Guest Lecture: Antwerpen

Gareth Thomas

Rachid Adarghal

Focal Points

- With MATLAB/Simulink my professors think I am clever
- Knowing MATLAB/Simulink will help you get a job!
- Multi-Modeling Techniques are often needed

Motivation

- With MATLAB/Simulink my professors think I am clever
  - **The tools will make your life easier.**
- Knowing MATLAB/Simulink will help you get a job!
  - **Put it on your CV, as you will encounter this after your degree**
- Multi-Modeling Techniques are often needed
  - **The real world is so complex, the solution comes from combining multiple domains**

Agenda

Power Window – Example of Model Based Design

Some Modeling Tools from MathWorks

- MATLAB - Textual
- Simulink - Blocks
- StateFlow - States
- State Transition - Tables
- Simscape - Physical Modeling

Introduction to Verification and Validation

- Test Generation
- Coverage
- Counter Examples
- Model Transformation is Key

Tools in Industry

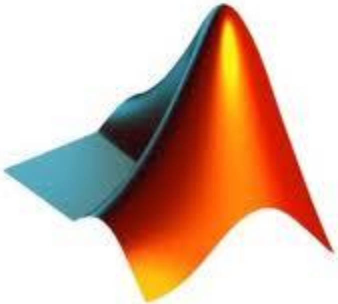
Introduction to the Speaker

Gareth Thomas

- Masters in Electronic Engineering at Instituto Superior Técnico
  Control Theory and Signal Processing
- Consultant at Altran CIS in Portugal
- Innovation Officer at Nokia Siemens Networks in Portugal
- Software Engineer at Oceanscan in Scotland
- Application Engineer at Mathworks Benelux

MathWorks Products

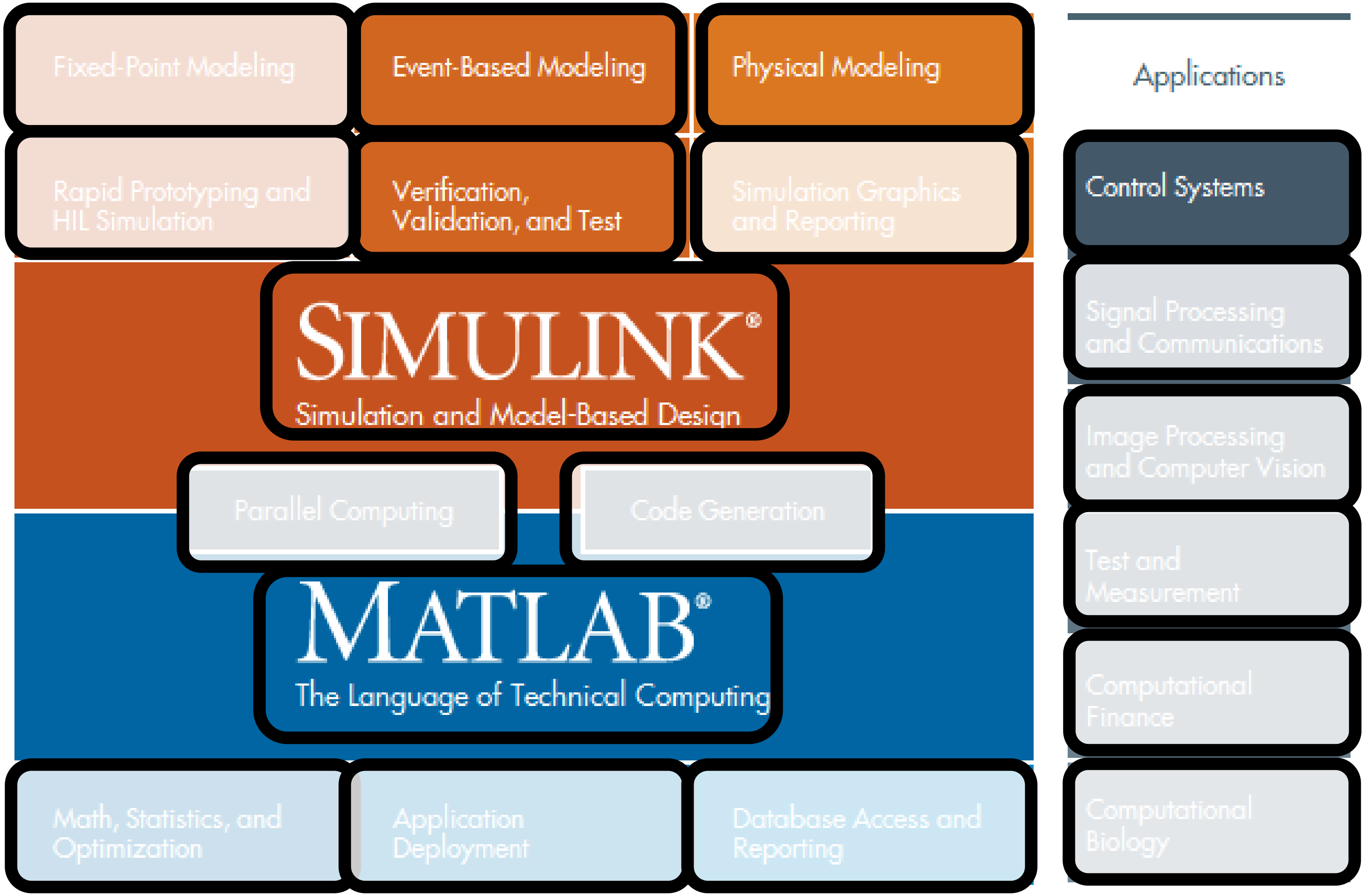
- How Many Toolboxes/Blocksets do you use?
  - 1 – 10
  - 10 – 15
  - 15 – 30
  - 30 – 60
  - >60



- How many toolboxes/Blockset do we offer?
  - 30 – 40
  - 40 – 50
  - 50 – 60
  - 60 – 70
  - >70

92!

MathWorks Products



Fixed-Point Modeling

Event-Based Modeling

Physical Modeling

Rapid Prototyping and HIL Simulation

Verification, Validation, and Test

Simulation Graphics and Reporting

SIMULINK®

Simulation and Model-Based Design

Parallel Computing

Code Generation

MATLAB®

The Language of Technical Computing

Math, Statistics, and Optimization

Application Deployment

Database Access and Reporting

Applications

Control Systems

Signal Processing and Communications

Image Processing and Computer Vision

Test and Measurement

Computational Finance

Computational Biology

MathWorks Vital Statistics

Developers of MATLAB & Simulink

2,800 staff worldwide

Support staff worldwide

Development staff in Natick, MA

30% of revenue invested in R&D

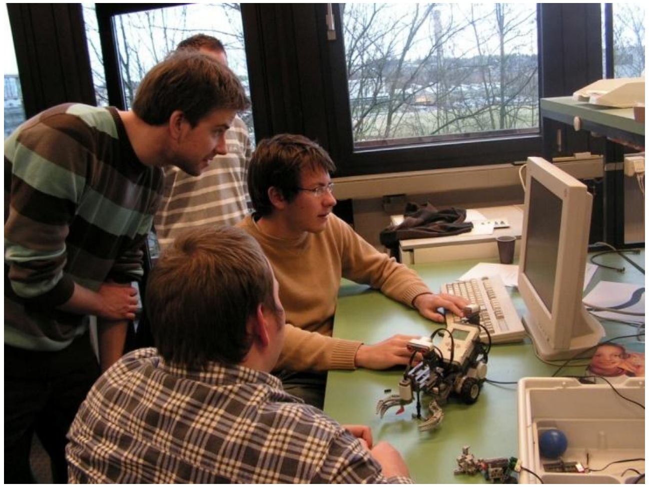
$500M annual revenue

__2009 -__ orders from
23,000 companies
in 128 countries

Revolution in Engineering Education

5000+ universities worldwide use MATLAB

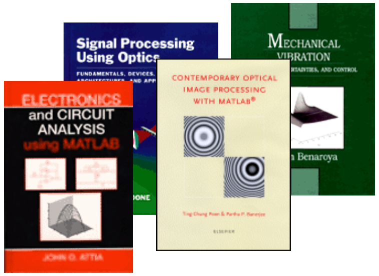
- > *Includes all of the Top 200 World Universities**



More than 1 million students and faculty have access to MathWorks tools through campus-wide licenses

- > *More than 130 academic institutions, including 10 of the Top 20 World Universities**

Over 1400 MATLAB based books in 27 languages


Signal Processing Using Optics
Mechanical Vibration
Contemporary Optical Image Processing with MATLAB
Electronics and Circuit Analysis using MATLAB

