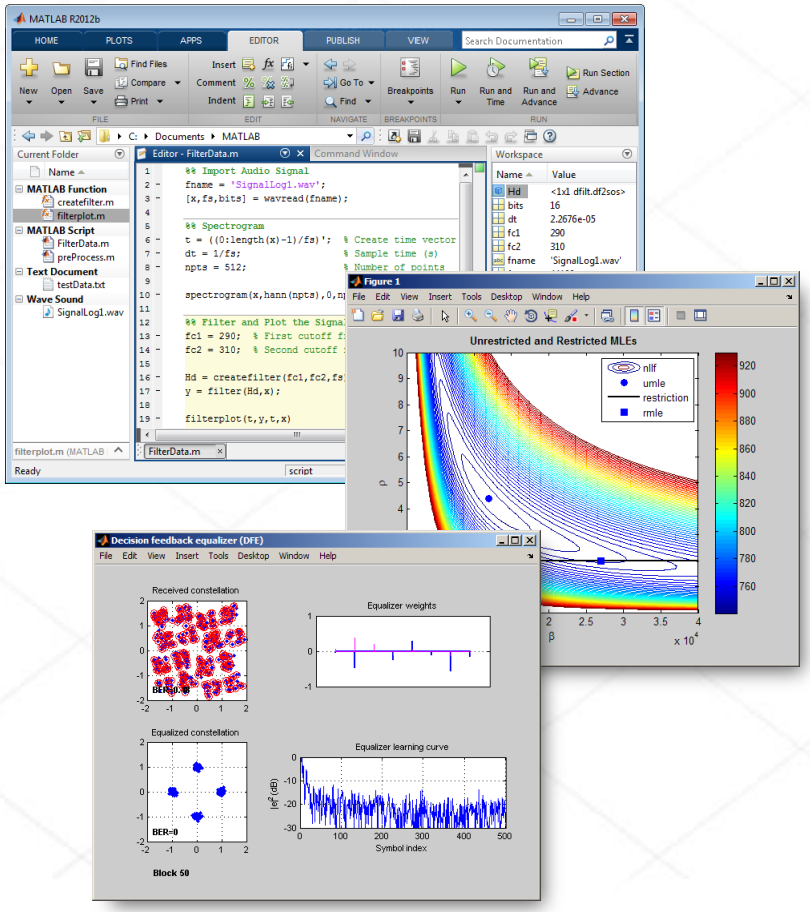
- Aerospace and defense
- Automotive
- Biotech and pharmaceutical
- Communications
- Education
- Electronics and semiconductors
- Energy production
- Financial services
- Industrial automation and machinery
- Medical devices



MATLAB®

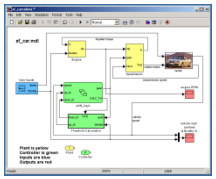
The leading environment for technical computing

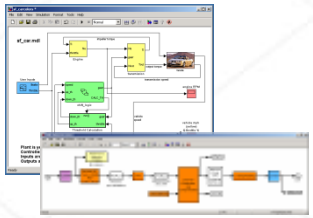
- The industry-standard, high-level programming language for algorithm development
- Numeric computation
- Parallel computing, with multicore and multiprocessor support
- Data analysis and visualization
- Toolboxes for signal and image processing, statistics, optimization, symbolic math, and other areas
- Tools for application development and deployment
- Foundation of MathWorks products



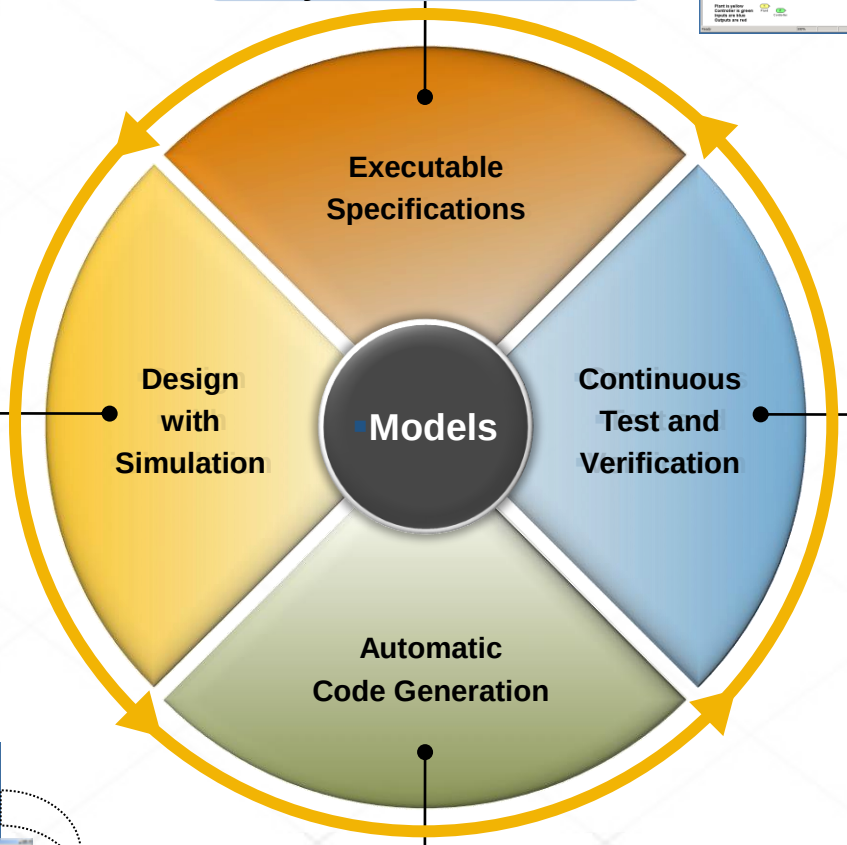
Advantages of Model-Based Design

Executable models
-unambiguous
-only "one truth"

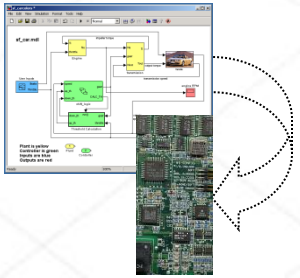


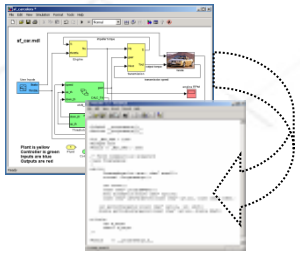


Simulation
-reduces "real" prototypes
-systematic "what-if" analysis



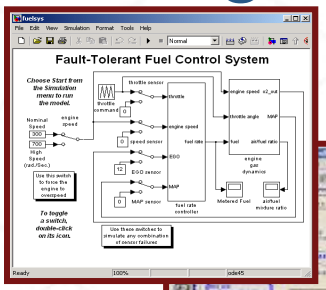
Test with Design
- detects errors earlier

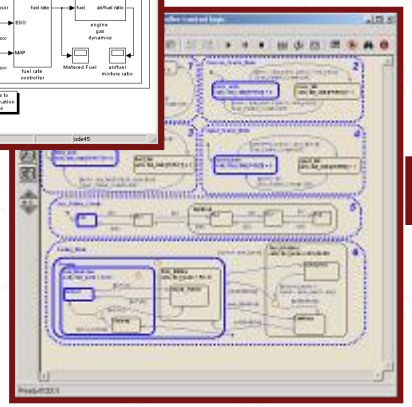




Automatic code generation
-minimizes coding errors

Design an Embedded Controller

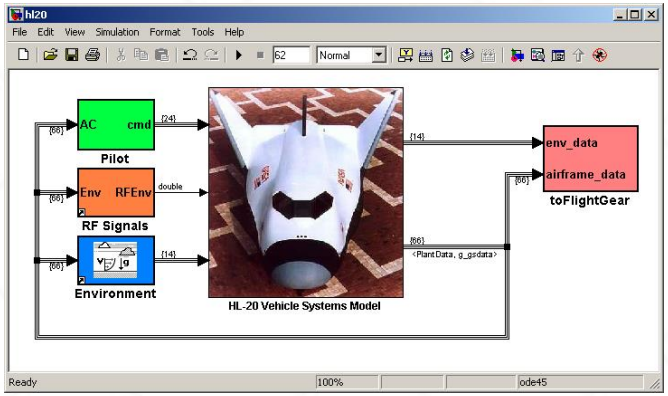




Algorithm Simulation

Code Generation

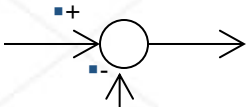
Plant Simulation



Controller

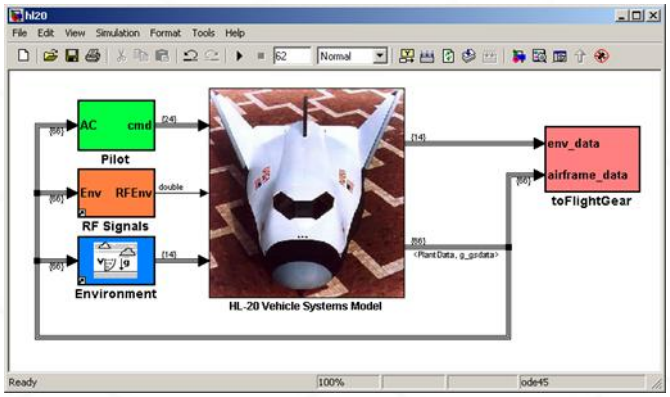
Plant

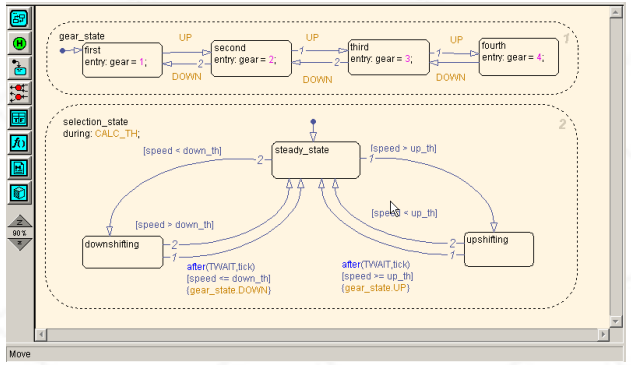
Command



Simulink environment

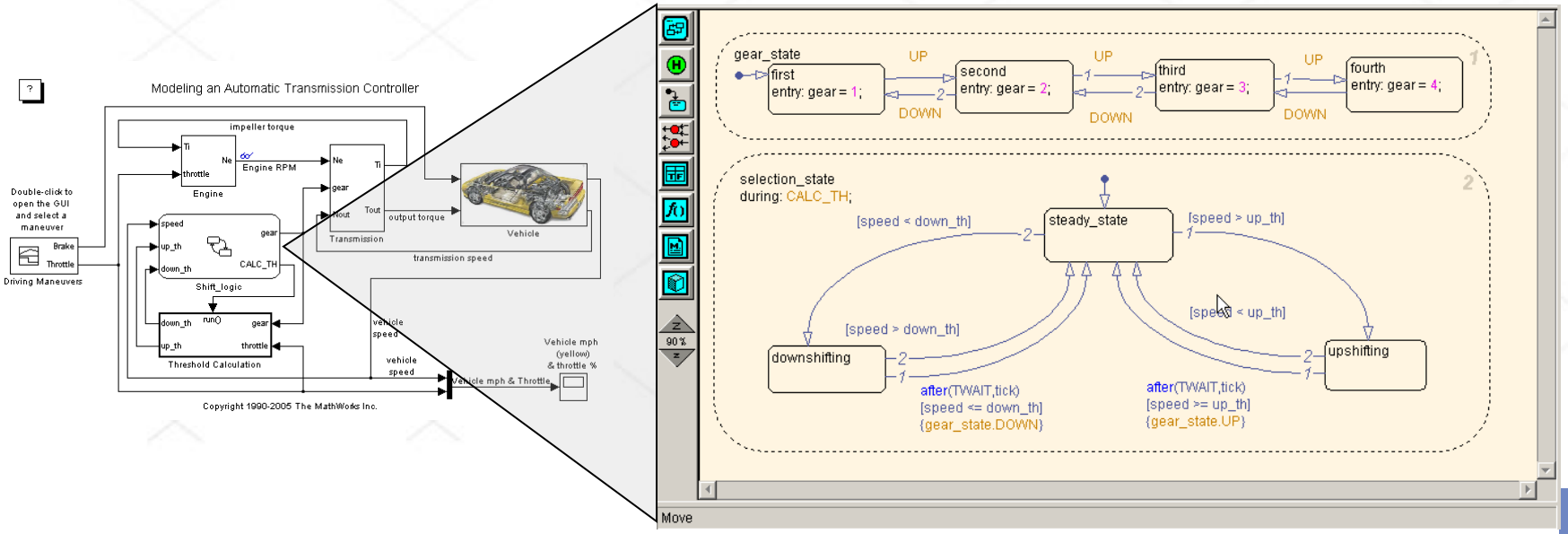
- System-level modeling
  - Graphical
  - Interactive
  - Hierarchical
- Simulation
  - Model is an “executable specification”
  - Analog + Digital Simulation
  - Easy code generation
- Multi - domain simulation
  - Stateflow – logic modeling
  - simEvents – event simulation
  - Physical modeling
- Animation capabilities





Stateflow

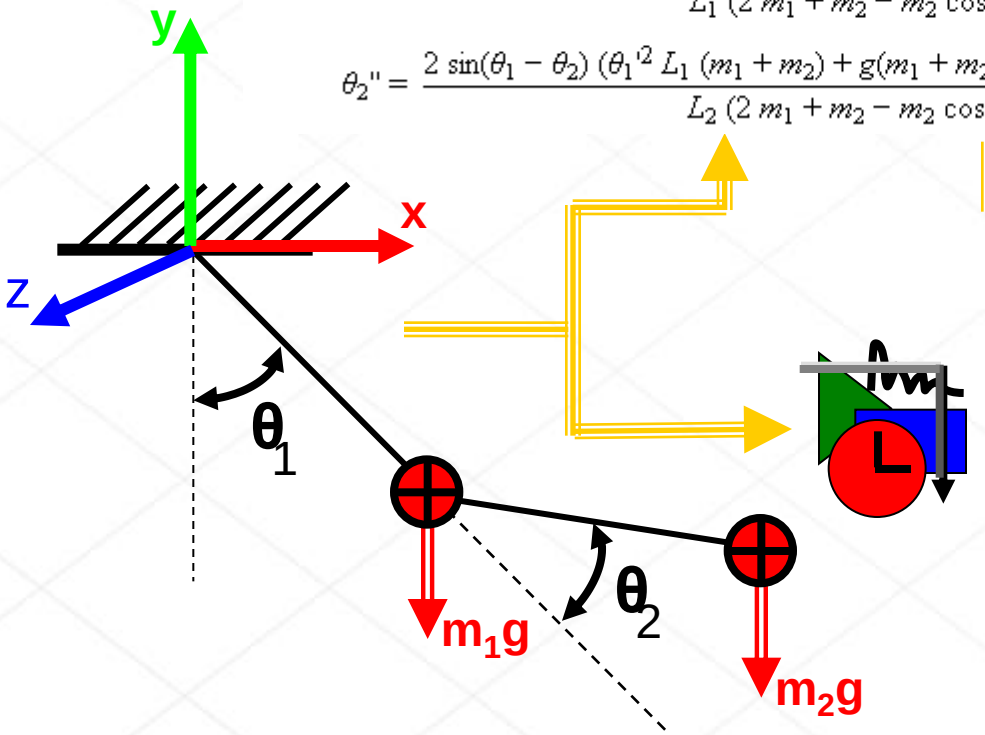
- Extend Simulink with a design environment for developing state machines and flow charts
- Design systems containing control, supervisory, and mode logic
- Describe logic in a natural and understandable form with deterministic execution semantics