* Source: Times Higher Education-QS World University Rankings 2009

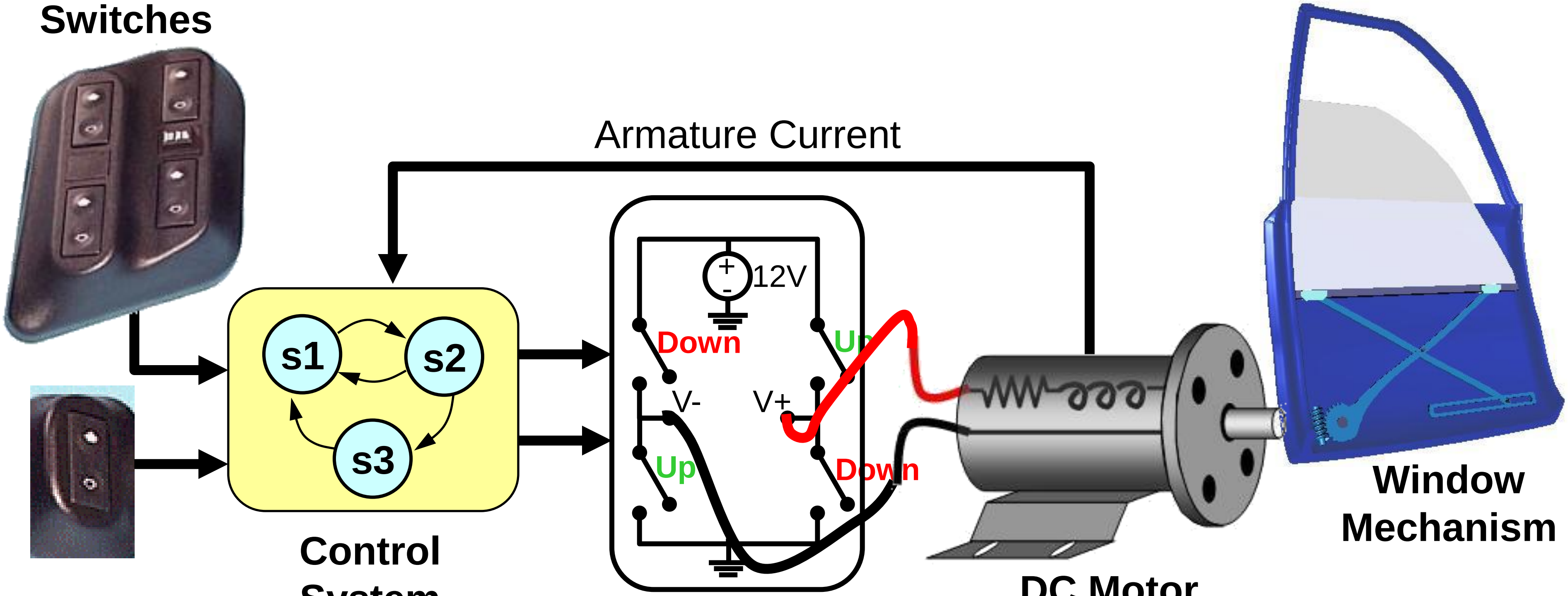
Model-Based Design

Power Window Video

Power Window System

Switches

Armature Current



Control System

H-Bridge

DC Motor

Window Mechanism

Stateflow

SimMechanics

SimElectronics

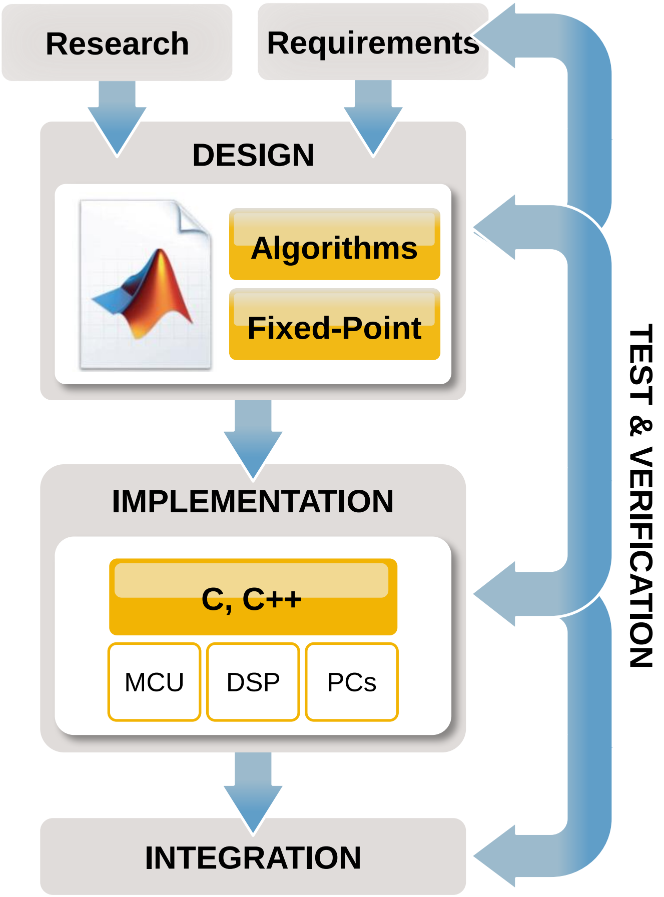
SimElectronics

SimElectronics

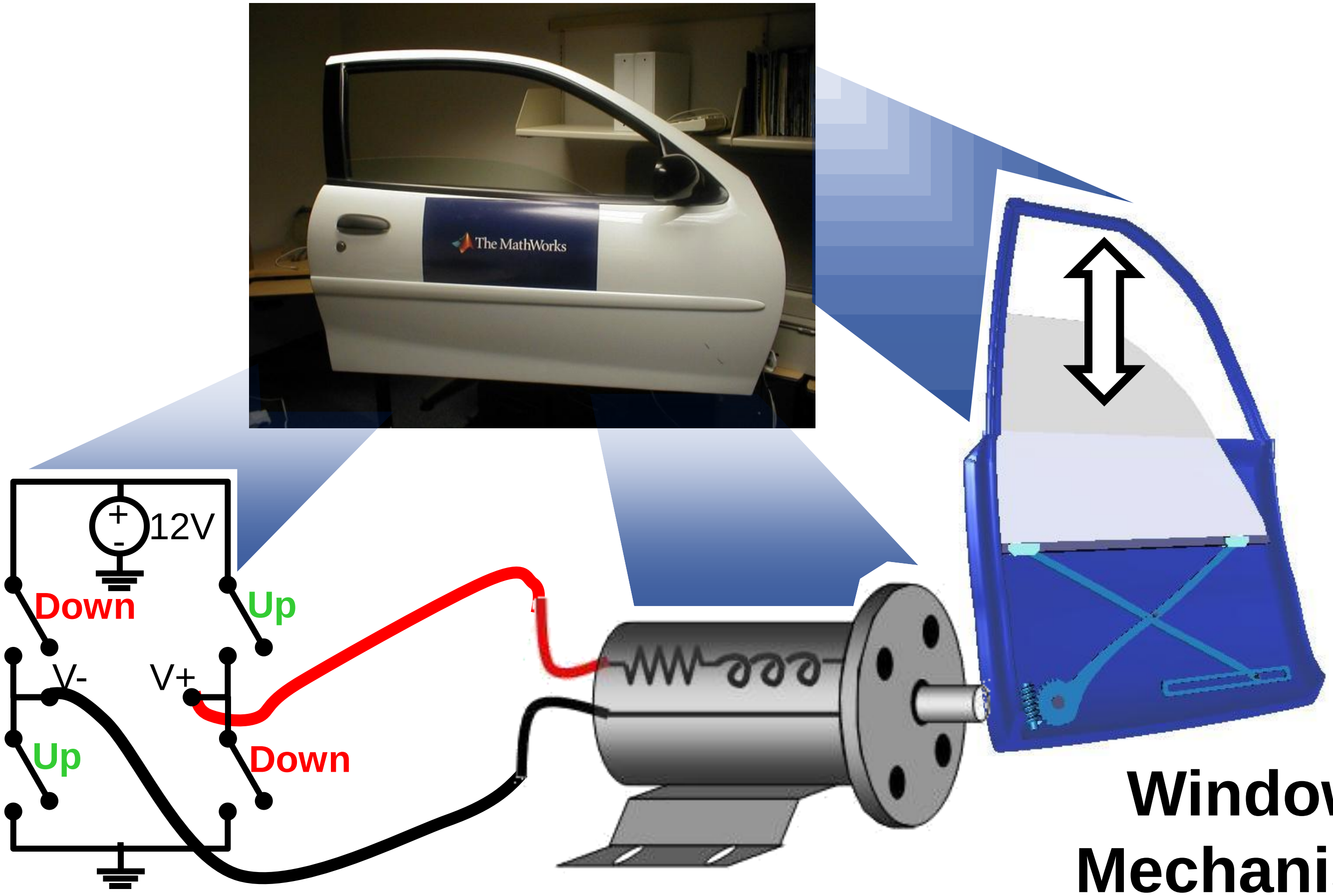
Steps Taken

- Define Problem (Requirements)
- Model Plant (window)
- Model Controller
- Test System - Simulation
- Generate C-Code, implement it
- Test System – Real Window

- *Model-Based Design*



Power Window: Modeling the Plant



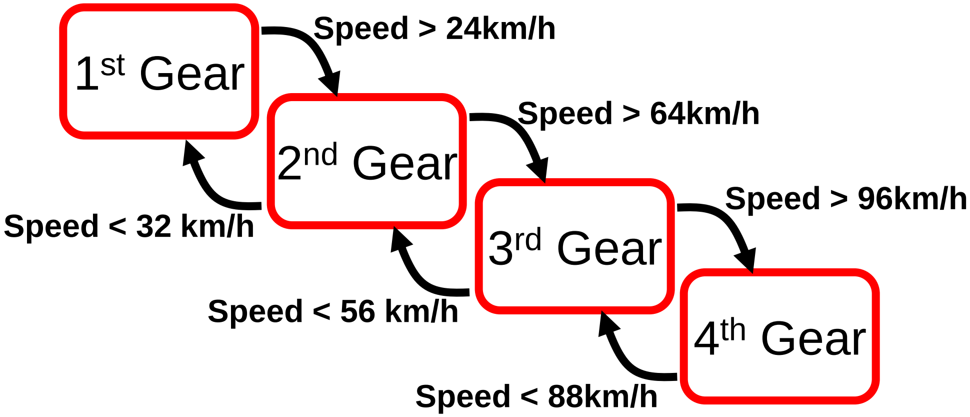
H-Bridge

DC Motor

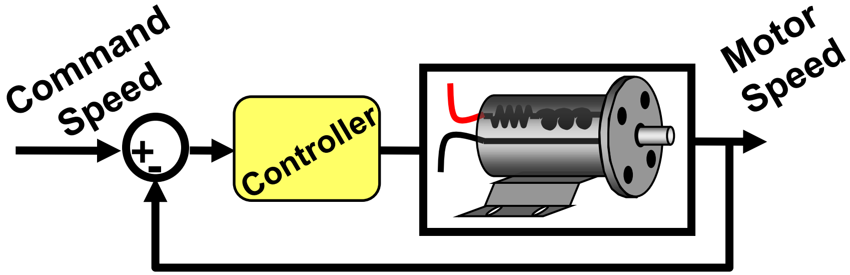
Window Mechanism

Power Window: Defining the Controller

- Event-Based Control
  - For systems that change mode based on events
  - Examples
    - Automatic transmission
    - Power window
  - Best modeled in Stateflow

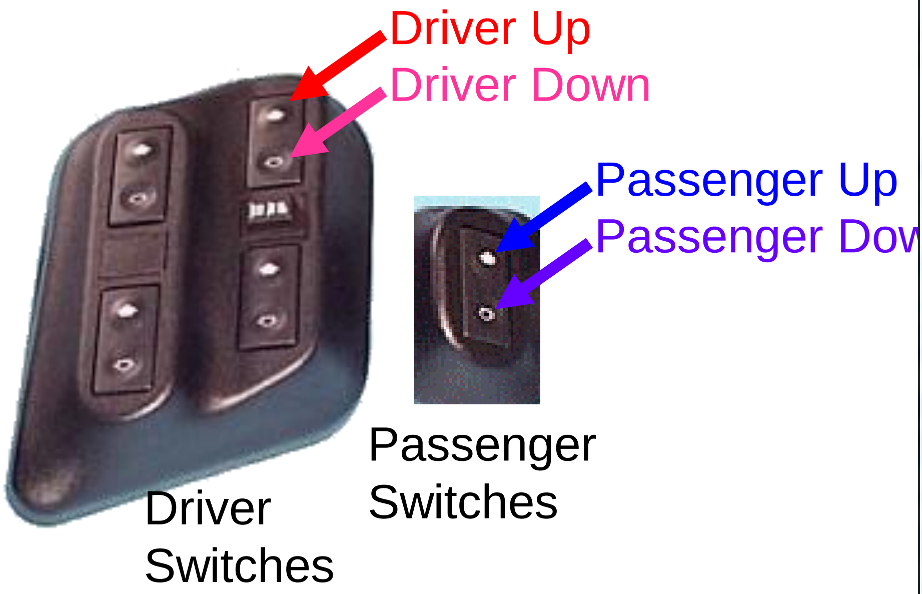


- Compensator Design
  - For systems where actuation is based on deviation from a commanded value (e.g. PID)
  - Examples
    - Robot position
    - Motor speed
  - Best modeled with Simulink Control Design and other control design tools

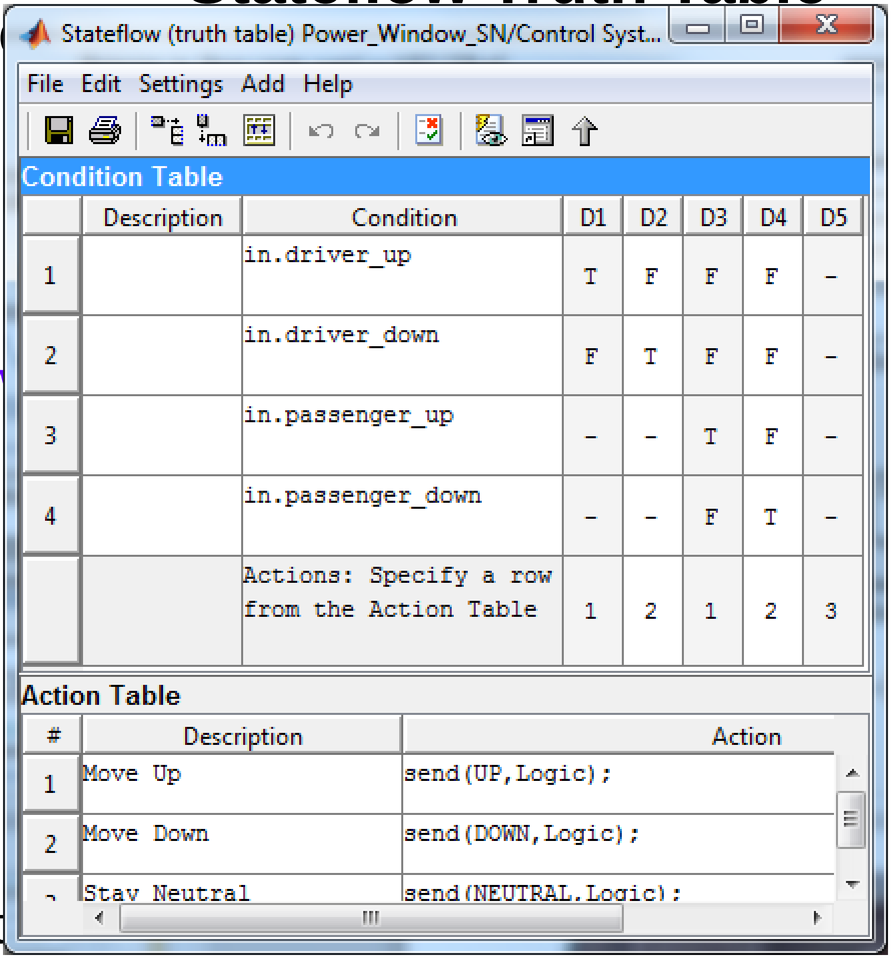


Defining the Controller: Inputs

- Input to controller are switch



Stateflow Truth Table



Stateflow (truth table) Power_Window_SN/Control Syst...
File Edit Settings Add Help

Condition Table

	Description	Condition	D1	D2	D3	D4	D5
1		in.driver_up	T	F	F	F	-
2		in.driver_down	F	T	F	F	-
3		in.passenger_up	-	-	T	F	-
4		in.passenger_down	-	-	F	T	-
		Actions: Specify a row from the Action Table	1	2	1	2	3

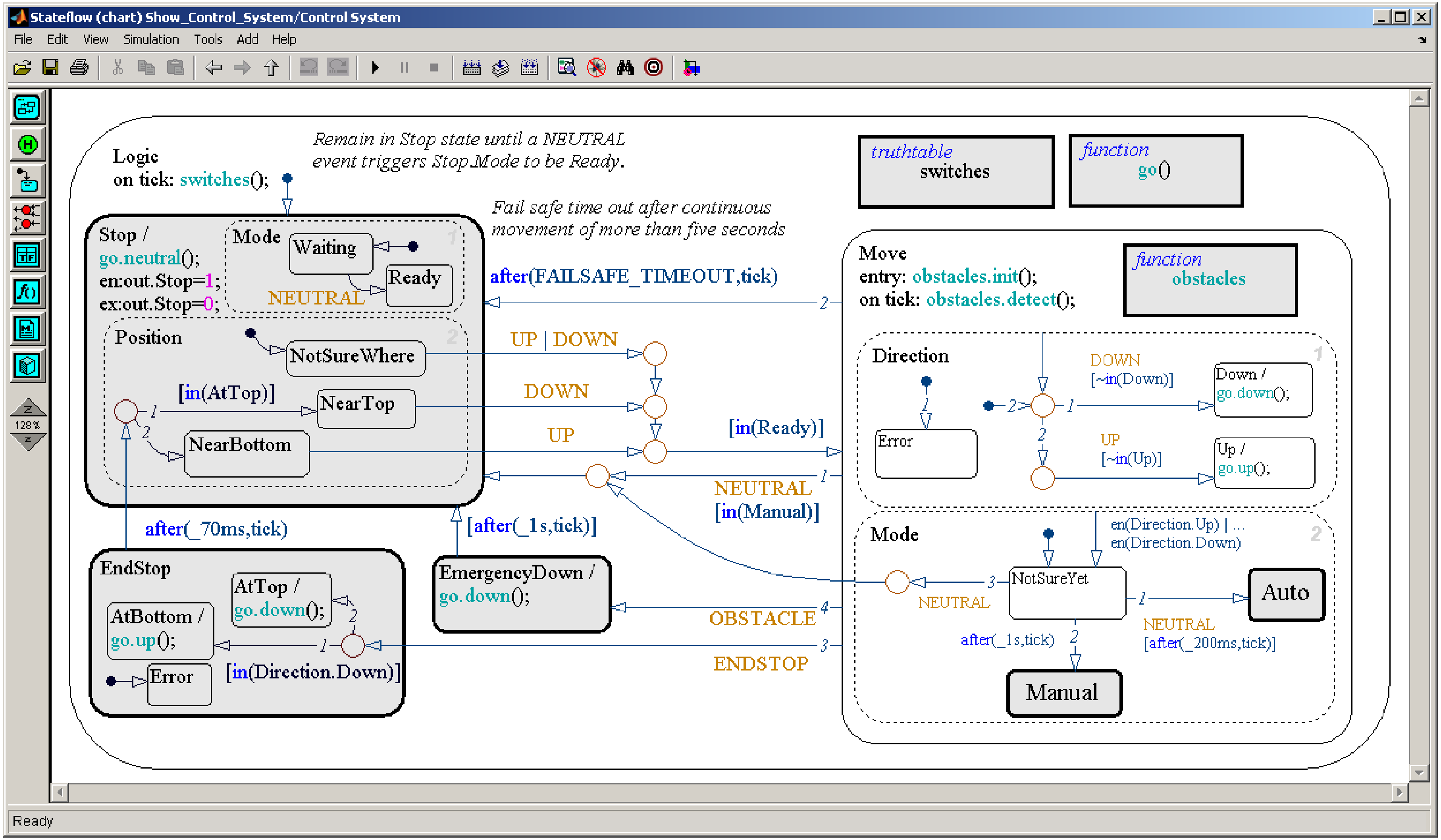
Action Table

#	Description	Action
1	Move Up	send(UP,Logic);
2	Move Down	send(DOWN,Logic);
3	Stay Neutral	send(NEUTRAL,Logic);

1) Driver's side switch has precedence over passenger switch
2) If no switches are closed, movement window is defined based on history

Defining the Controller: States

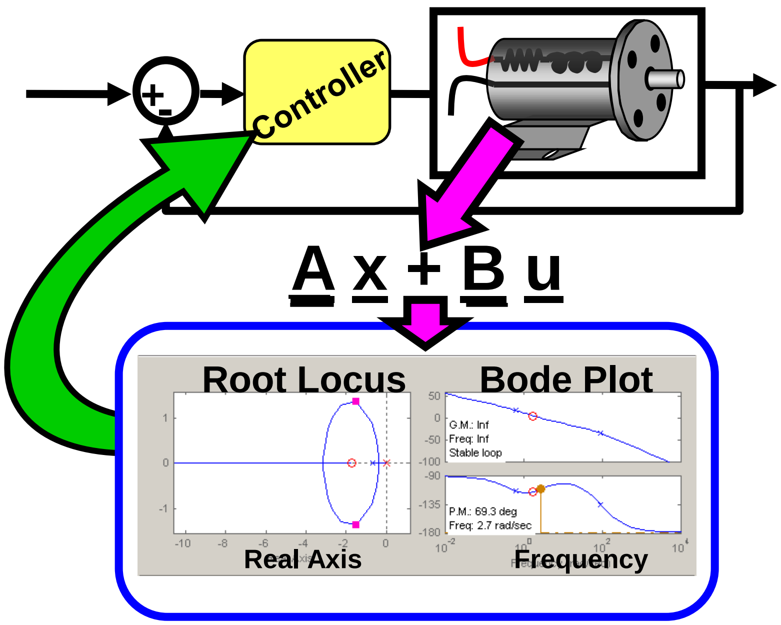
Stateflow Chart

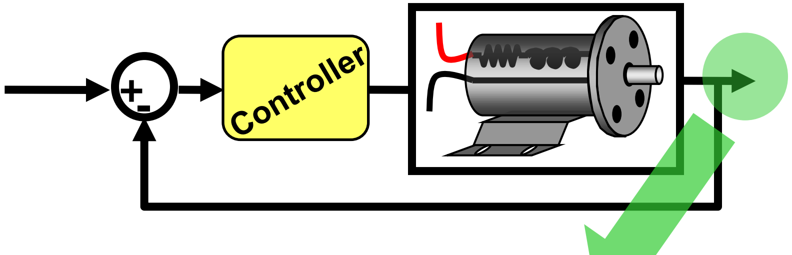


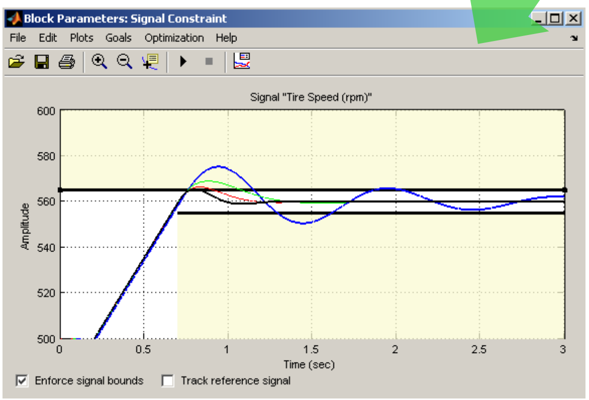
Possibilities for Compensator Design

- Linear Control Theory
  - Linearize system using Simulink Control Design
  - Perform linear control design with Control System Toolbox
  - Test controller in nonlinear system

- Specify System Response
  - Specify response characteristics
  - Automatic tuning using Simulink Design Optimization