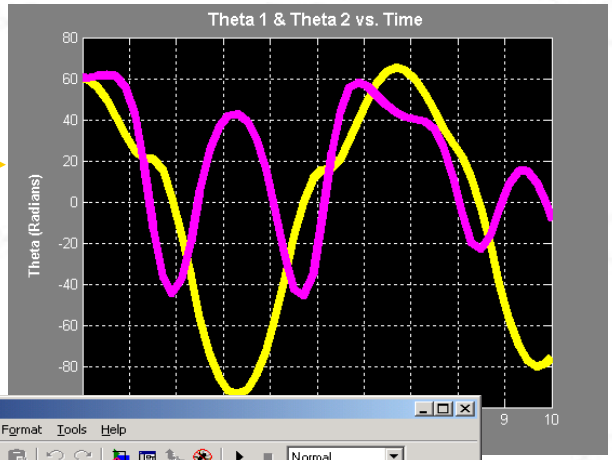


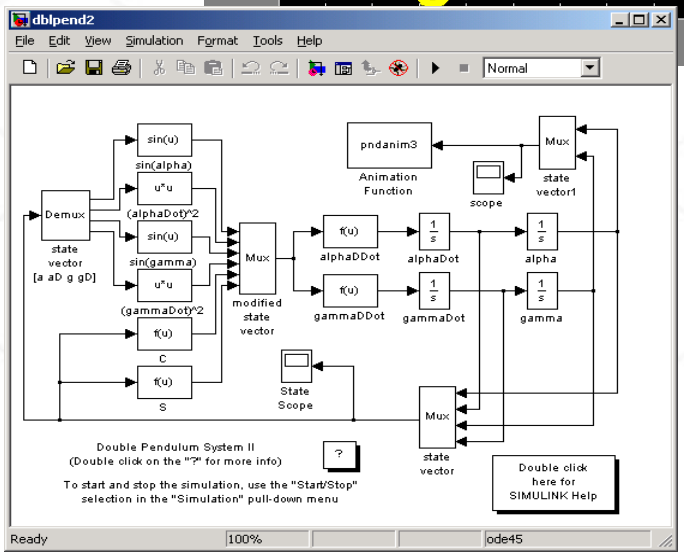
Introduction to SimMechanics

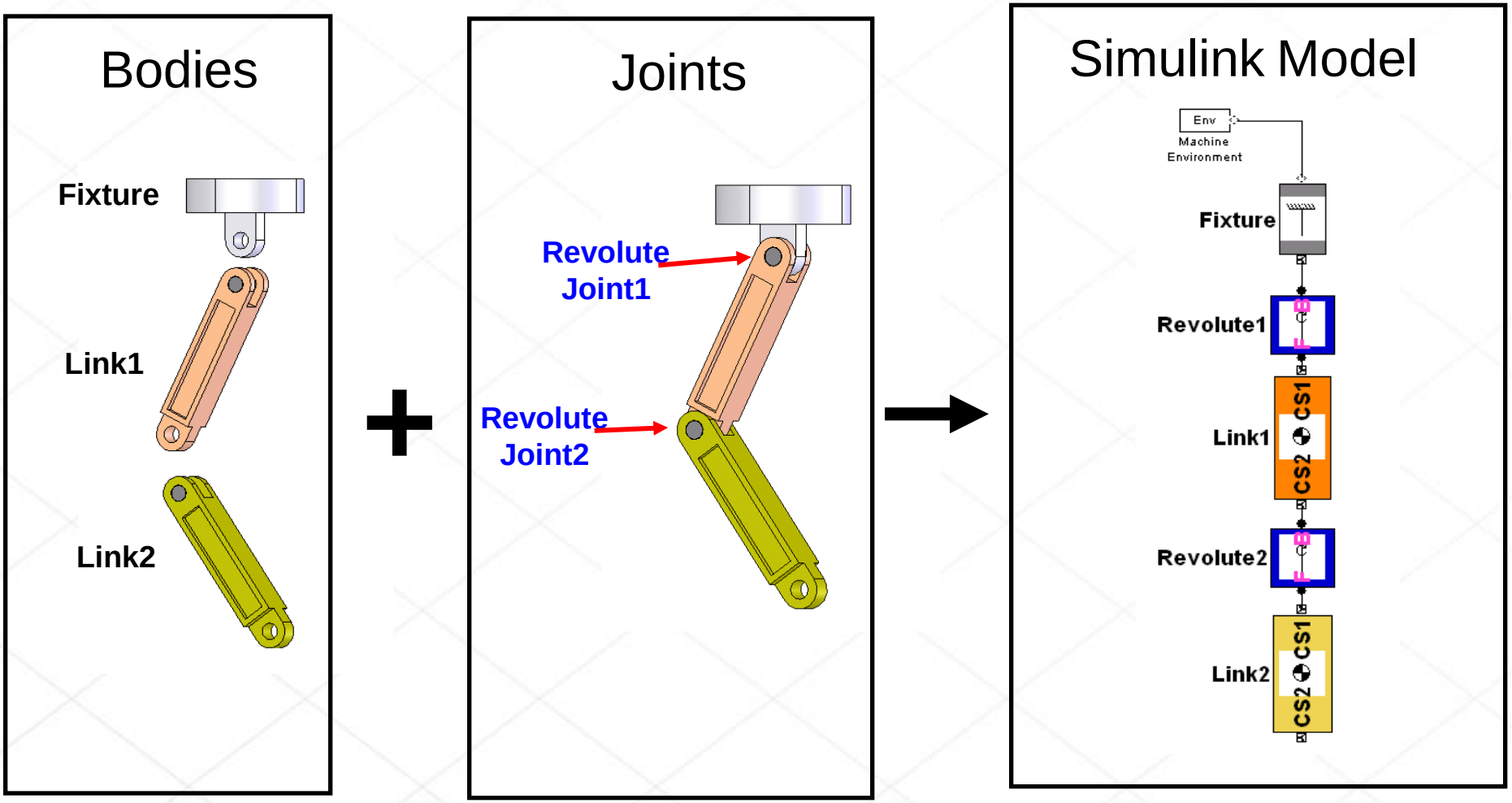
$$\theta_1'' = \frac{-g\,(2\,m_1 + m_2)\,\sin\theta_1 - m_2\,g\,\sin(\theta_1 - 2\,\theta_2) - 2\,\sin(\theta_1 - \theta_2)\,m_2\,(\theta_2'^2\,L_2 + \theta_1'^2\,L_1\,\cos(\theta_1 - \theta_2))}{L_1\,(2\,m_1 + m_2 - m_2\,\cos(2\,\theta_1 - 2\,\theta_2))}$$

$$\theta_2'' = \frac{2\,\sin(\theta_1 - \theta_2)\,(\theta_1'^2\,L_1\,(m_1 + m_2) + g(m_1 + m_2)\,\cos\theta_1 + \theta_2'^2\,L_2\,m_2\,\cos(\theta_1 - \theta_2))}{L_2\,(2\,m_1 + m_2 - m_2\,\cos(2\,\theta_1 - 2\,\theta_2))}$$



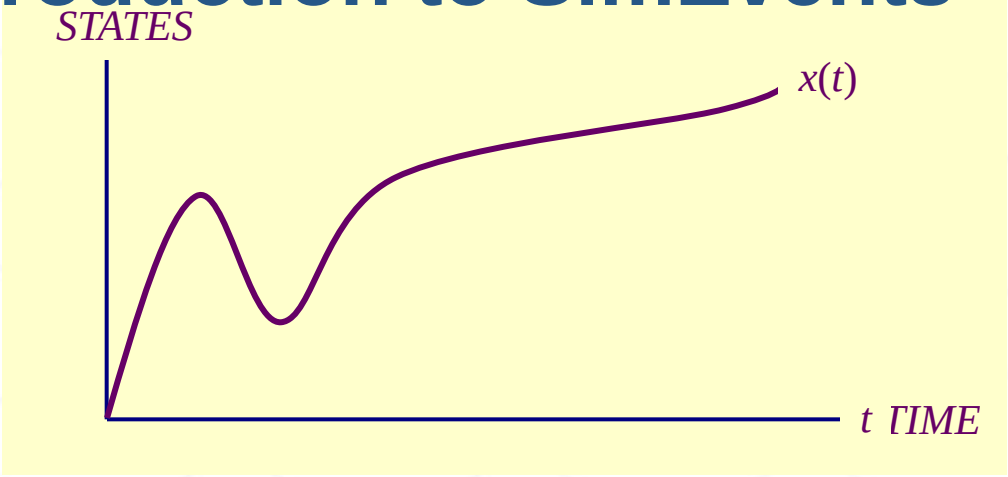





Bodies
Fixture
Link1
Link2
Joints
Revolute Joint1
Revolute Joint2
Simulink Model
Env
Machine Environment
Fixture
Revolute1
Link1
CS1
CS2
Revolute2
Link2
CS1
CS2

Introduction to SimEvents

TIME-DRIVEN SYSTEM



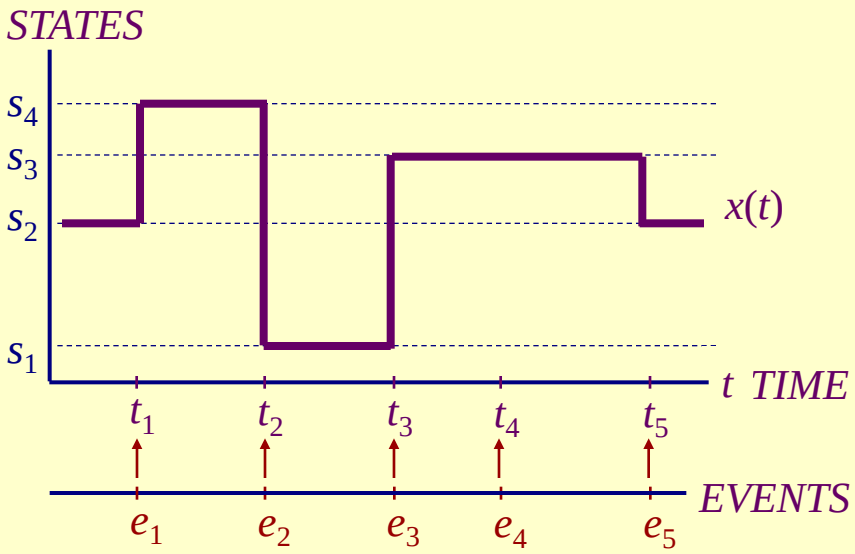
STATE SPACE:

$$X \subseteq \Re$$

DYNAMICS:

$$\dot{x} = f(x,t)$$

EVENT-DRIVEN SYSTEM

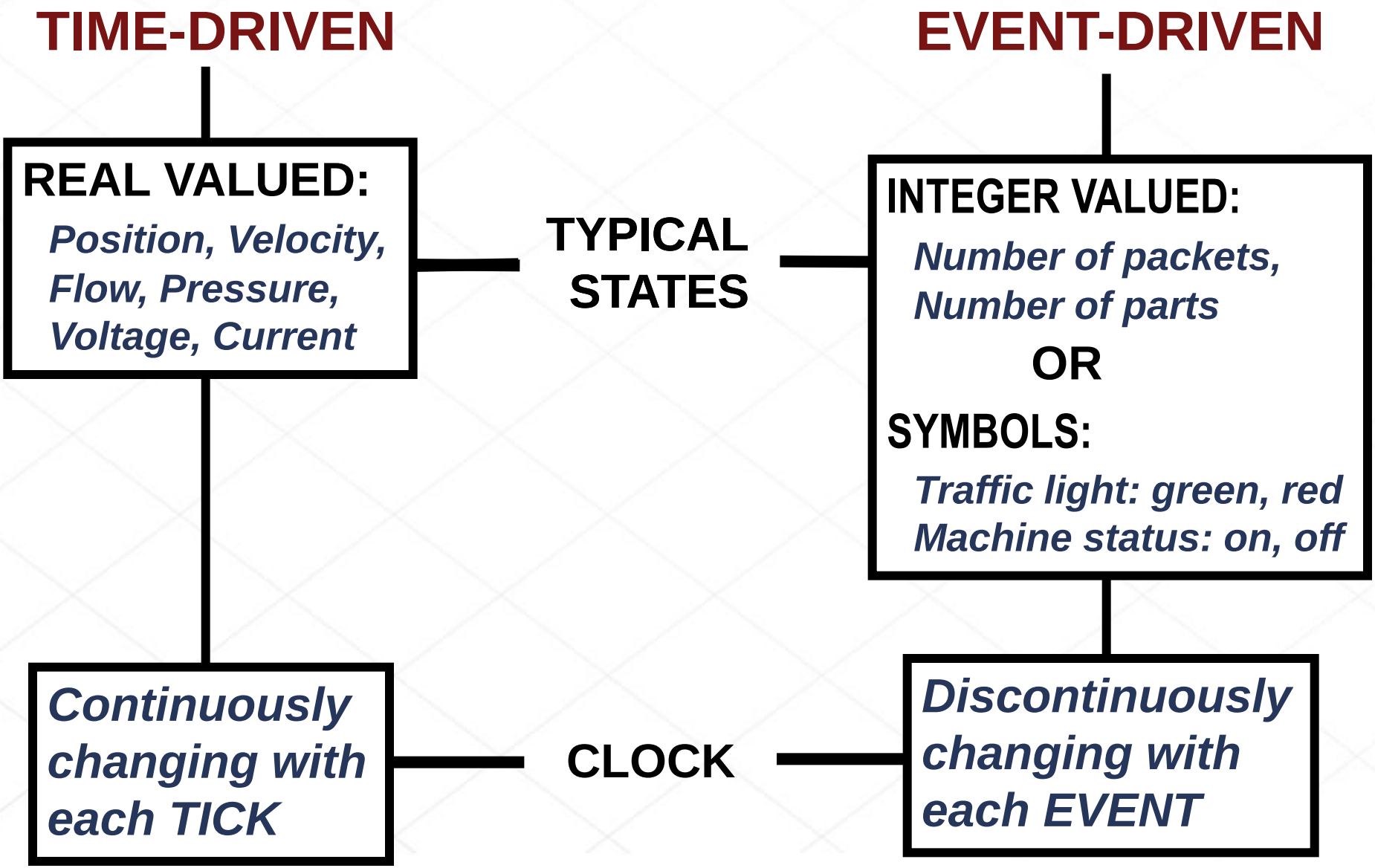


STATE SPACE:

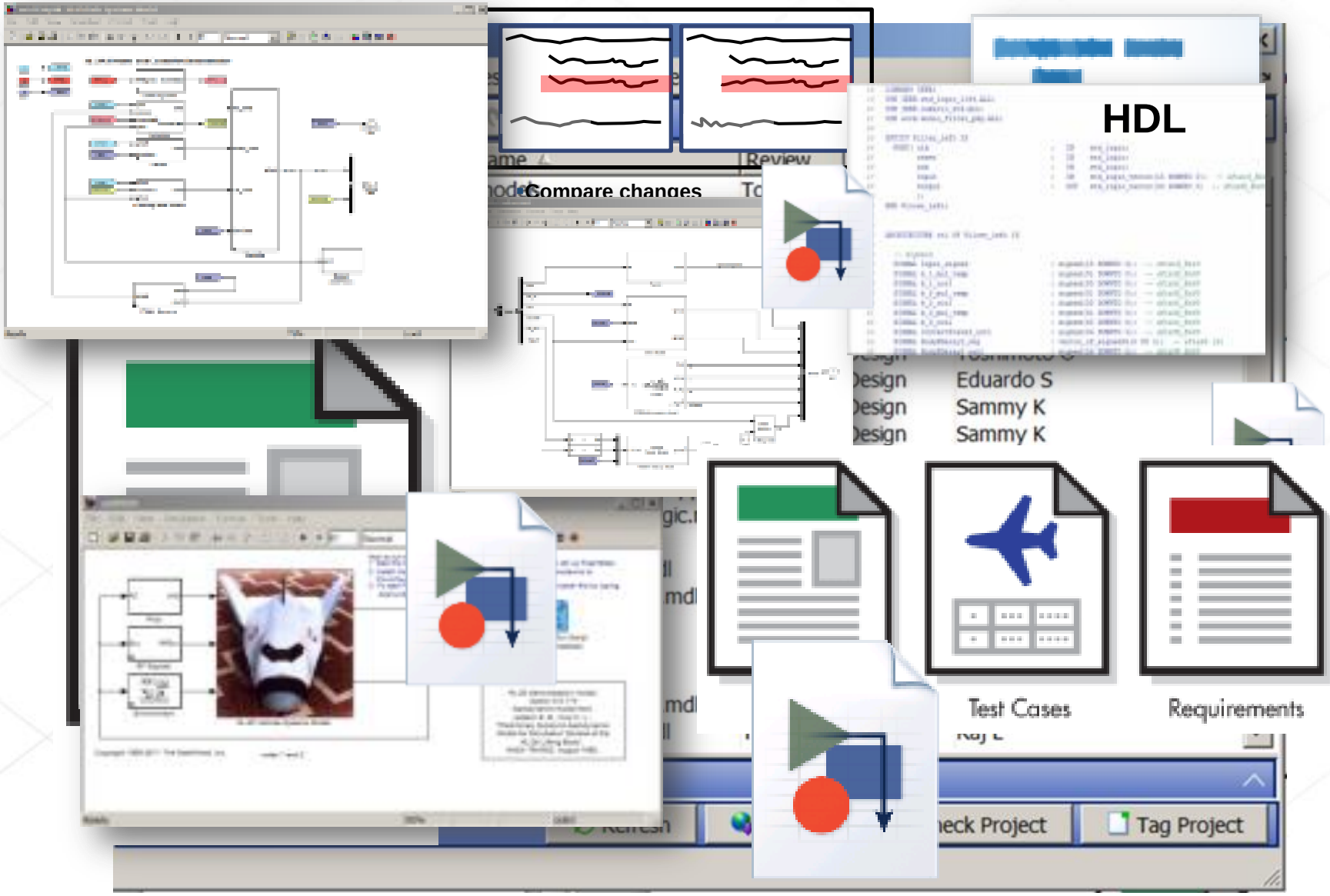
$$X = \{s_1, s_2, s_3, s_4\}$$

DYNAMICS:

$$x' = f(x,e)$$


TIME-DRIVEN
EVENT-DRIVEN
REAL VALUED:
Position, Velocity,
Flow, Pressure,
Voltage, Current
TYPICAL STATES
INTEGER VALUED:
Number of packets,
Number of parts
OR
SYMBOLS:
Traffic light: green, red
Machine status: on, off
Continuously changing with each TICK
CLOCK
Discontinuously changing with each EVENT

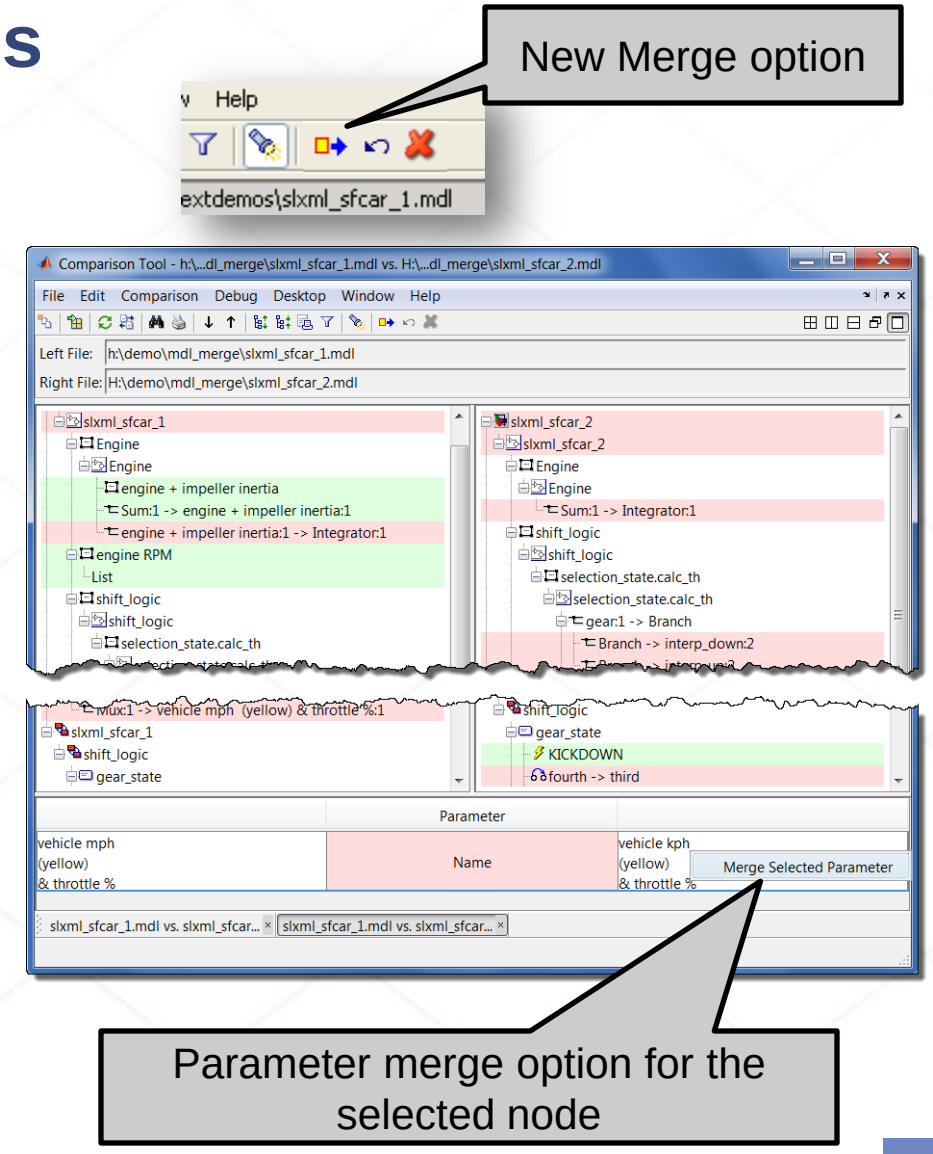
Simulink Projects Helps You Get Organized



Merge Simulink Models Based on XML Comparison Differences

Merge Simulink models from within XML comparison report

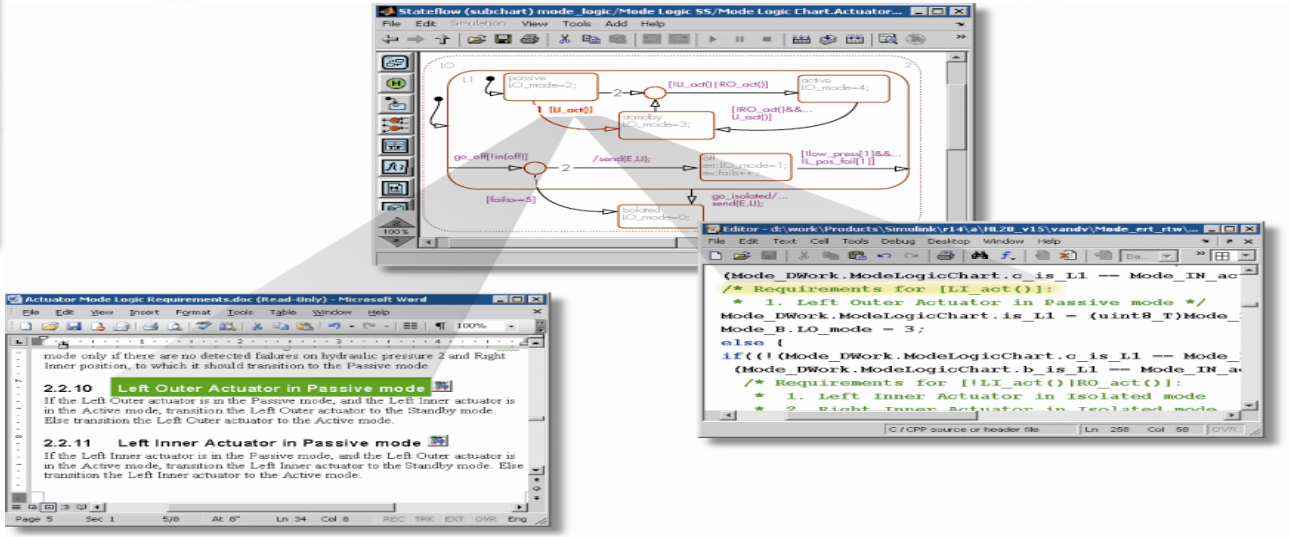
- Merge models within the tool by merging changes from left to right:
  - Left model is the base
  - Right model is the one edited
- Merge individual parameters, blocks, or entire subsystems



Requirements Traceability - Overview

IBM Rational DOORS®
Microsoft Word ®
Microsoft Excel ®
Simulink/Stateflow
MATLAB scripts

Supported document formats

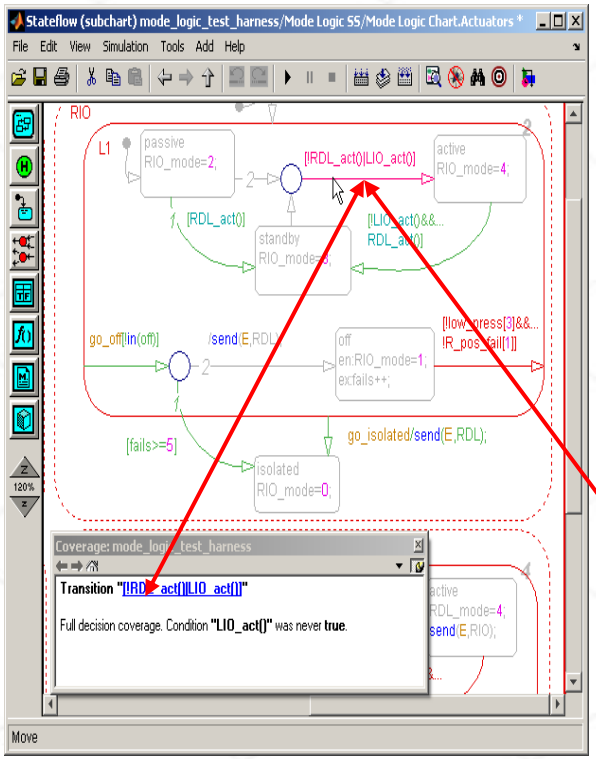


- Bi-directional linking with external documents
  - Requirements consistency checks
  - Extensibility API
  - Report generation
- HDL Coder / Embedded Coder integration
  - Embeds requirements as comments in source code

Simulink
Stateflow

Embedded Coder
HDL Coder

Test Coverage Analysis for Models



- Decision coverage
- Condition coverage
- MC/DC
- Lookup table coverage
- Signal range coverage