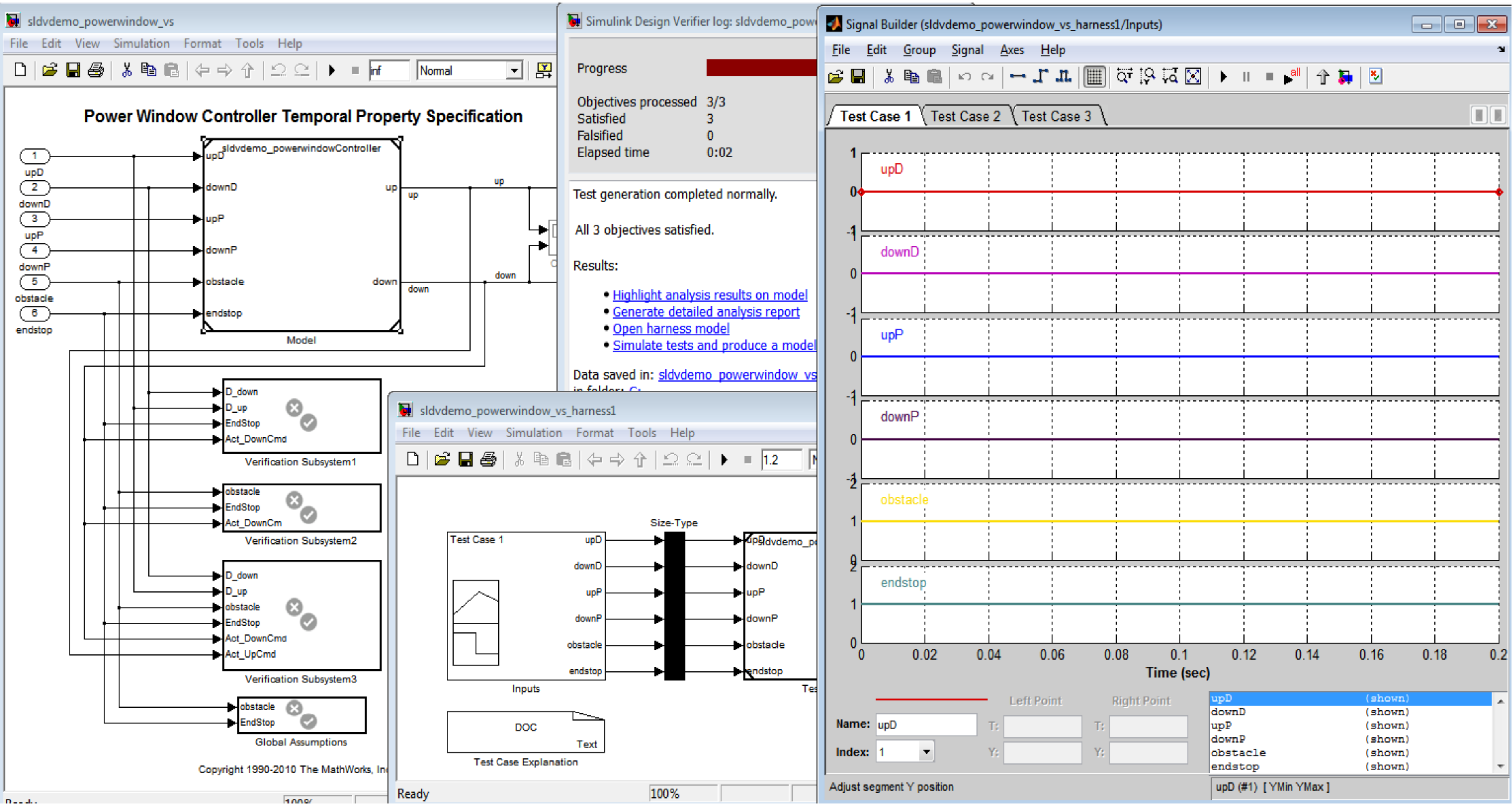






☑ Enforce signal bounds □ Track reference signal

Power Window: Generate Test Cases



Progress

Objectives processed 3/3
Satisfied 3
Falsified 0
Elapsed time 0:02

Test generation completed normally.

All 3 objectives satisfied.

Results:

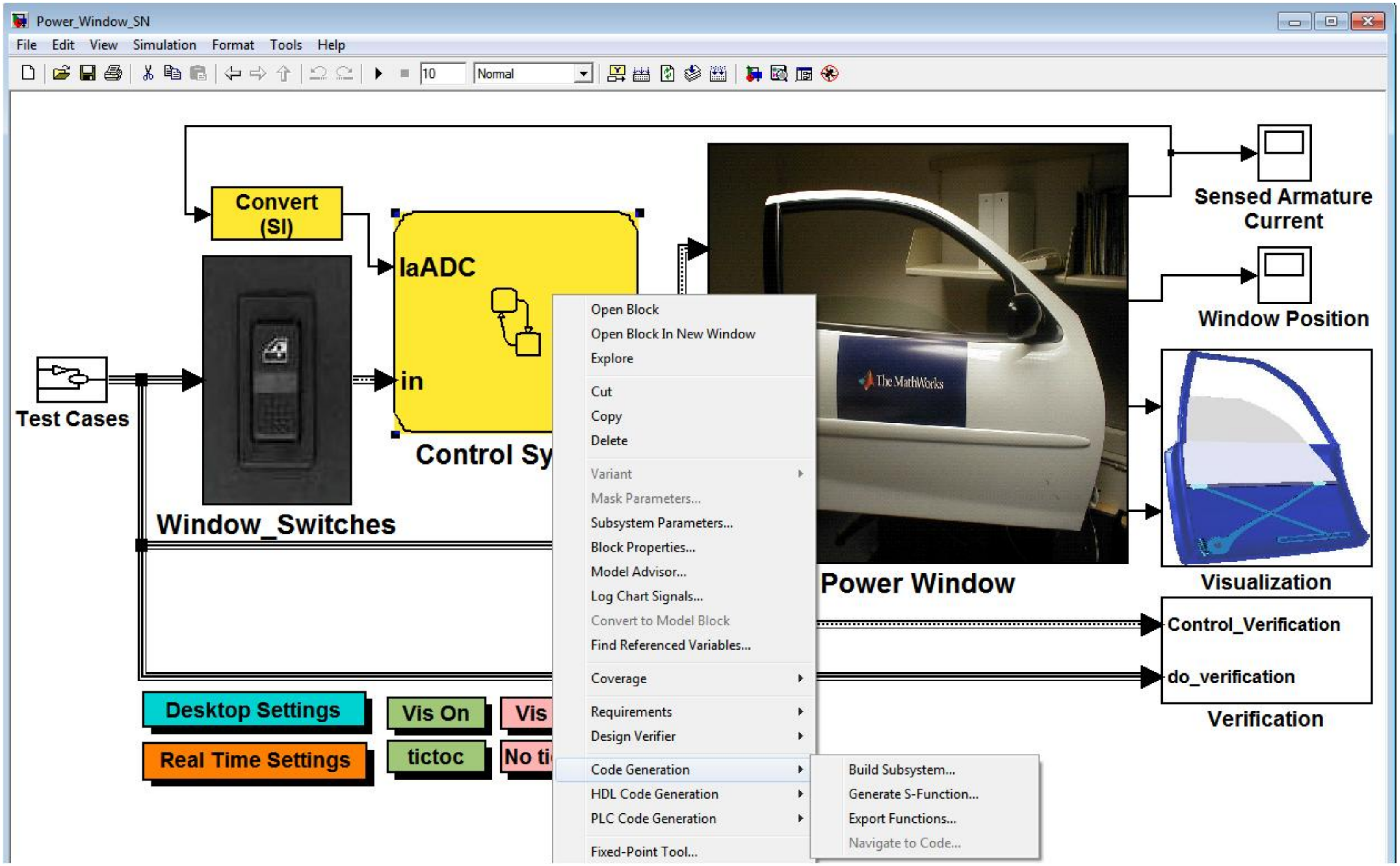
- Highlight analysis results on model
- Generate detailed analysis report
- Open harness model
- Simulate tests and produce a model

Data saved in: sldvdemo_powerwindow_vs

Name: upD

Index: 1

Power Window: Generate C/C++ Code

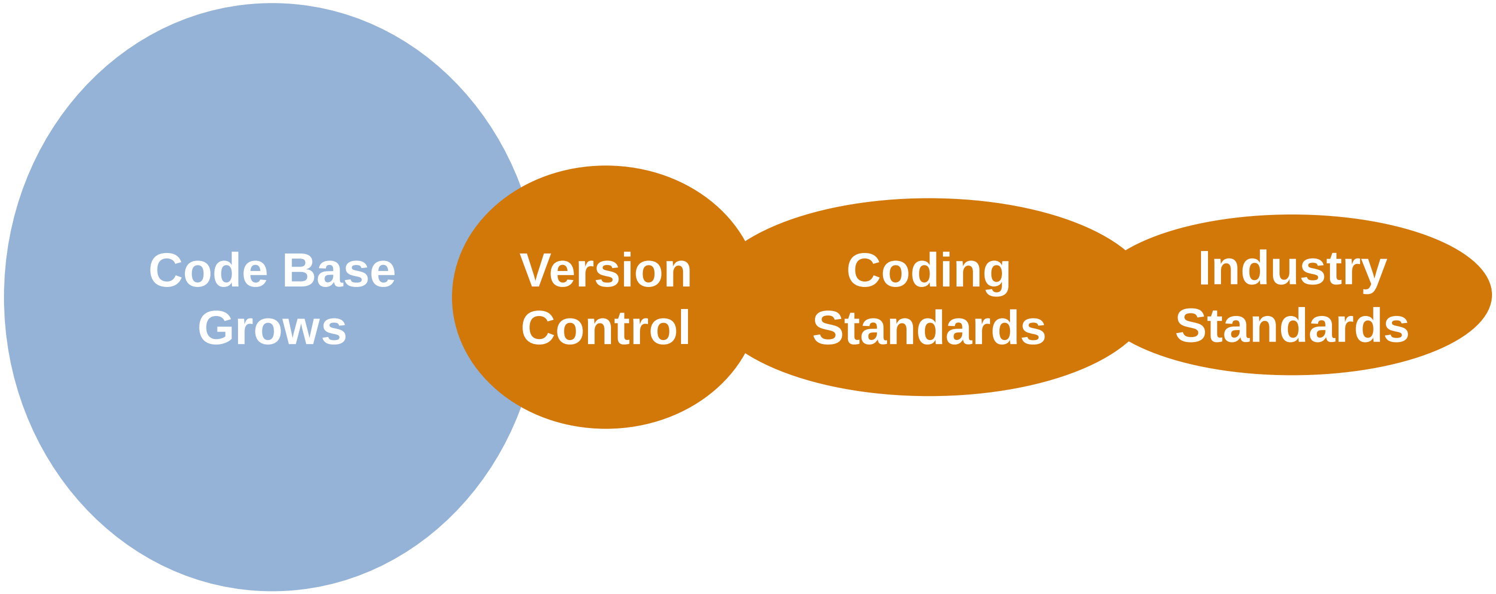


Power Window: Test System in Real Life

Putting it all together: Model-Based Design

History of Software Development

- Challenge appears… and so do proposed solutions.



History of Software Development - Challenges

Multiple domains come together

Industries grow

Time pressu

Auditing to protect consumer

Software teams grow

Finding the right people

Technology grows faster than population

Code base grows

Processing power increases

Quality is more important

Budget restrictions

New programming languages

New hardware platforms

Companies grow

Software Development - Concepts

Multiple domains come together

Industries grow

Time pressu

Auditing t protect consumer

Software teams grow

Finding the right people

Technolog grows fas than population

Code base grows

Processing power increases

Quality is more important

Budget restrictions

New programming languages

New hardware platforms

Companies grow

Coding standards

Reports

Industry standards

Task automation

Traceability

Translatio between languages

Verification and validation

Separate hardware from softwa

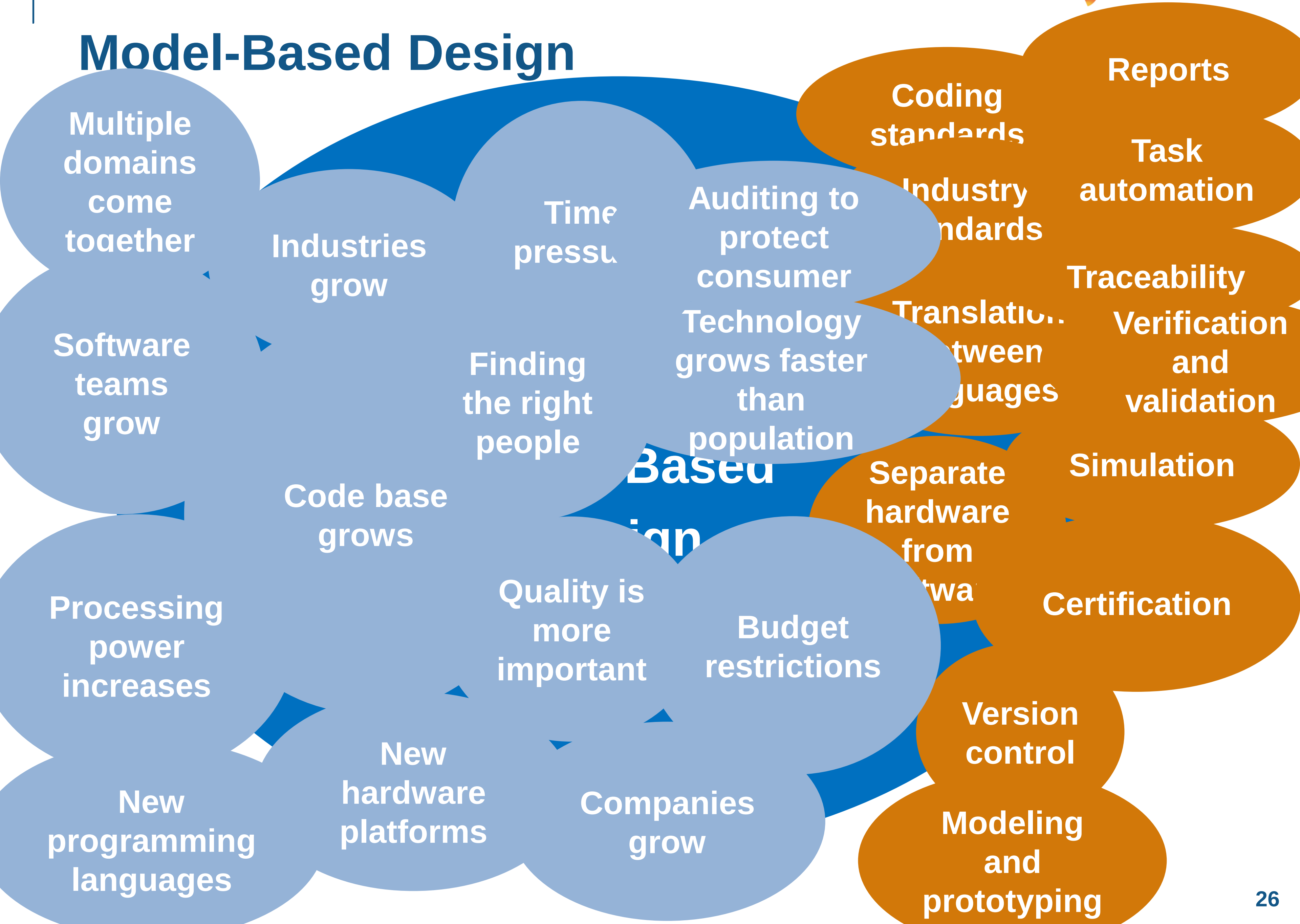
Simulation

Certification

Version control

Model-Based Design

Model-Based Design



Multiple domains come together

Industries grow

Time pressu

Auditing to protect consumer

Software teams grow

Finding the right people

Technology grows faster than population

Code base grows

Quality is more important

Budget restrictions

Processing power increases

New hardware platforms

Companies grow

New programming languages

Based

ign

Coding standards

Reports

Industry standards

Task automation

Traceability

Translation between languages

Verification and validation

Separate hardware from softwa

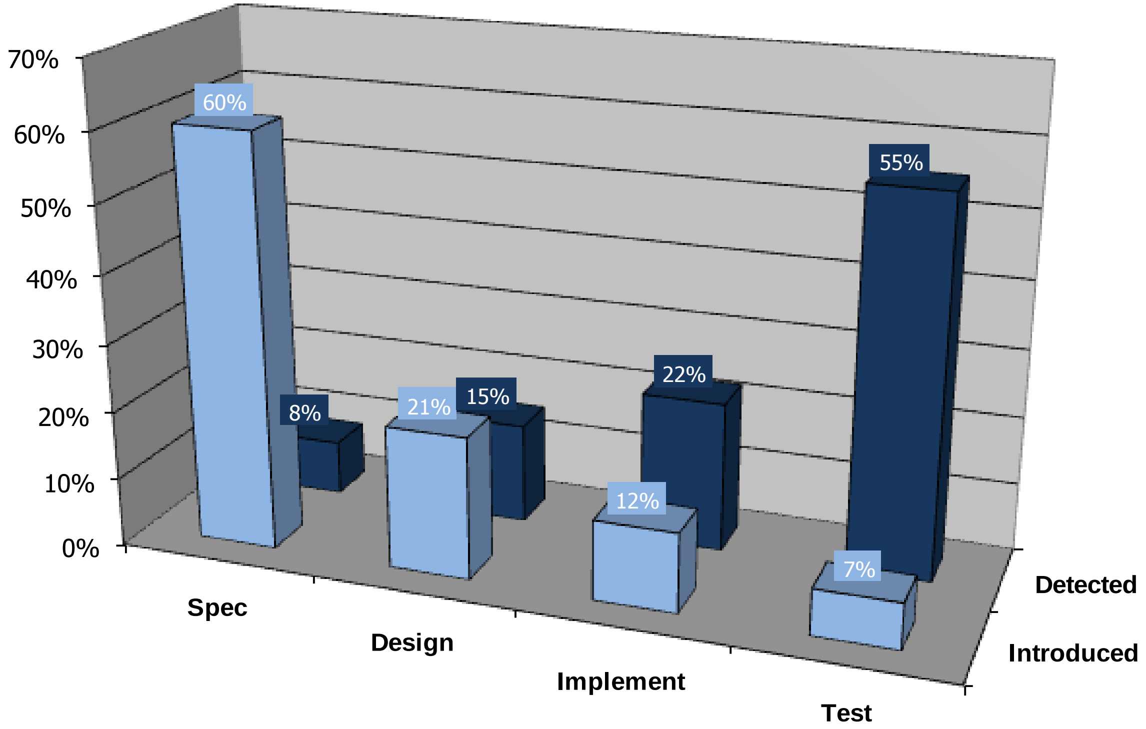
Simulation

Certification

Version control

Modeling and prototyping

Expensive to fix errors found late in the process

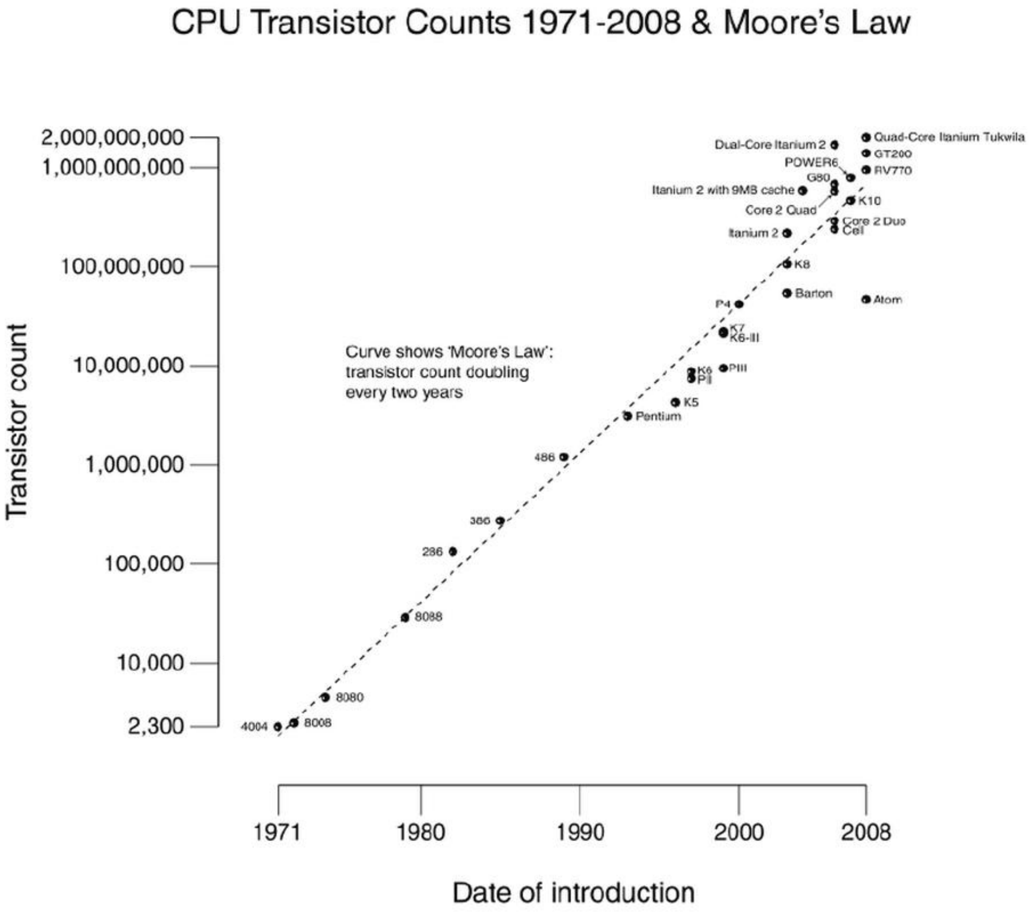


"…each delay in the detection and correction of a design problem makes it an order of magnitude more expensive to fix…"

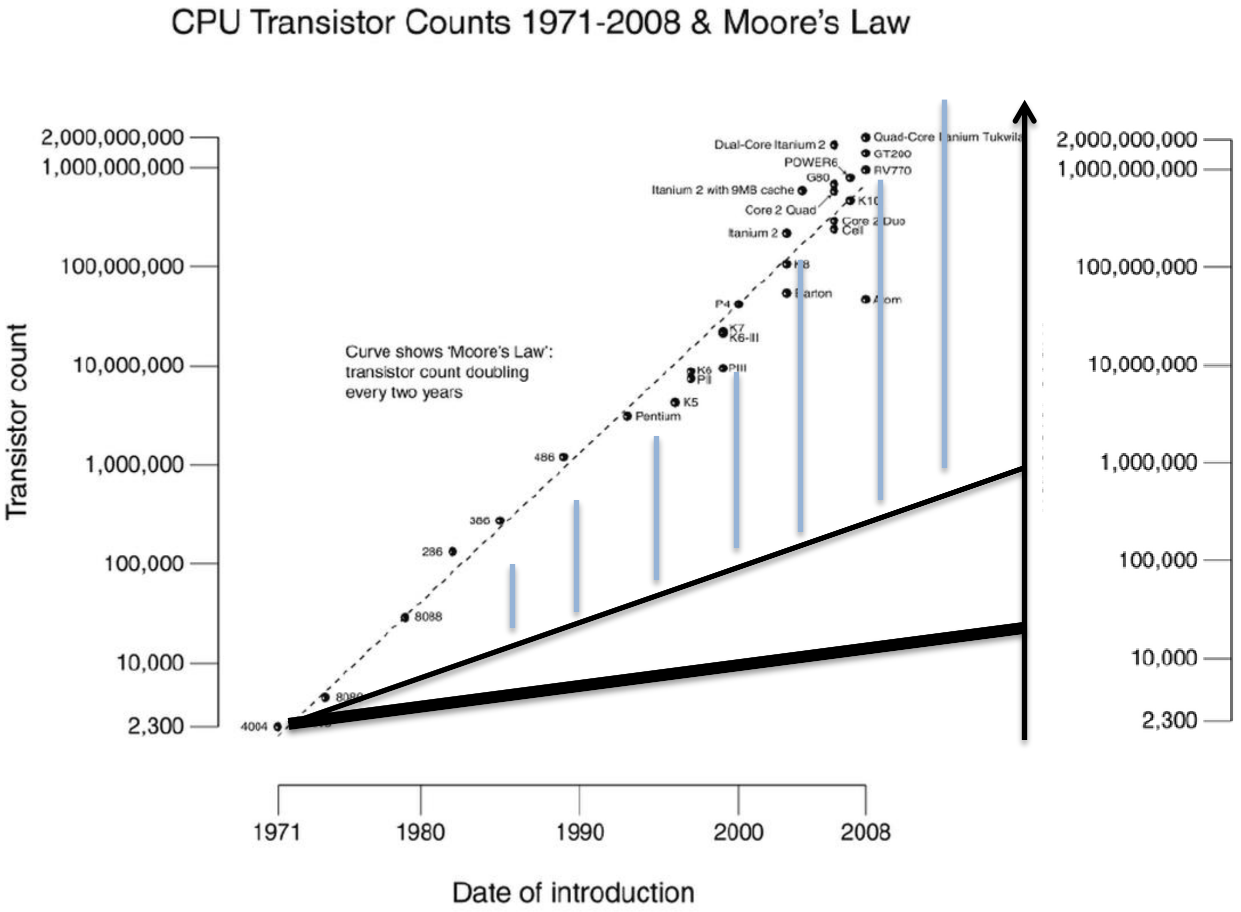
Clive Maxfield and Kuhoo Goyal
"EDA: Where Electronics Begins"
TechBites Interactive, October 1, 2001
ISBN: 0971406308]