Supported coverage types

Coverage from first simulation

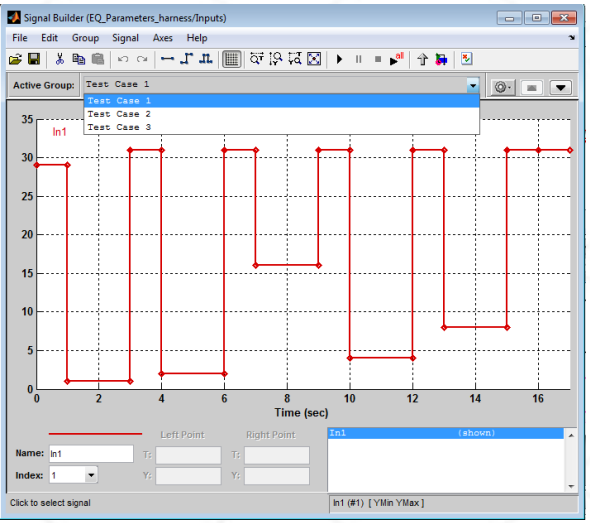
Coverage from second simulation

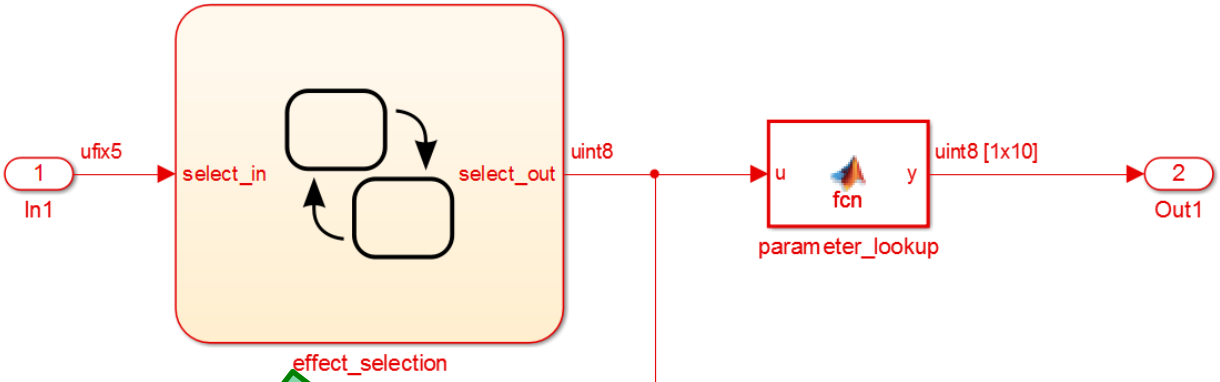
Total coverage

Summary

Model Hierarchy/Complexity:		Test 1 MCDC	Test 2 MCDC	Total MCDC
1. fuelsys_cov	**81**	31%	63%	63%
2. . . . fuel rate controller	75	31%	63%	63%
3. . . . . Airflow calculation	1	100%	100%	100%
4. . . . . Fuel Calculation	11	100%	100%	100%
5. . . . . Switchable Compensation	7	100%	100%	100%
6. . . . . control logic	**54**	8%	50%	50%
7. . . . . SF: control logic	53	8%	50%	50%
8. . . . . SF: Fueling_Mode	20	50%	50%	50%
9. . . . . SF: Fuel_Disabled	5	50%	50%	50%
10. . . . . SF: Pressure_Sensor_Mode	5	0%	100%	100%
11. . . . . SF: Speed_Sensor_Mode	4	0%	0%	0%
12. . . . . SF: Throttle_Sensor_Mode	5	0%	25%	25%

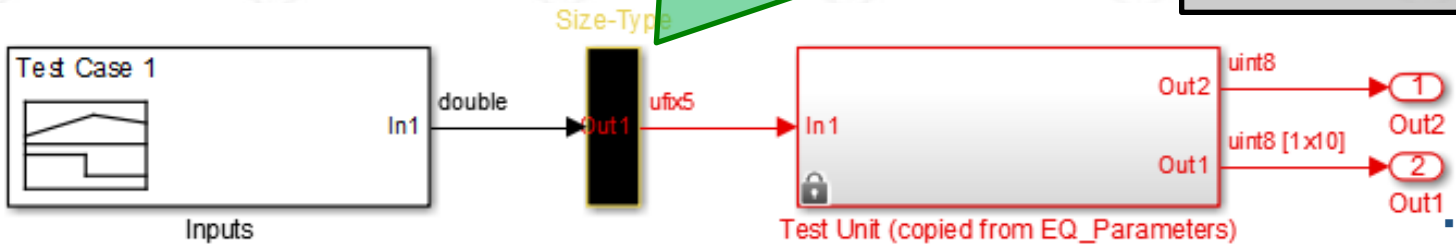
Test Generation for 100% Coverage





Test Generation

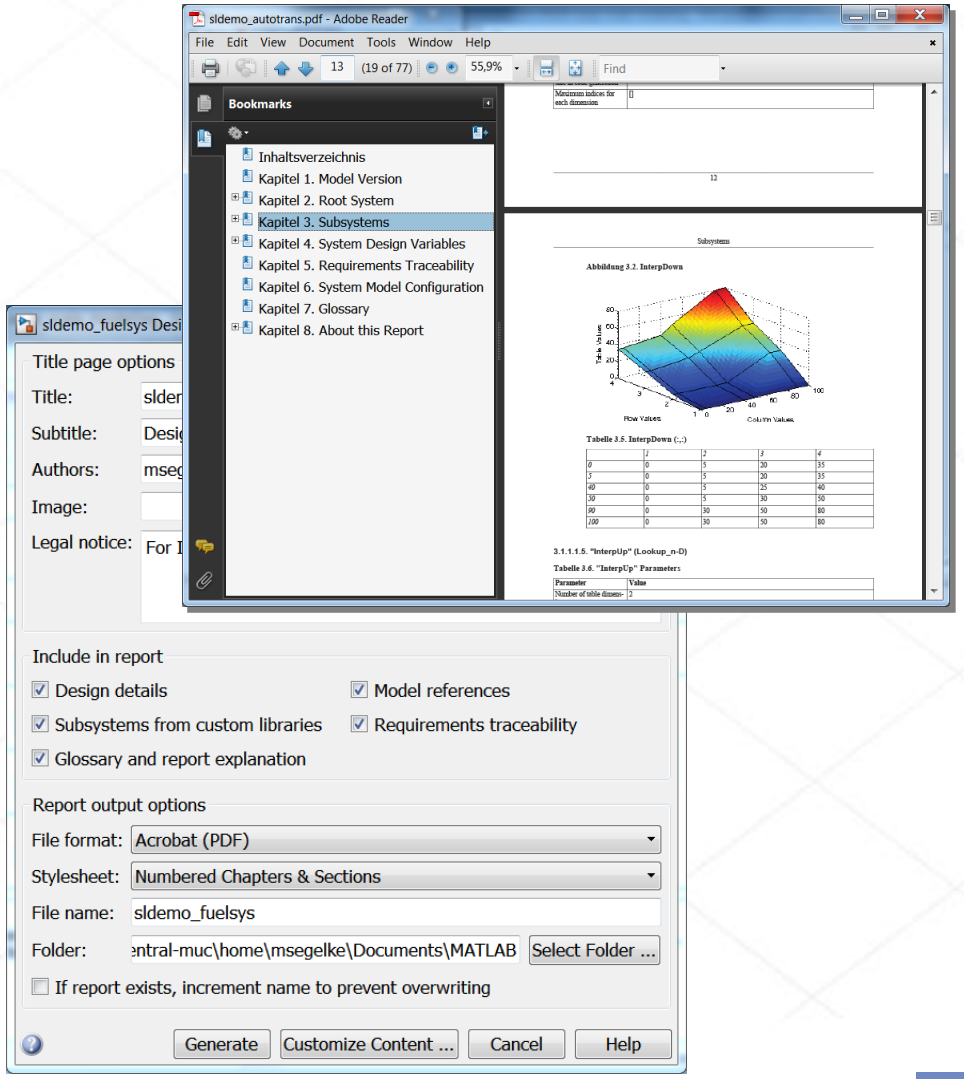
Automatically generate tests to reach coverage objectives



- >>sf_security
- (change chart for
- fast run)

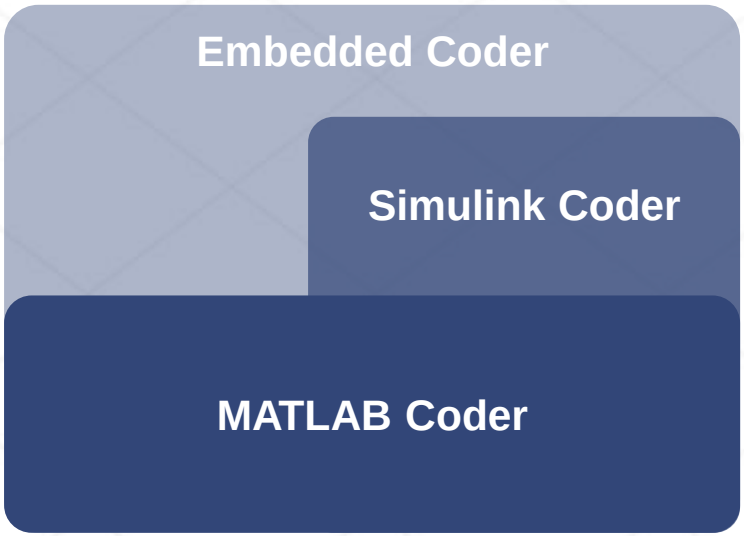
Generation of System Design Descriptions

- Generate different formats
- Select from predefined stylesheets
- Customization on Simulink Report Generator



C-Code Generation Tools

- MATLAB Coder
- Simulink Coder
- Embedded Coder

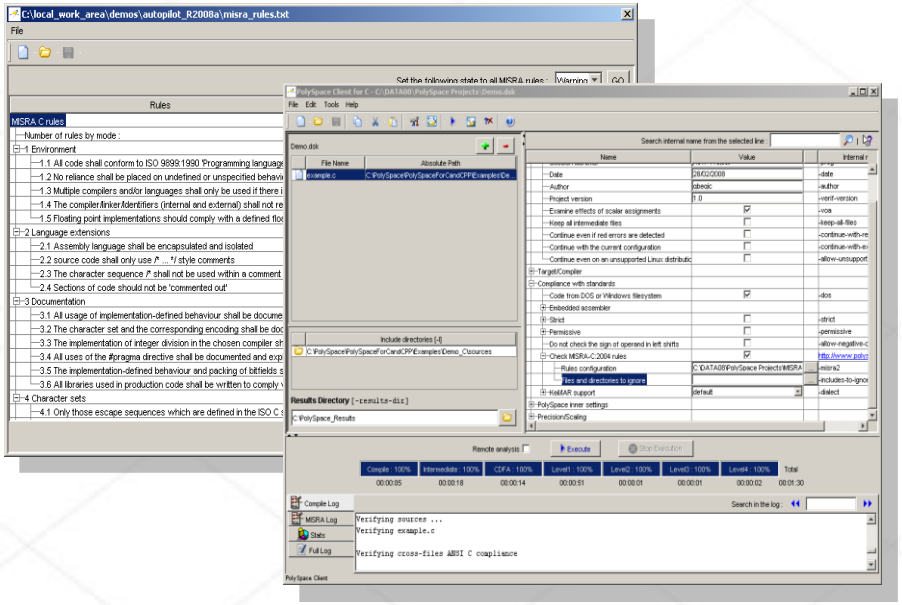


Code Correctness: Absence of Runtime Errors

PolySpace™ Solution
For hand written code

- Formal method: Abstract Interpretation

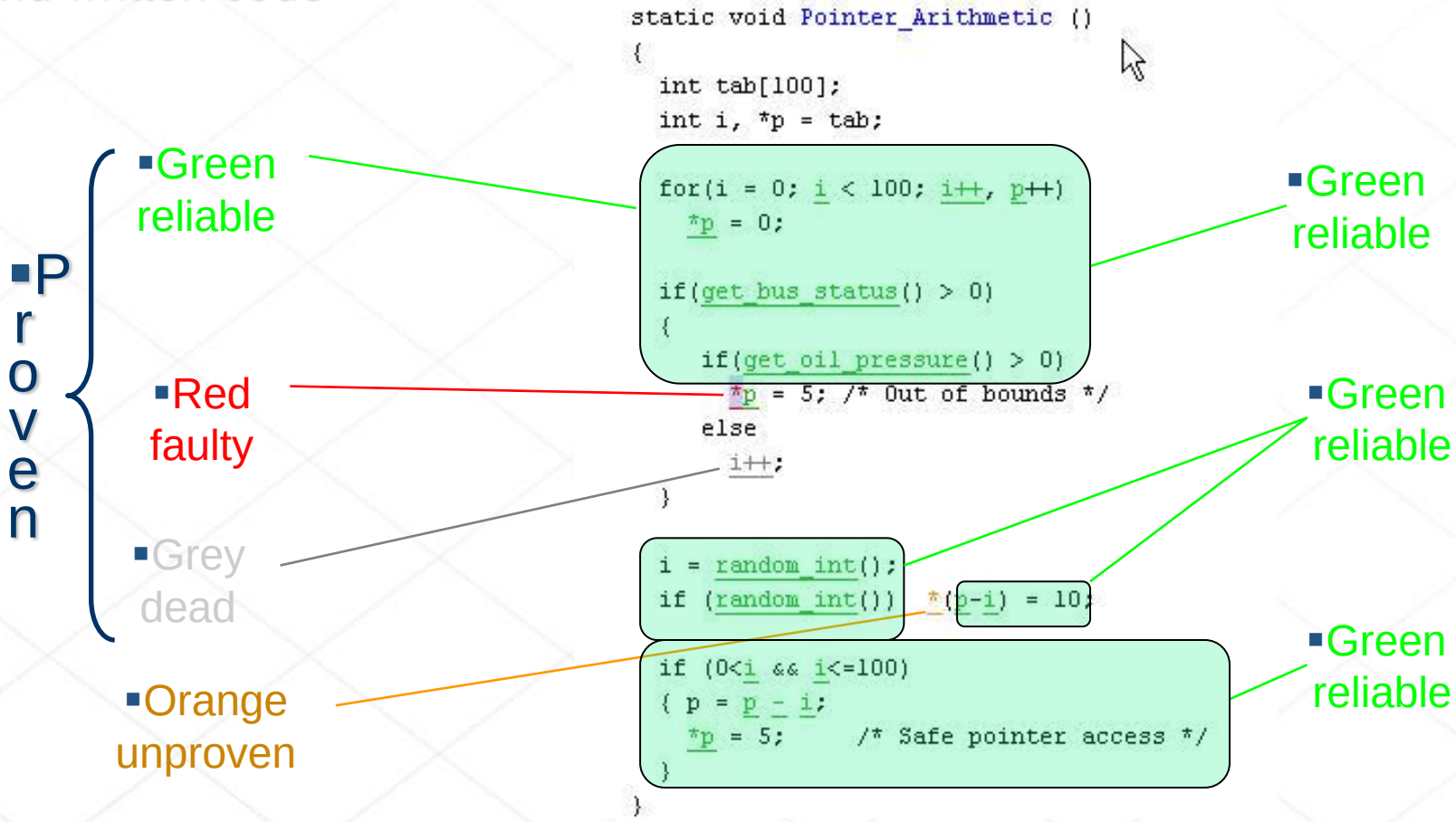
- Read access to non-initialized data
- Out-of-bounds array access
- Dereferencing through null or out-of-bounds pointers
- Illegal type conversion
- Incorrect computation
  - Overflow/Underflow
  - Division by zero
  - Square root of negative value
- And more…