Source: "Migration from Simulation to Verification with ModelSim®" by Paul Yanik. *EDA Tech Forum*, 2004 Mar 11, Newton MA

Moore’s Law

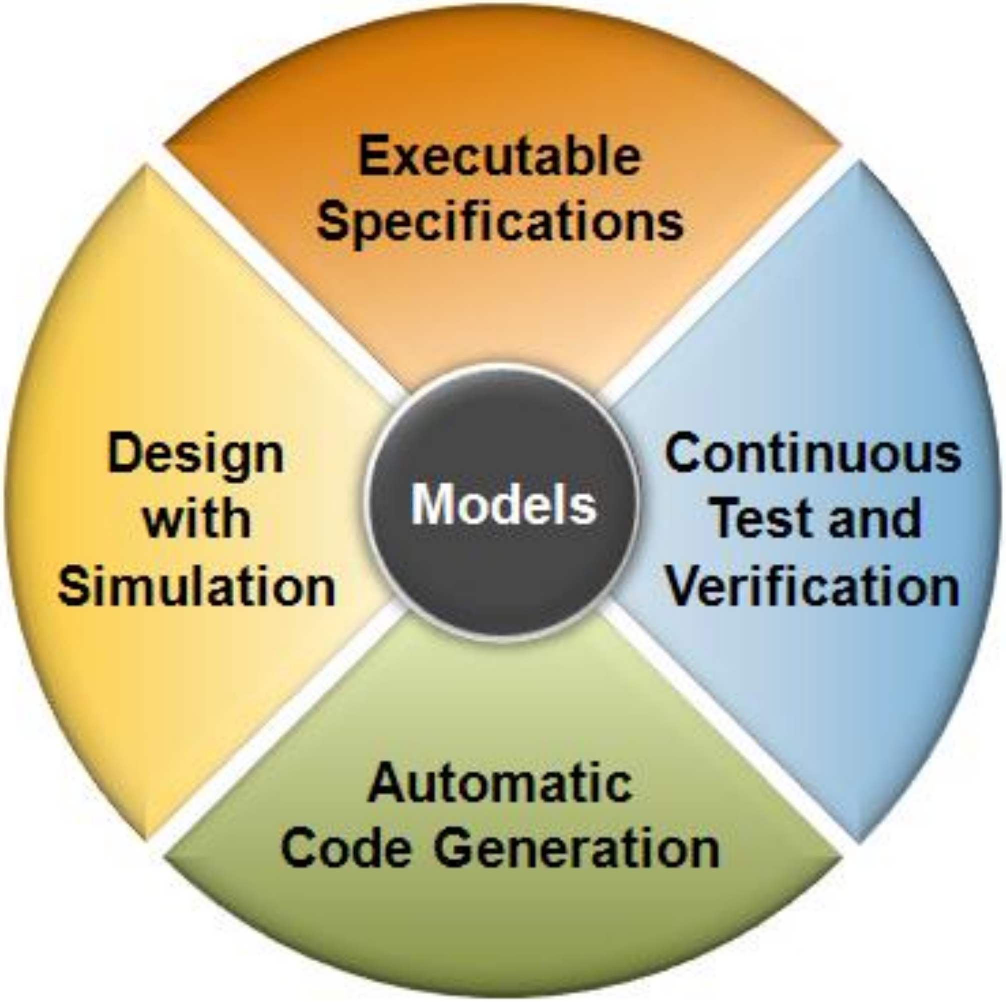

CPU Transistor Counts 1971-2008 & Moore’s Law
Transistor count
2,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
Date of introduction
1971
1980
1990
2000
2008
Curve shows ‘Moore’s Law’: transistor count doubling every two years
4004
8008
8080
8088
286
386
486
Pentium
K5
K6
PII
PIII
K7
K6-III
P4
Barton
K8
Itanium 2
Core 2 Duo
Cell
K10
Core 2 Quad
Itanium 2 with 9MB cache
G80
POWER6
Dual-Core Itanium 2
Quad-Core Itanium Tukwila
GT200
RV770
Atom

Moore's Law



Number of Employees: Increase 10% every year

Workflow for Model-Based Design


Executable Specifications
Design with Simulation
Models
Continuous Test and Verification
Automatic Code Generation

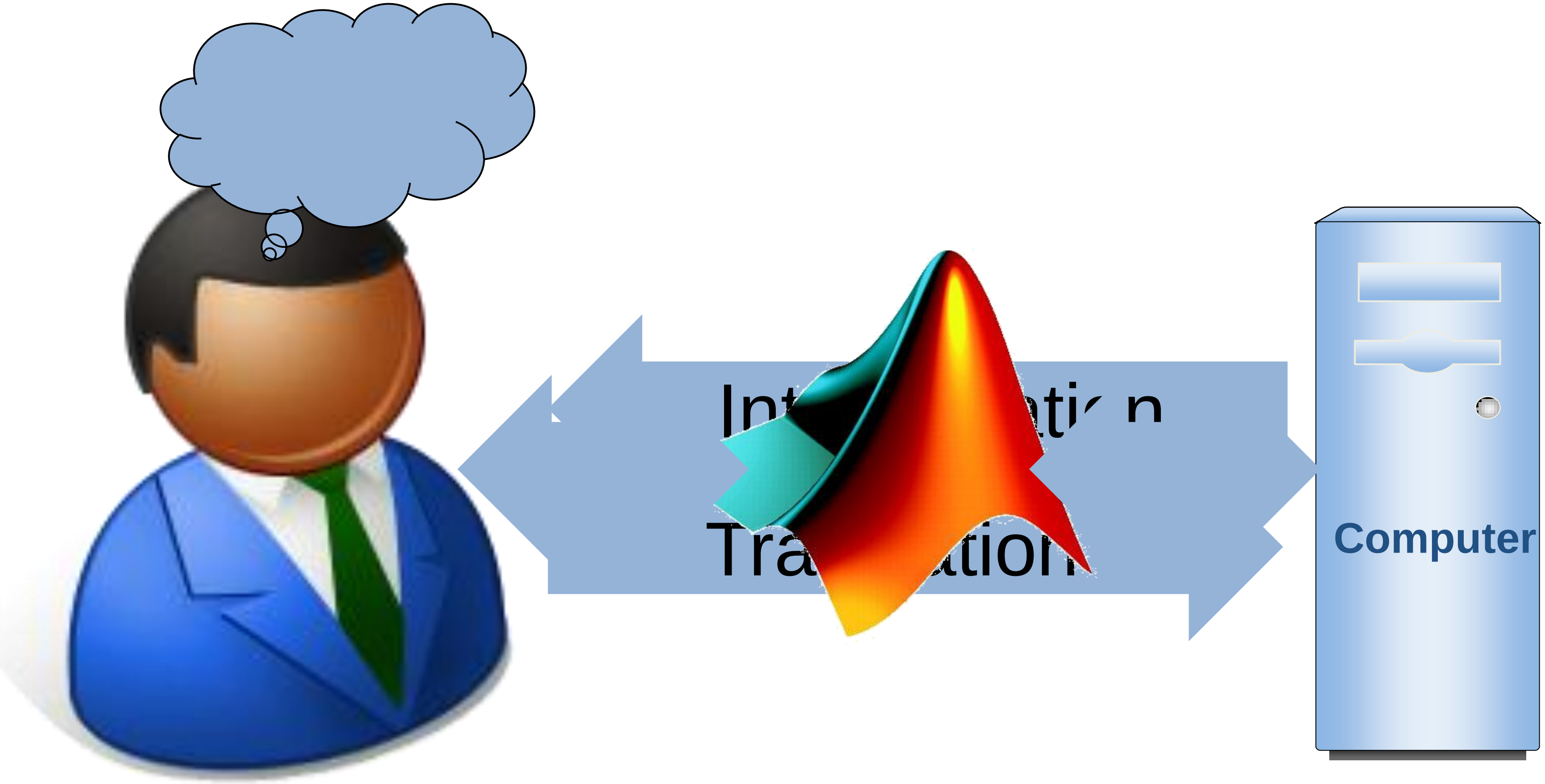
Summer of 95!

“Summer of ’95”

Steve Miller

MATLAB - Textual

Accelerating the Pace of Engineering and Science



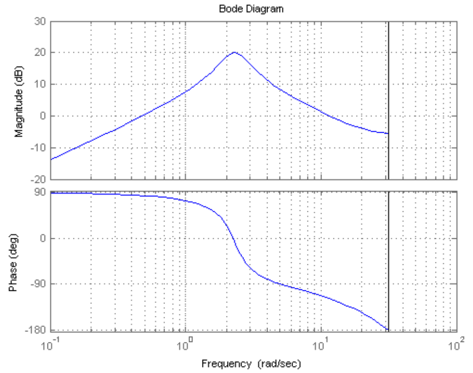
Reduce the effort to:

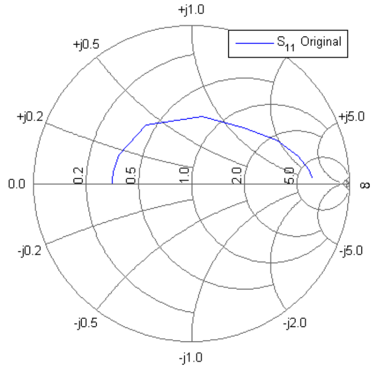
- Translate your thoughts for the computer
- Interpret the results from the computer

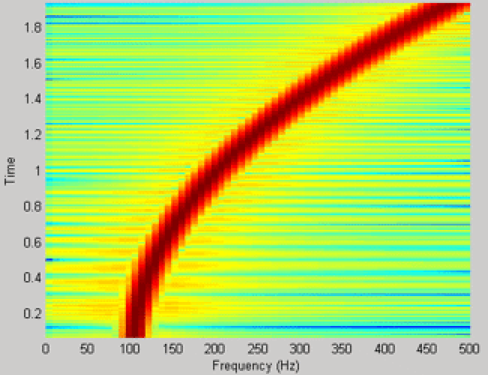
Accelerating the Pace of Engineering and Science

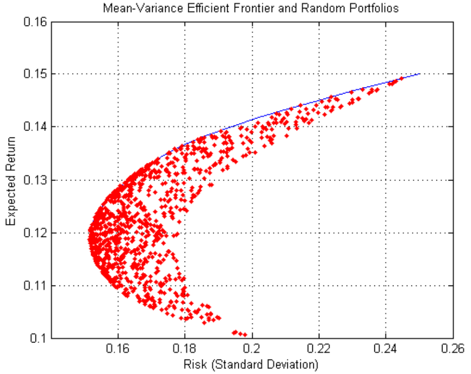
Similarities between Industries:

- Matrix / vector based mathematics
- Standard and specific operations
- Specific analysis charts









Accelerating the Pace of Engineering and Science

- Direct use of matrix equations

Accelerating the Pace of Engineering and Science

- Direct use of matrix equations

$$y = Ax + b$$

Accelerating the Pace of Engineering and Science

- Direct use of matrix equations

MATLAB Code

```
function y=MatrixEquation(A,x,b)
y=A*x+b;
```

Accelerating the Pace of Engineering and Science

- Direct use of matrix equations
- Interactive - immediate response

MATLAB Code

```
function y=MatrixEquation(A,x,b)
y=A*x+b;
```

C Code

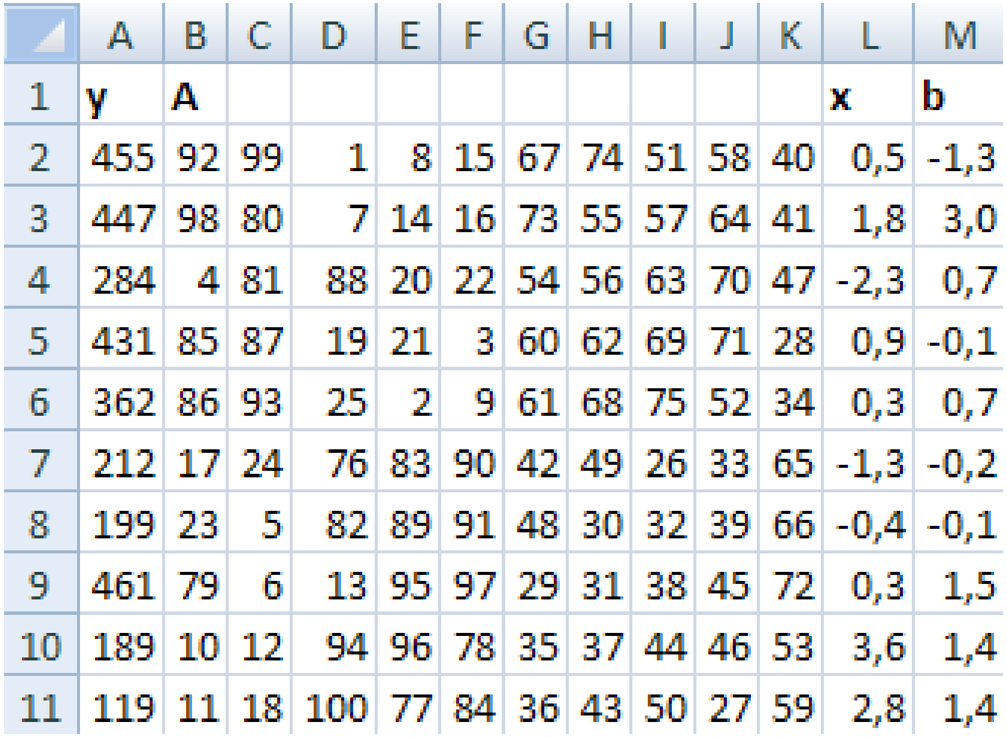
```
void MatrixEquation(float A[100], float x[10], float
    b[10], float y[10])
{
    int32 i0;
    float d0;
    int32 i1;
    for(i0 = 0; i0 < 10; i0++) {
        d0 = 0.0;
        for(i1 = 0; i1 < 10; i1++) {
            d0 += A[i0 + (i1 << 1)] * x[i1];
        }
        y[i0] = d0 + b[i0];
    }
}
```

Accelerating the Pace of Engineering and Science

- Direct use of matrix equations
- Interactive - immediate response
- Built-in engineering functions

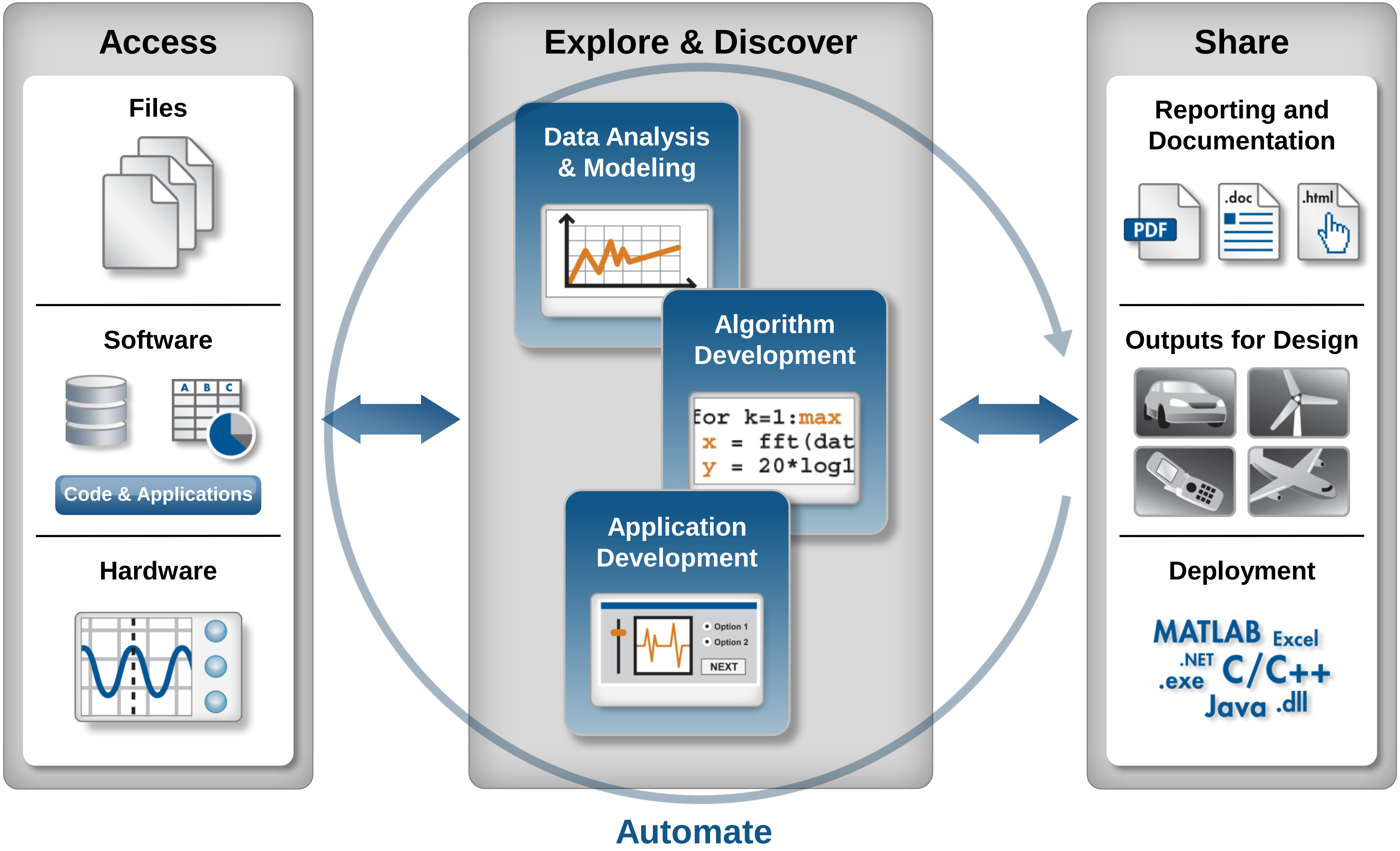
MATLAB Code

```
function y=MatrixEquation(A,x,b)
y=A*x+b;
```



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	y	A										x	b
2	455	92	99	1	8	15	67	74	51	58	40	0,5	-1,3
3	447	98	80	7	14	16	73	55	57	64	41	1,8	3,0
4	284	4	81	88	20	22	54	56	63	70	47	-2,3	0,7
5	431	85	87	19	21	3	60	62	69	71	28	0,9	-0,1
6	362	86	93	25	2	9	61	68	75	52	34	0,3	0,7
7	212	17	24	76	83	90	42	49	26	33	65	-1,3	-0,2
8	199	23	5	82	89	91	48	30	32	39	66	-0,4	-0,1
9	461	79	6	13	95	97	29	31	38	45	72	0,3	1,5
10	189	10	12	94	96	78	35	37	44	46	53	3,6	1,4
11	119	11	18	100	77	84	36	43	50	27	59	2,8	1,4