PolySpace™ Solution

For hand written code

Formal method:
Abstract Interpretation

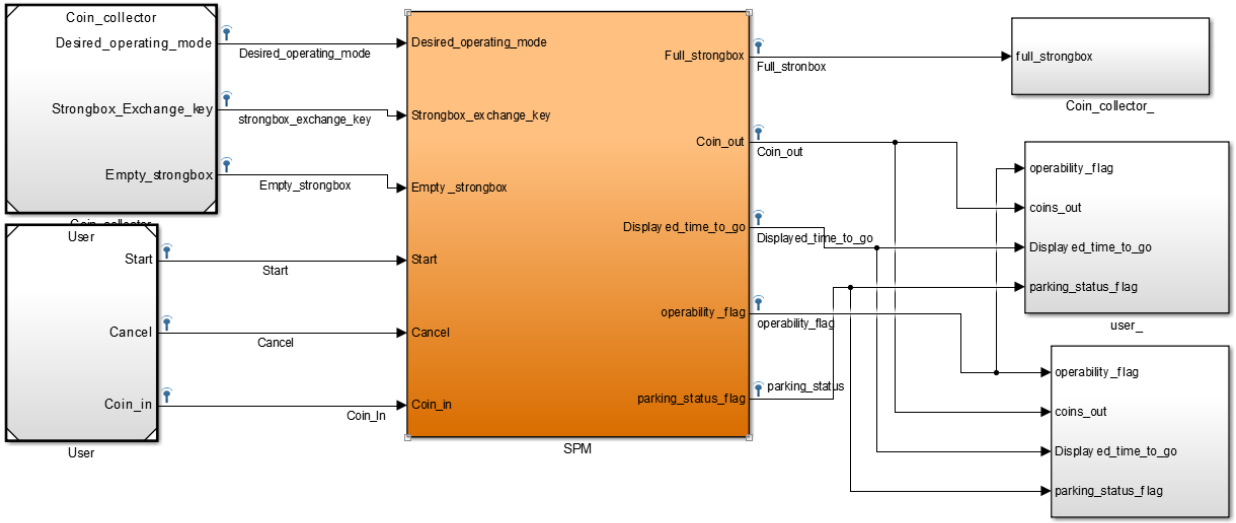


Results are proven for all possible executions of the code!!

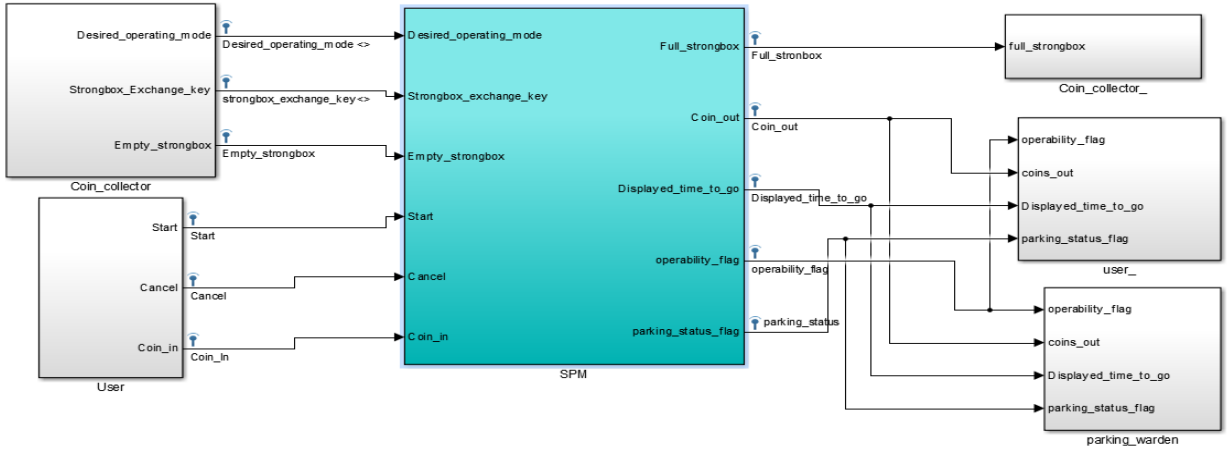
Workflow example for Street Parking Meter (SPM)

System in its Environment

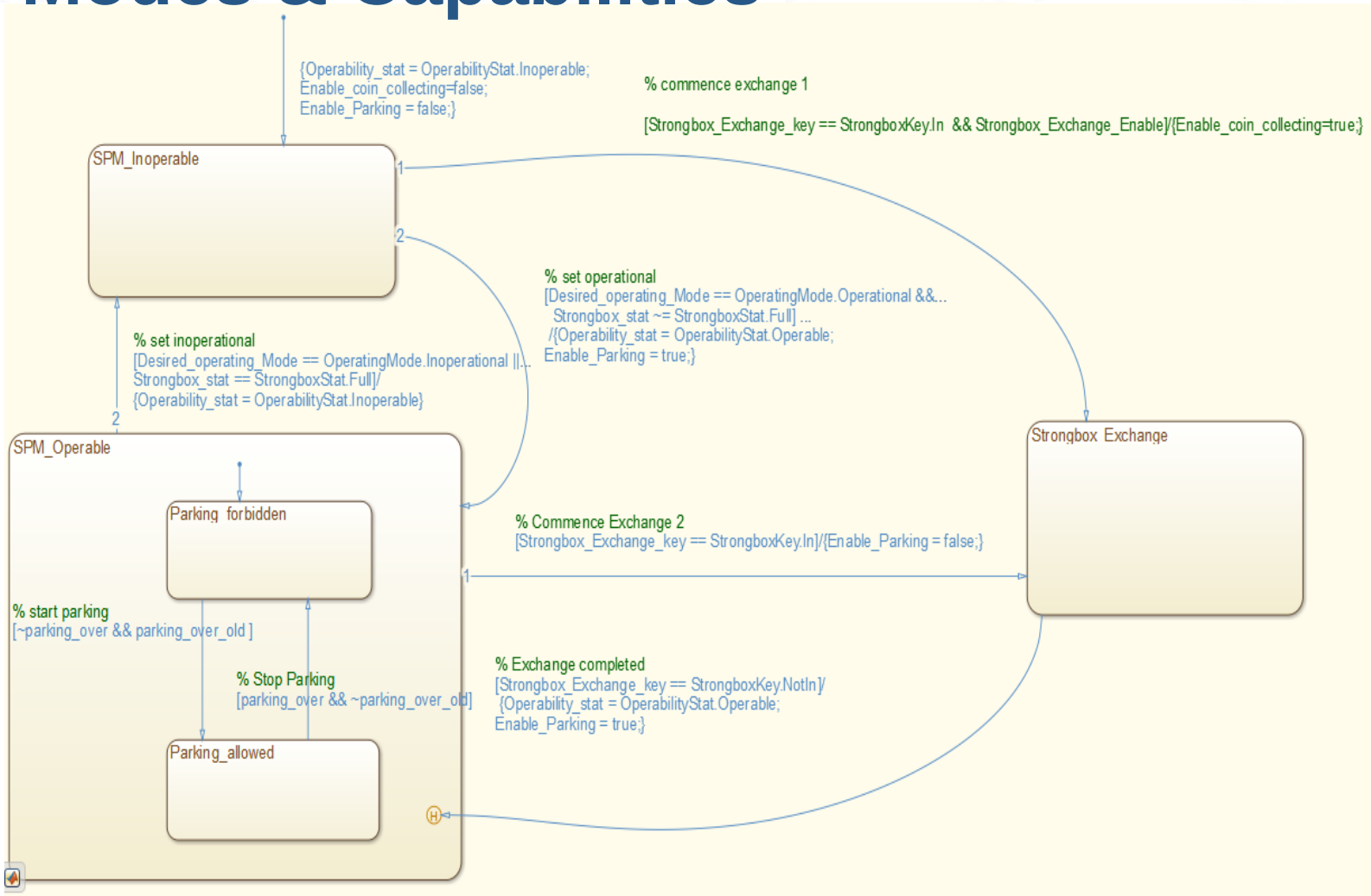
SPM Functionality



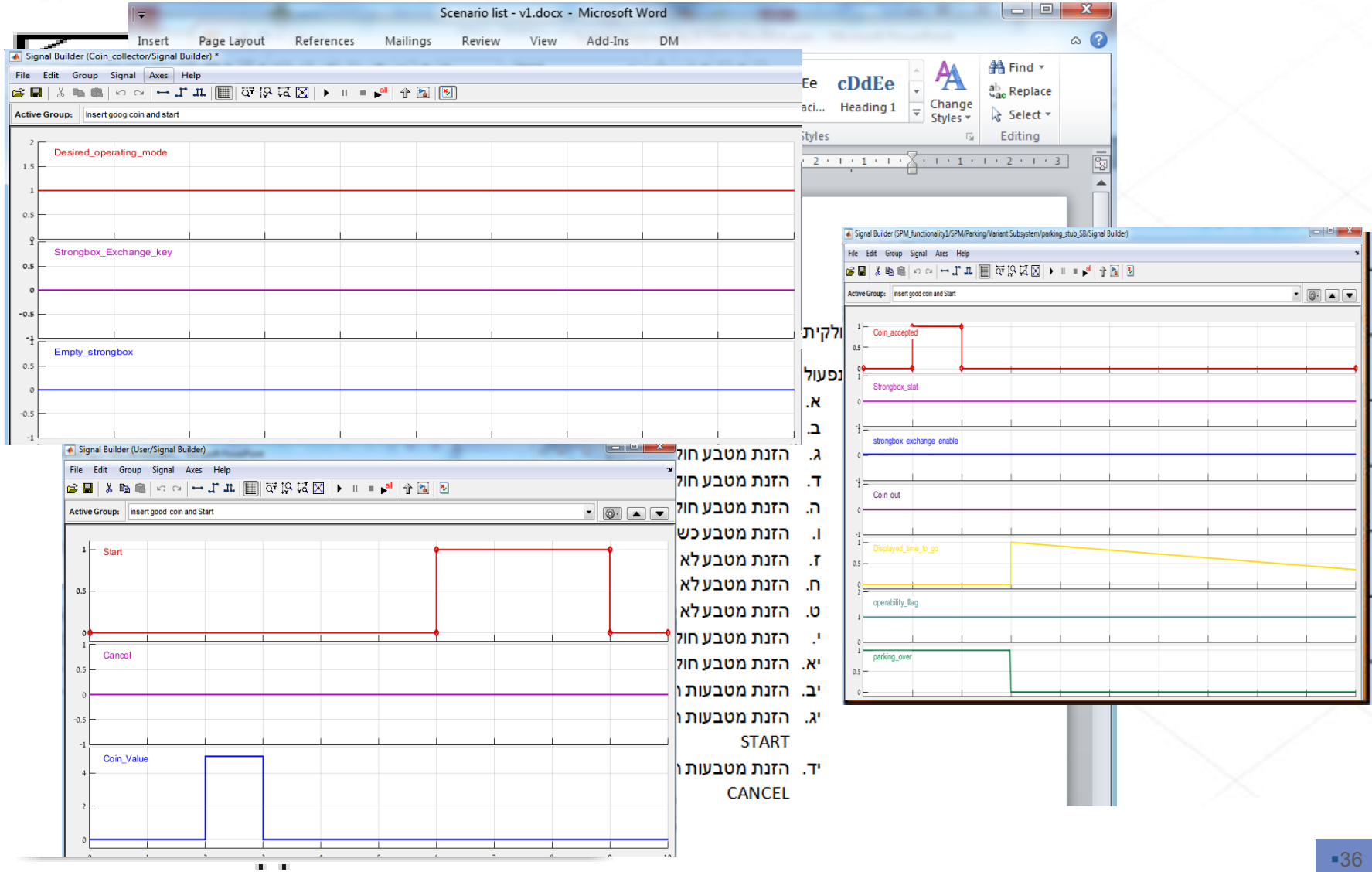
SPM Architecture



Elaborate SPM Functionality Node: Modes & Capabilities



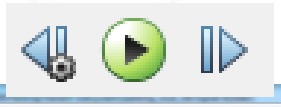
Generate Test Cases and Simulate

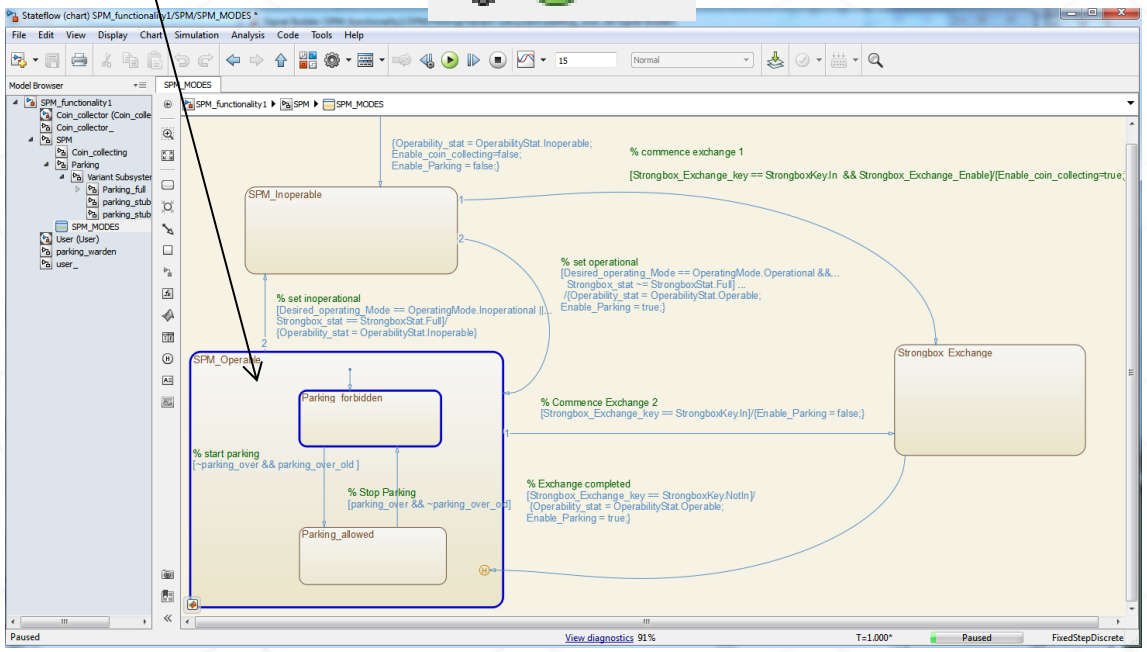


Simulation & Analysis

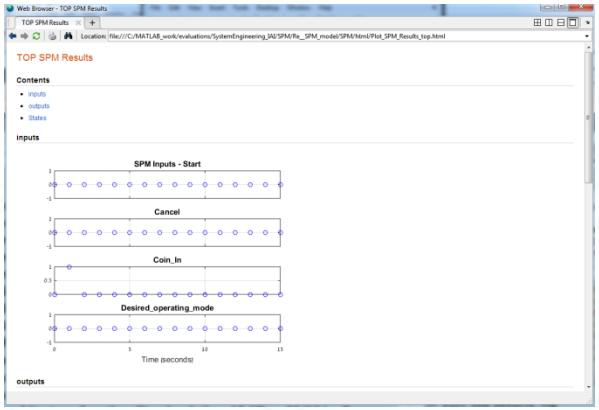
Step Back Step Forward

Stateflow Animation

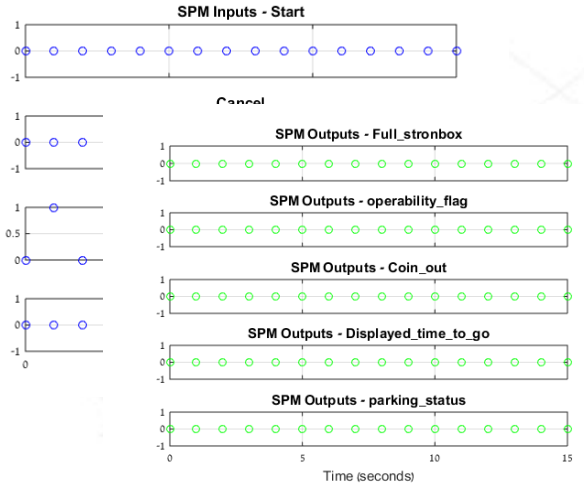




Test Report



MATLAB Graphs

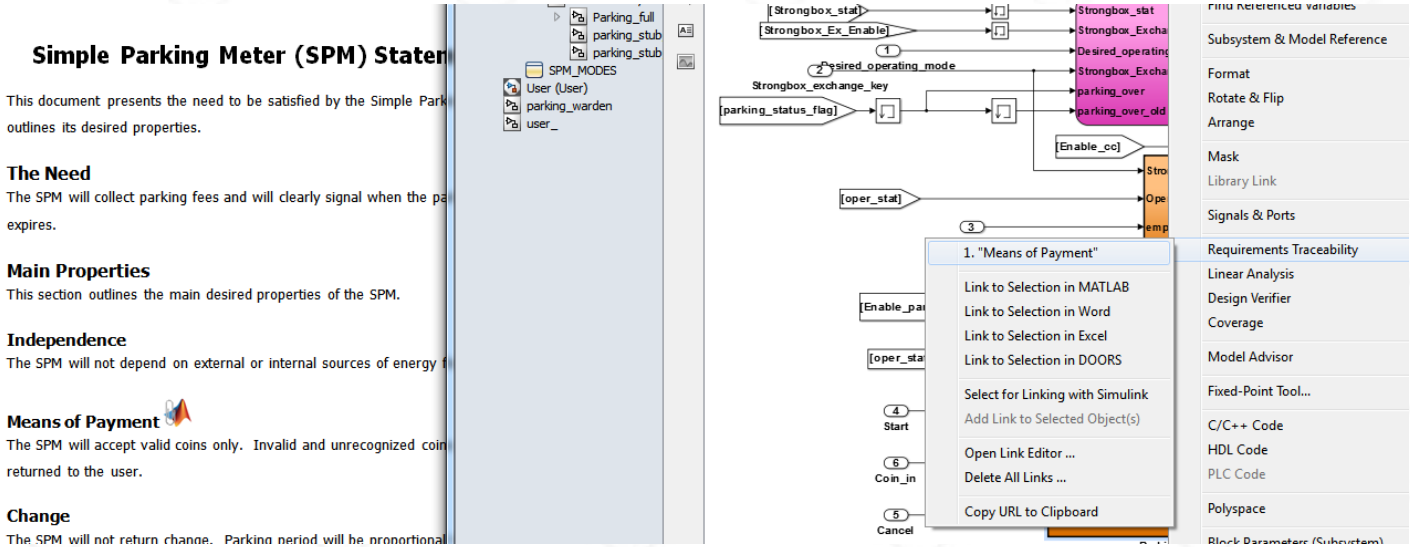


Coverage Report

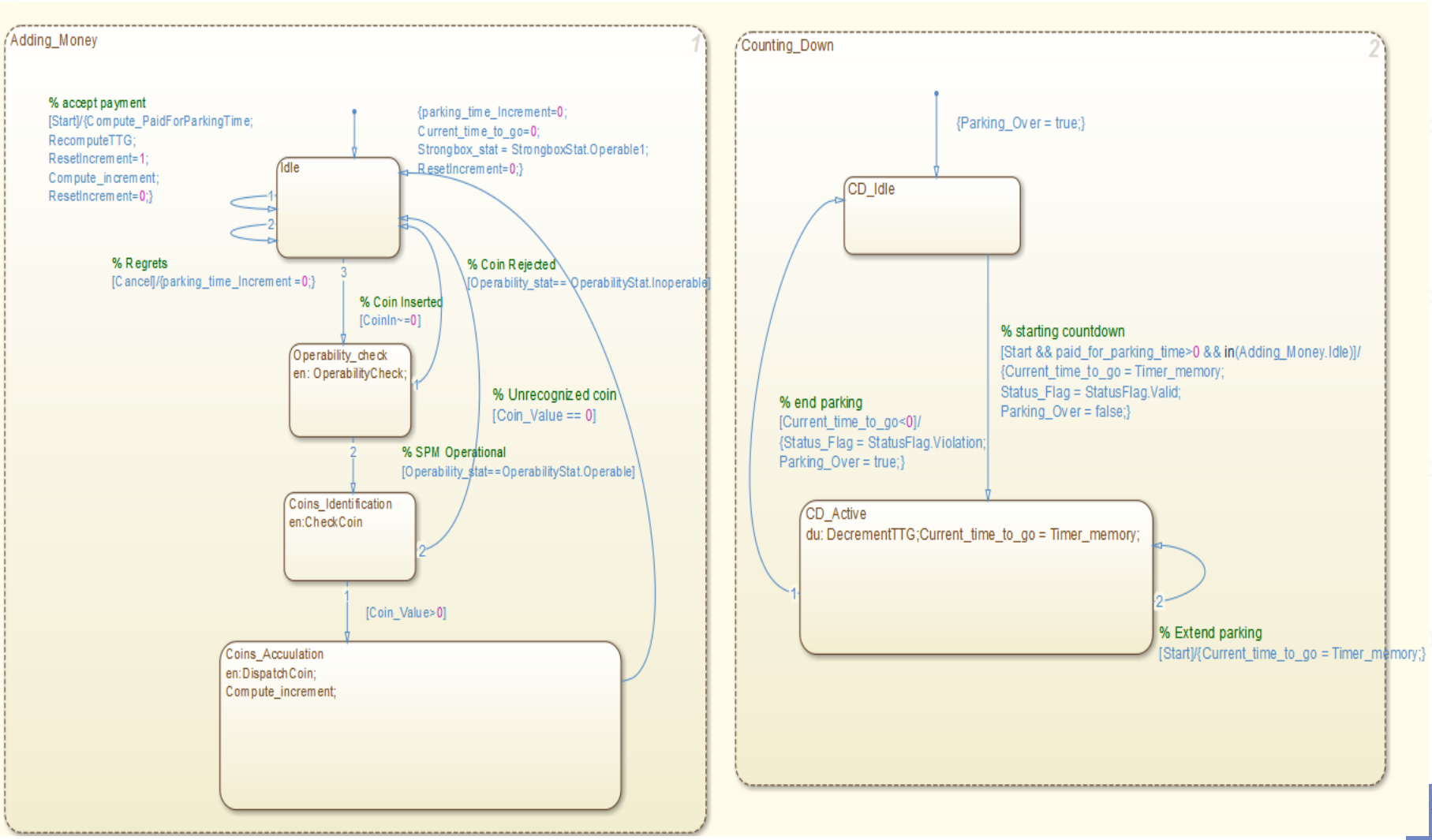
Summary

Model Hierarchy/Complexity		Current Run				Delta				Cumulative			
		D1		C1	MCDC	D1		C1	MCDC	D1		C1	MCDC
1. SPM_functionality1	23	19%		10%	0%	4%		5%	0%	56%		50%	20%
2. . . . SPM	22	19%		10%	0%	4%		5%	0%	56%		50%	20%
3. . . . . Coin_collecting	2	50%		NA	NA	0%		NA	NA	50%		NA	NA
4. . . . . Parking	2	50%		NA	NA	0%		NA	NA	100%		NA	NA
5. . . . . SPM_MODES	18	13%		10%	0%	4%		5%	0%	52%		50%	20%
6. . . . . SF: SPM_MODES	17	13%		10%	0%	4%		5%	0%	52%		50%	20%
7. . . . . SF: SPM_Operable	7	0%		0%	0%	0%		0%	0%	50%		50%	25%

Link To Stake-Holders Requirements and generate System Specification Report



Elaborate Capabilities: states & functions



Integrating Code with Simulink

MATLAB Function

M

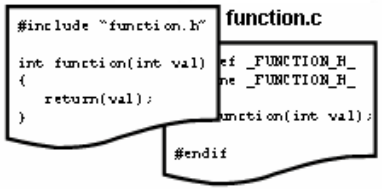
C++

C

Fortran

Ada

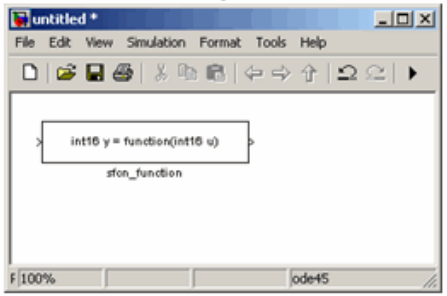
Legacy C code



Legacy Code Tool (LCT)

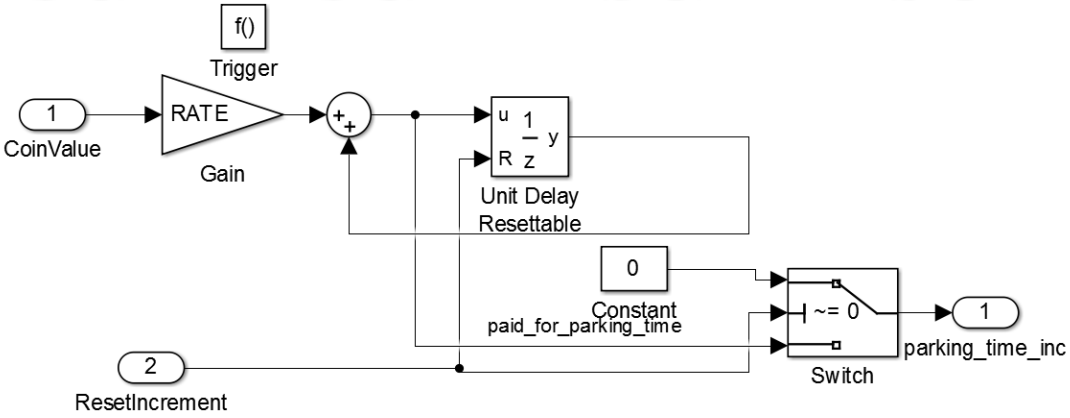
1. Initialize LCT data structure
2. Populate LCT data structure
3. Generate S-function source file
4. Compile S-function source file
5. Create masked S-Function block

C MEX S-function



S-Function block

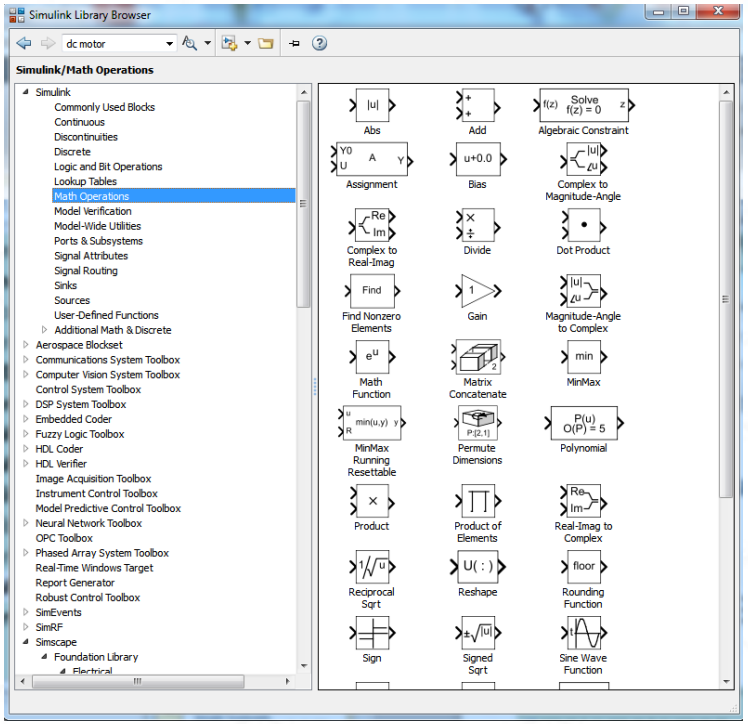
Simulate functionality



Compute_TimeInc

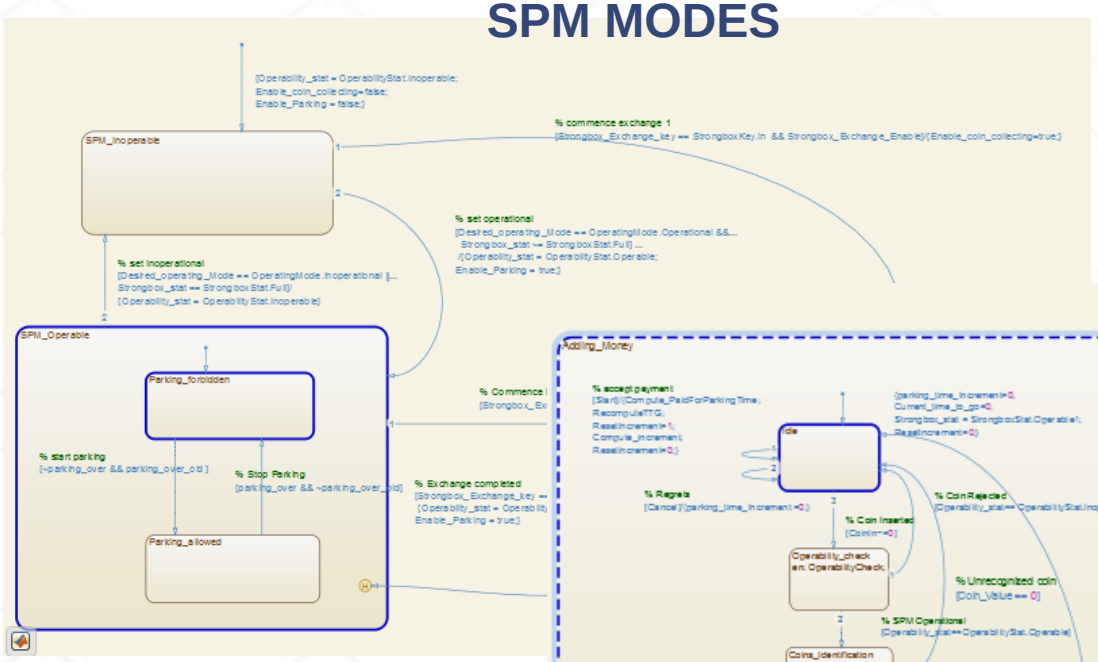
```
function [coin_value,good_coin,bad_coin] = check_coin(coin)
%#codegen

if coin == 1 || coin == 5 || coin == 10
    coin_value = coin;
    good_coin = coin;
    bad_coin = 0;
else
    coin_value = 0;
    good_coin = 0;
    bad_coin = coin;

end
```

check_coin

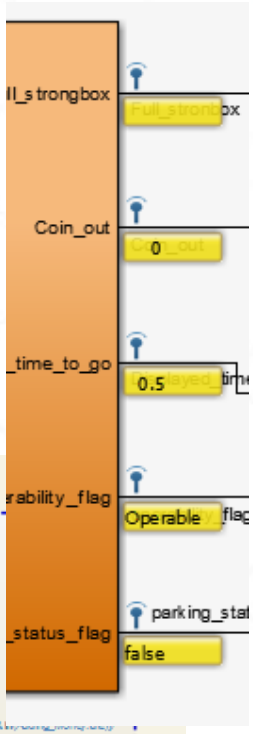


Simulate Capability (Parking)

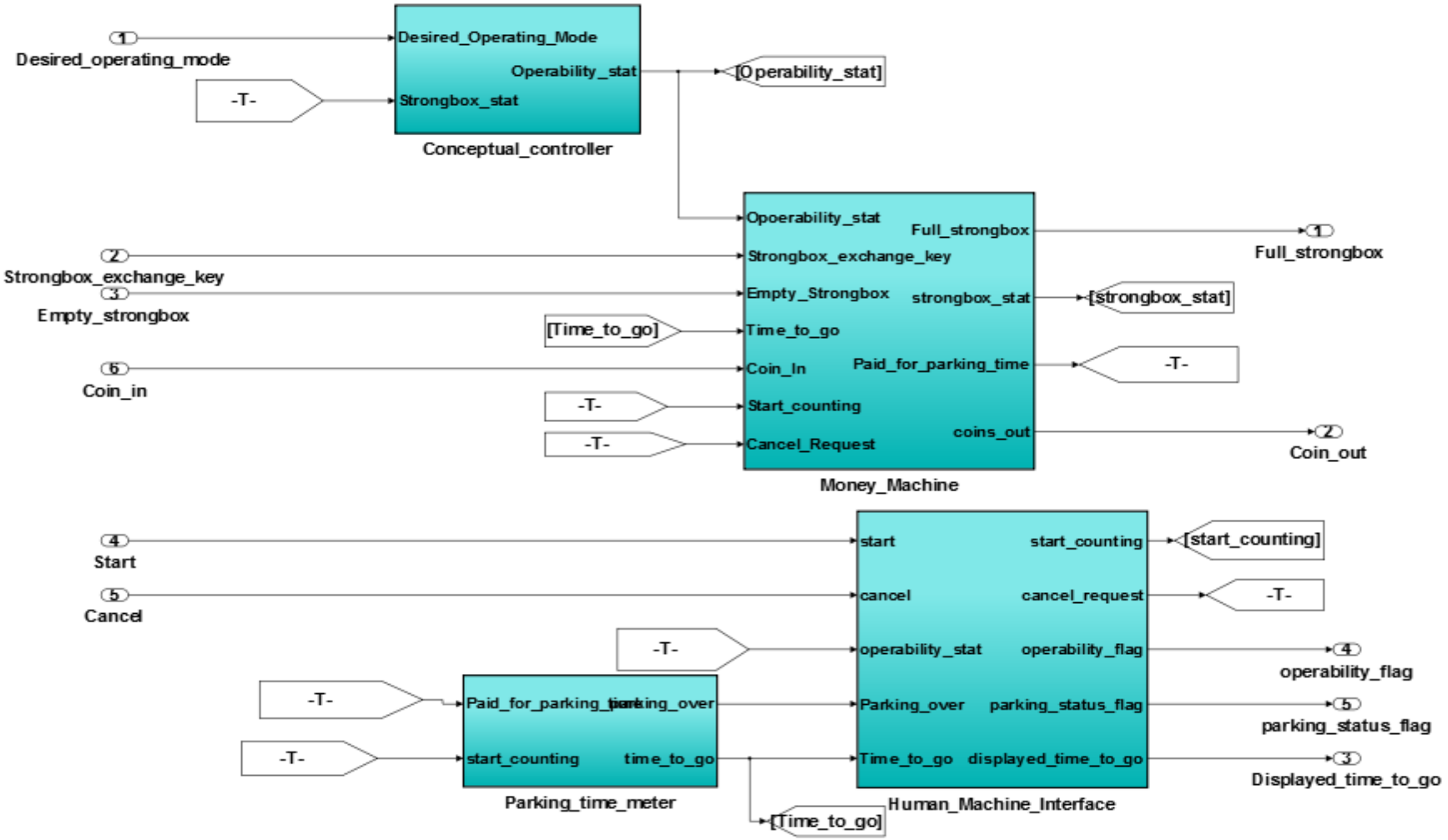
SPM MODES



Parking CTRL

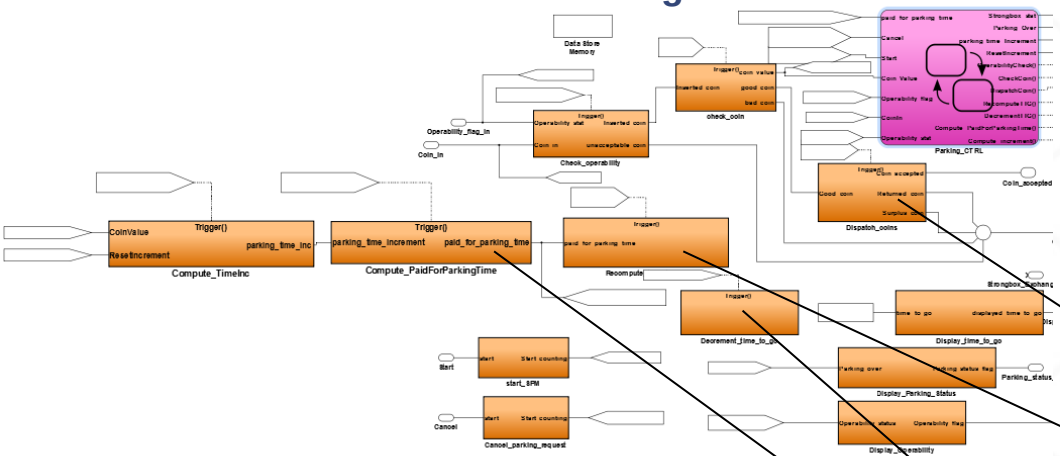


Elaborate physical subsystems

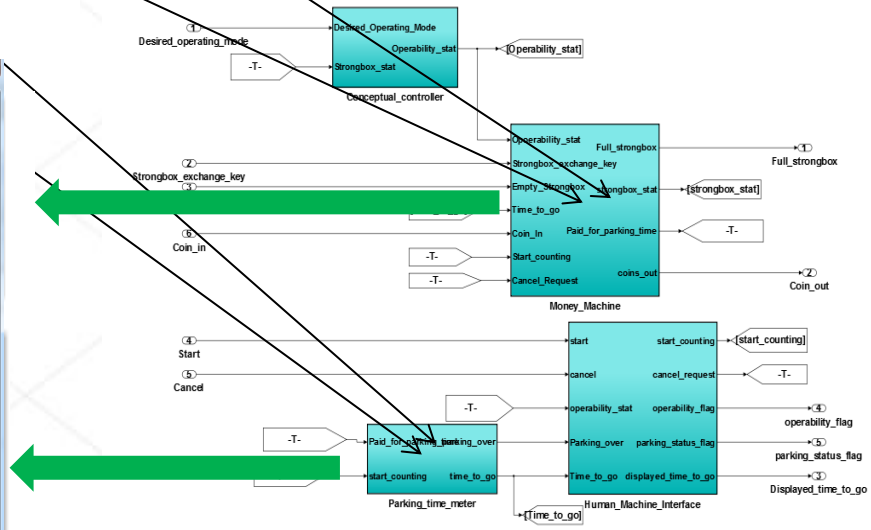


Allocate functions to subsystems

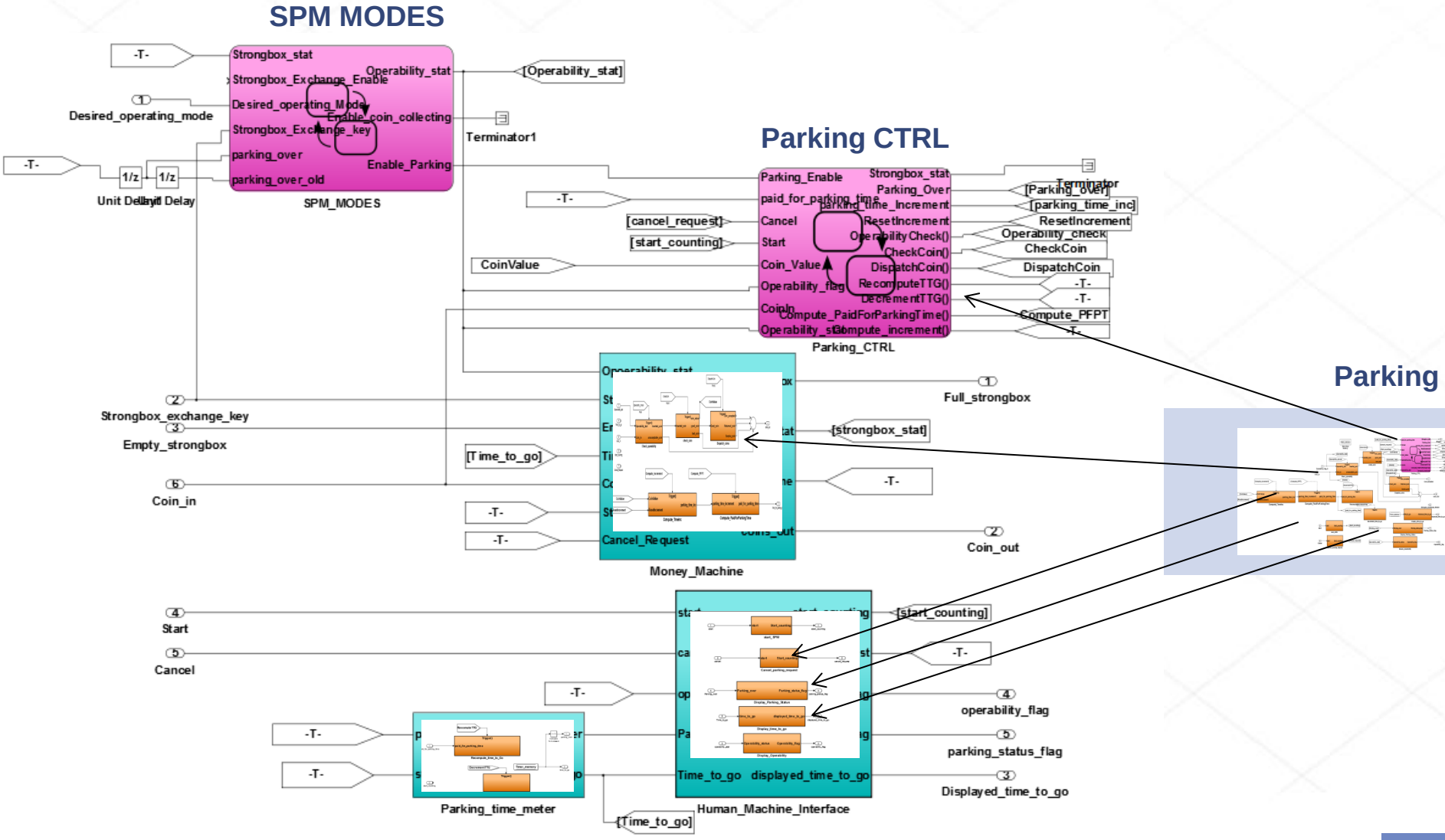
Parking



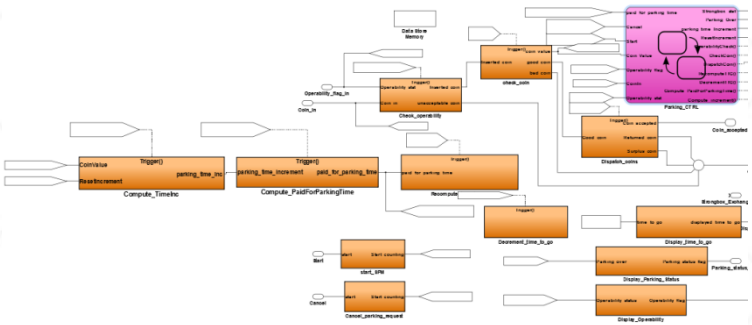
RMI Linking between Simulink Models

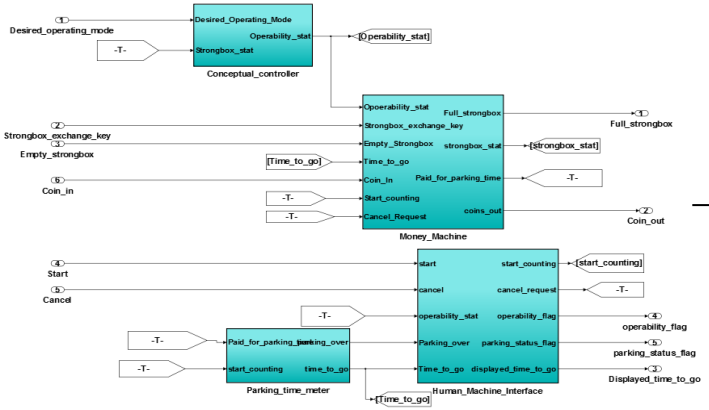


Validate allocation by Simulation(Optional)



Allocate Subsystems Requirements for Subsystems and Generate SSDD report





Simple Parking Meter (SPM) Statement of Need

This document presents the need to be satisfied by the Simple Parking Meter (SPM) and outlines its desired properties.

The Need

The SPM will collect parking fees and will clearly signal when the parking period paid for expires.

Main Properties

This section outlines the main desired properties of the SPM.

Independence

The SPM will not depend on external or internal sources of energy for its operation.

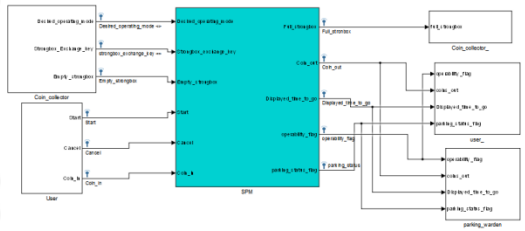
Means of Payment

The SPM will accept valid coins only. Invalid and unrecognized coins will be rejected, and returned to the user.

SSDD Report

Chapter 2. Root System

Figure 2.1. SPM_architecture

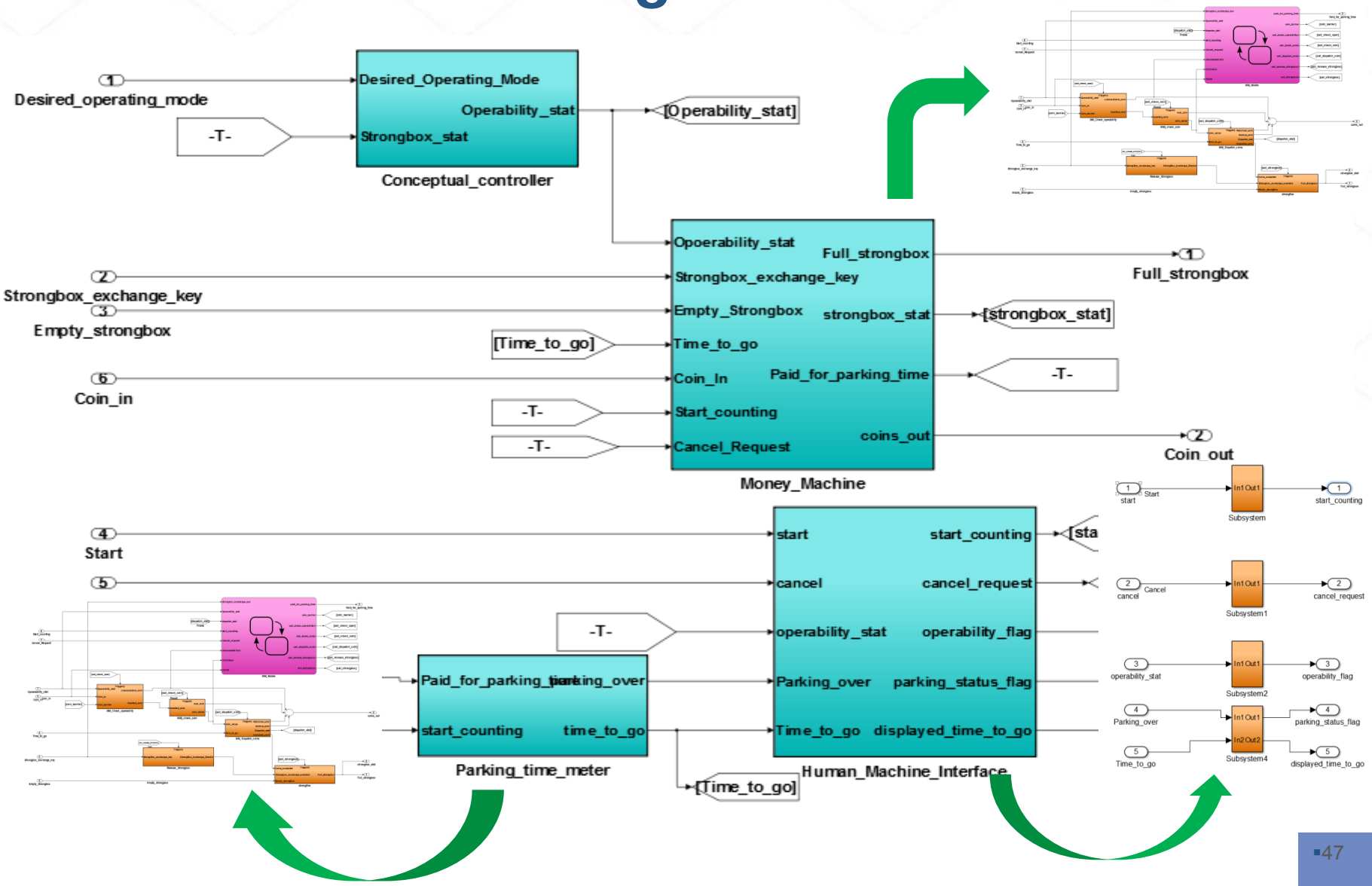


Blocks

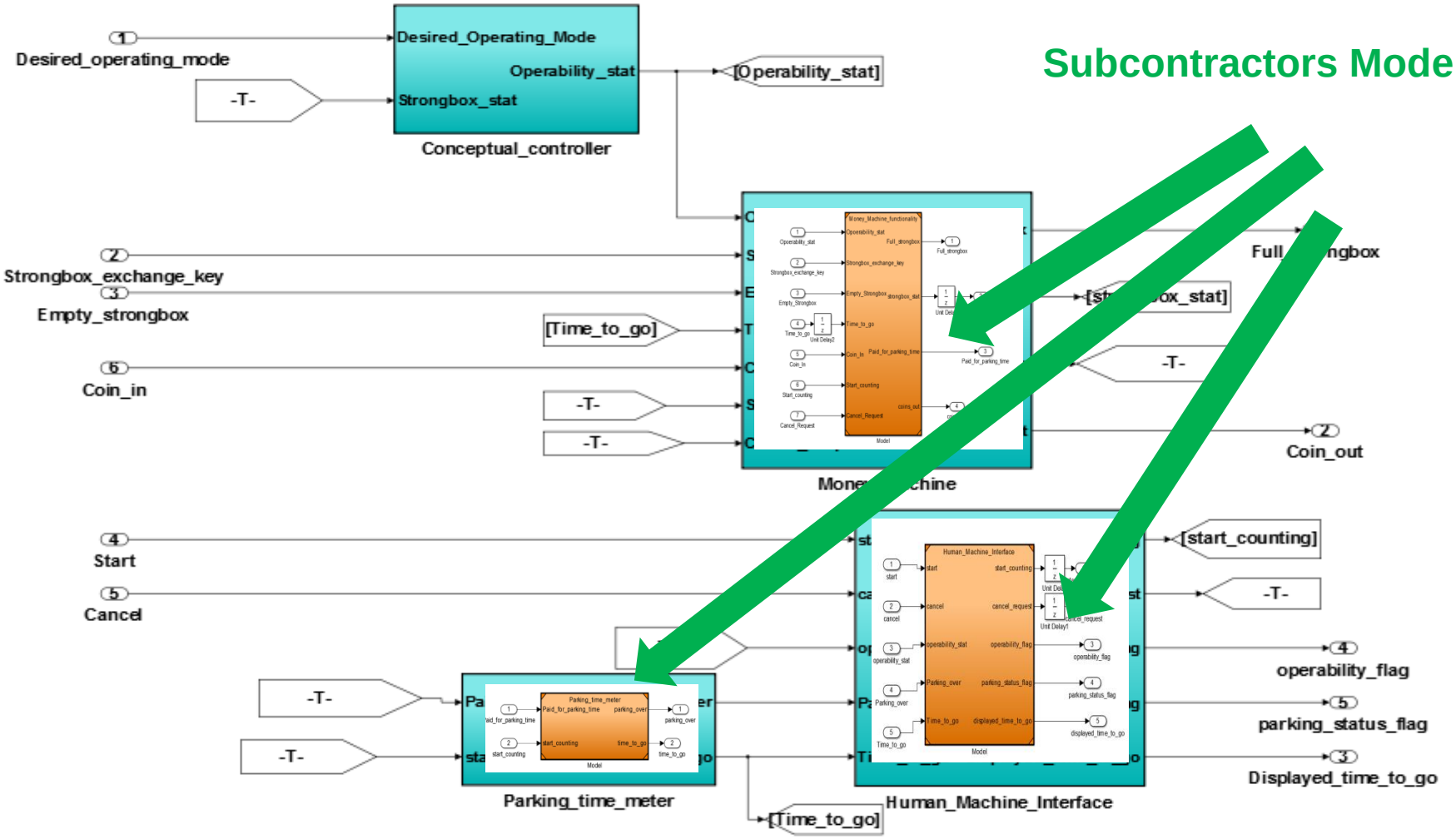
Parameters

Block Execution Order

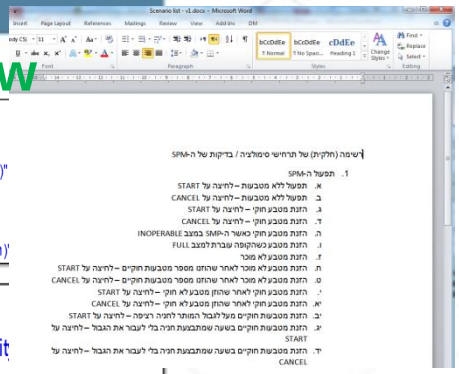
Subcontractors design models



Models Integration – System level simulation



Requirements View



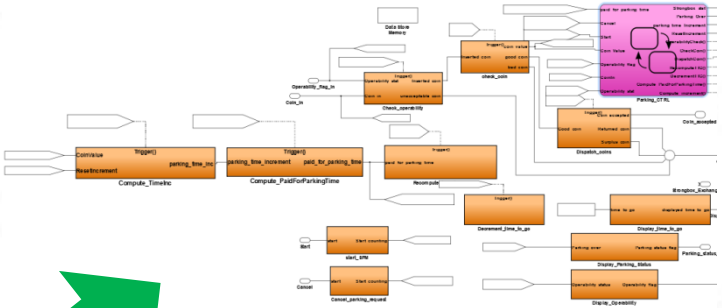
System Requirements

1. "SPM_functionality1/.../Parking_full/Display_time_to_go (SubSystem)"
2. "SPM_functionality1/.../Parking_full/Display_Parking_Status (SubSystem)"
3. "SPM_functionality1/.../Parking_full/Display_Operability (SubSystem)"
4. "SPM_functionality1/.../Parking_full/start_SPM (SubSystem)"
5. "SPM_functionality1/.../Parking_full/Cancel_parking_request (SubSystem)"

System Requirements

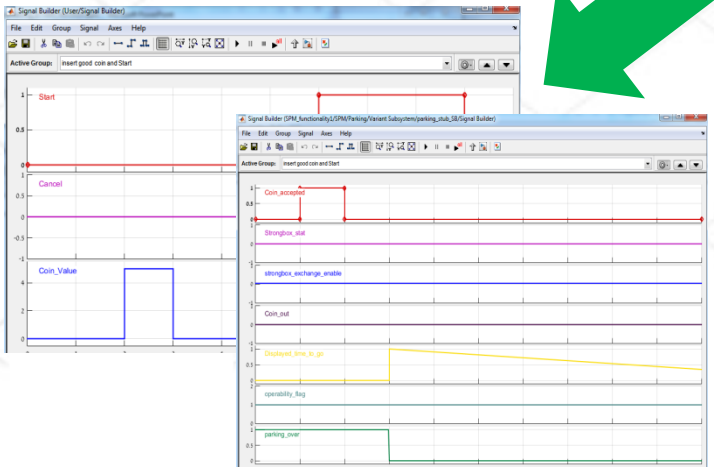
1. "SPM_functionality1/.../Parking_full/Check_operabilit
2. "SPM_functionality1/.../Parking_full/check_coin (SubSystem)"
3. "SPM_functionality1/.../Parking_full/Dispatch_coins (SubSystem)"

Behavior View

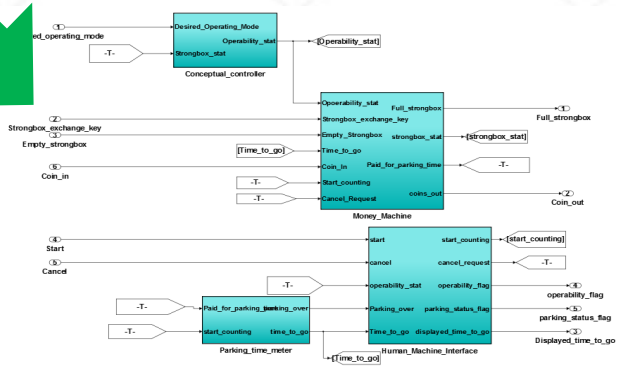


MODEL

Test View



Architecture View



Workflow Summary

- System in its Environment diagrams (functionality & architecture)
- Elaboration of top functionality diagram - Modes & Capabilities
- Validate by simulation of top functionality Node
- Link to SH Requirements and generate System Specification report
- Elaborate Capabilities – Capabilities States & functions
- Validate by Capabilities simulation
- Elaborate top level Architecture to physical subsystems
- Allocate capabilities functions to physical subsystems

Workflow Summary - Continue

- Validate allocation by simulation (optional)
- Assign subsystem requirements and generate SSDD report
- Use subcontractors to design subsystems – receive models from subcontractors
- Integrate subcontractors models to build system level simulation and validate design

Conclusion

- Model-Based system Engineering (MBSE) is a proven methodology for reducing errors during the early requirements phase and therefore save time & money
- Mathworks have a set of tools based on Simulink, to implement Model-Based Design(MBD) paradigm
- The 2 paradigms are similar so it is possible to use the same Simulink based tools also for MBSE
- The big advantage of using Simulink is its strong simulation capabilities, simulation is a very strong tool for validating design.