Data Analysis Tasks



Access

Files

Software

Code & Applications

Hardware

Explore & Discover

Data Analysis & Modeling

Algorithm Development

```
for k=1:max
 x = fft(dat
 y = 20*log1
```

Application Development

Share

Reporting and Documentation

Outputs for Design

Deployment

MATLAB Excel .NET .exe C/C++ Java .dll

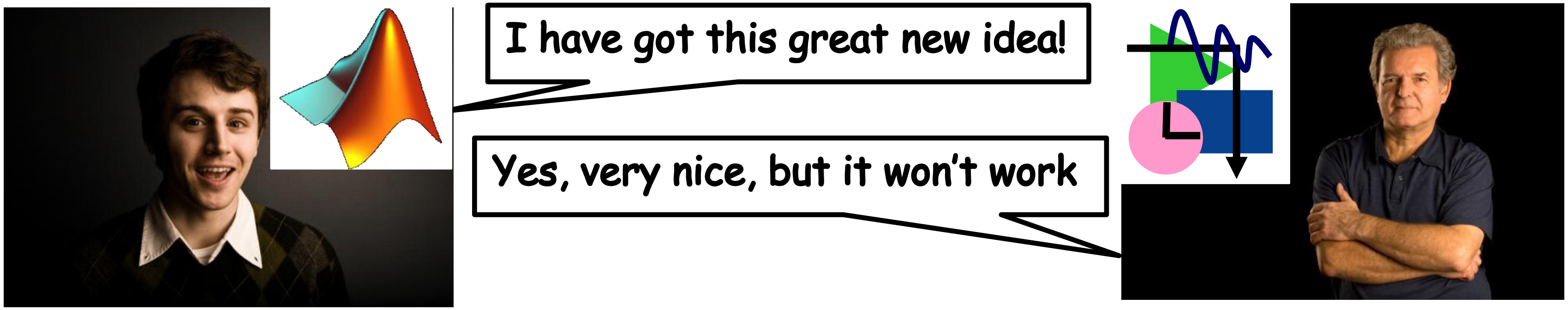
Automate

Simulink – Visual Block Diagram

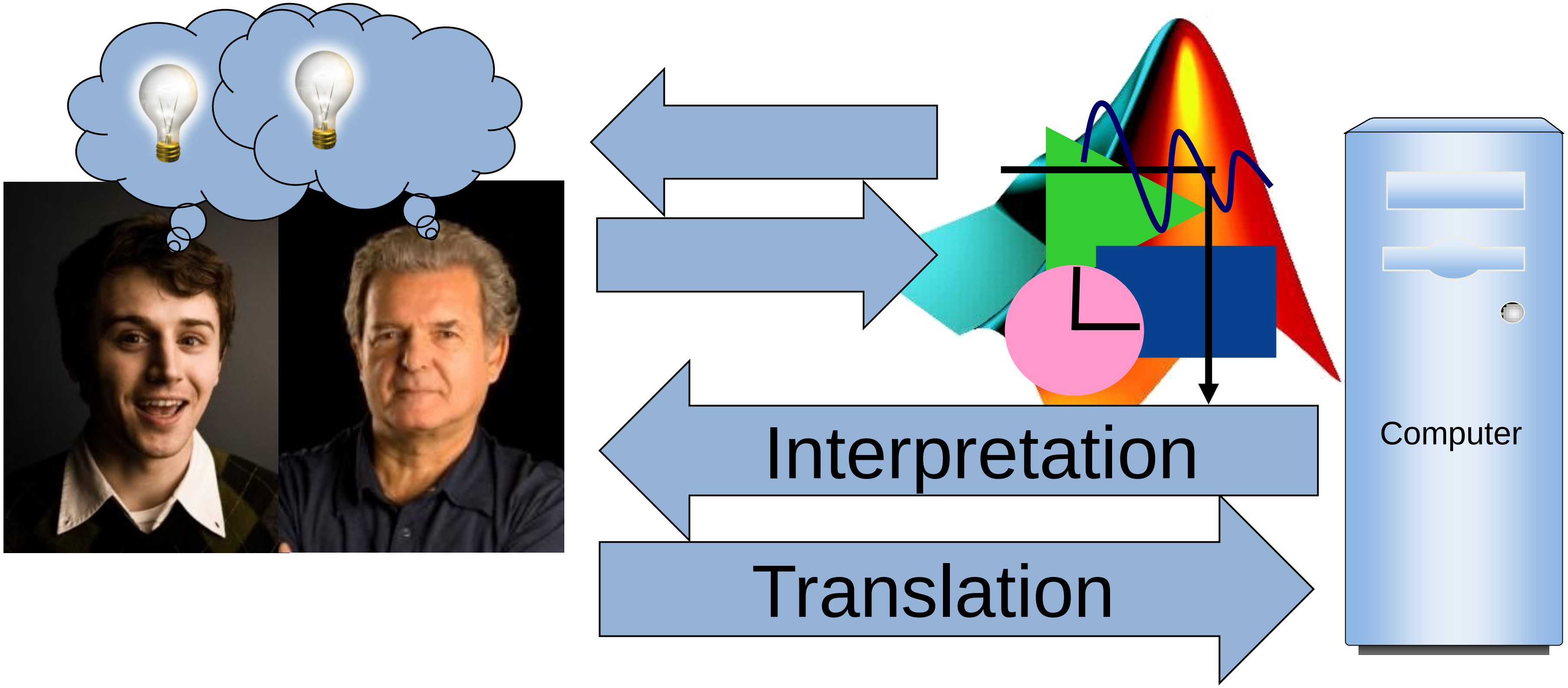
Accelerating the Pace of Engineering and Science

What is the value of their/your engineering department?

- Creativity/Innovation: bringing new ideas into practice
- Knowledge/Experience: knowing what will work and what will not


I have got this great new idea!
Yes, very nice, but it won't work

Accelerating the Pace of Engineering and Science with MATLAB & Simulink



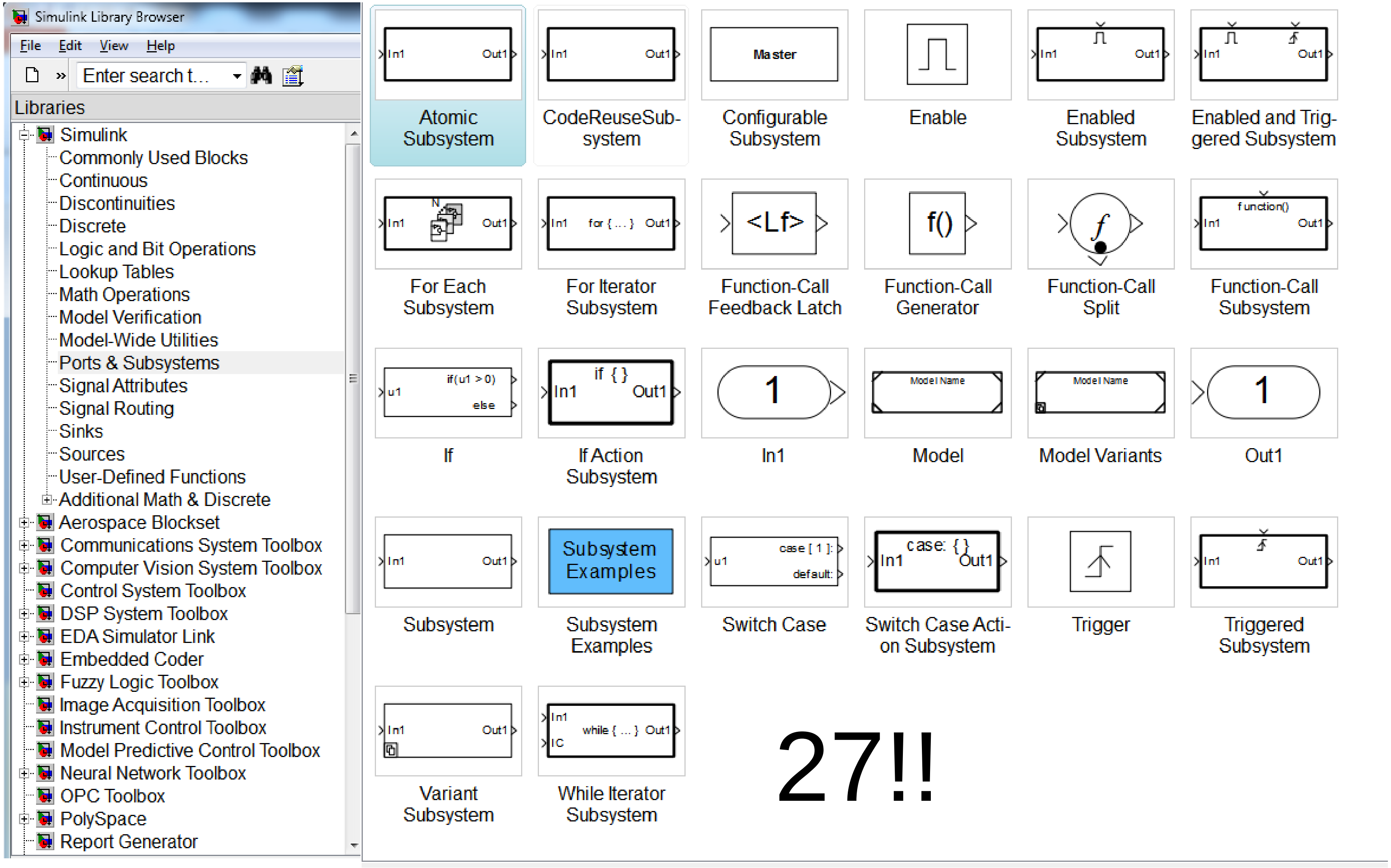
Interpretation

Translation

Computer

Reduce the effort to
- translate your thoughts for the computer
- interpret the results from the computer

How many Ports and Subsystem are there?



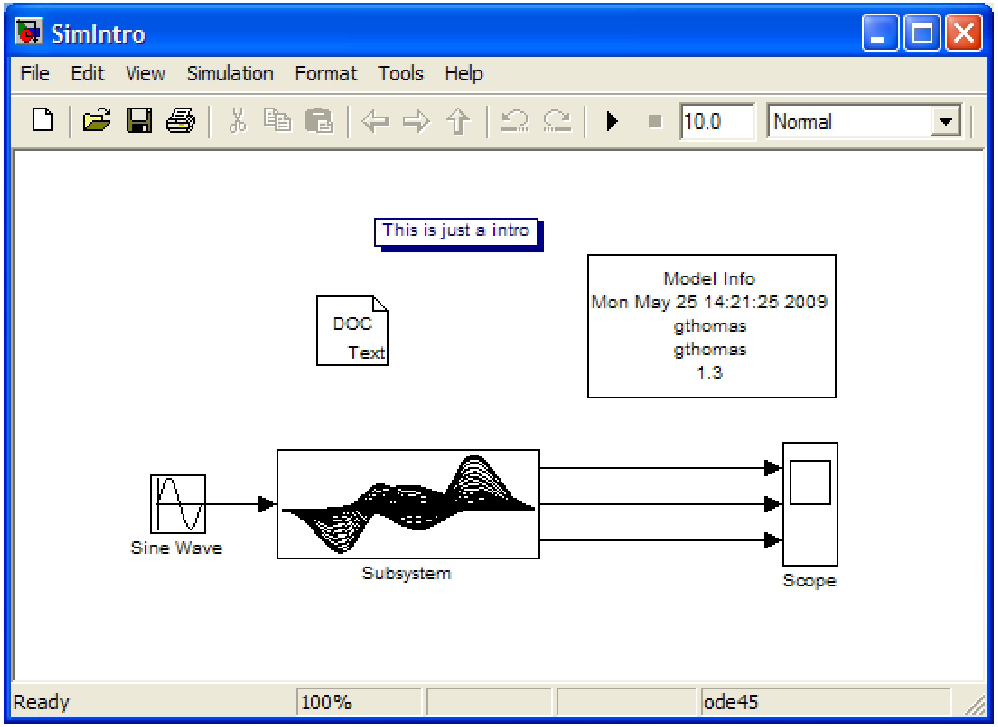
- Simulink
  - Commonly Used Blocks
  - Continuous
  - Discontinuities
  - Discrete
  - Logic and Bit Operations
  - Lookup Tables
  - Math Operations
  - Model Verification
  - Model-Wide Utilities
  - Ports & Subsystems
  - Signal Attributes
  - Signal Routing
  - Sinks
  - Sources
  - User-Defined Functions
  - Additional Math & Discrete
- Aerospace Blockset
- Communications System Toolbox
- Computer Vision System Toolbox
- Control System Toolbox
- DSP System Toolbox
- EDA Simulator Link
- Embedded Coder
- Fuzzy Logic Toolbox
- Image Acquisition Toolbox
- Instrument Control Toolbox
- Model Predictive Control Toolbox
- Neural Network Toolbox
- OPC Toolbox
- PolySpace
- Report Generator

Atomic Subsystem	CodeReuseSubsystem	Configurable Subsystem	Enable	Enabled Subsystem	Enabled and Triggered Subsystem
For Each Subsystem	For Iterator Subsystem	Function-Call Feedback Latch	Function-Call Generator	Function-Call Split	Function-Call Subsystem
If	If Action Subsystem	In1	Model	Model Variants	Out1
Subsystem	Subsystem Examples	Switch Case	Switch Case Action Subsystem	Trigger	Triggered Subsystem
Variant Subsystem	While Iterator Subsystem				

27!!

What is Simulink?

- **Explains Reality (Communication)**
  A description using basic principles that has some predictive value about behavior
- **Specifies Reality (Design)**
  A description using basic principles that specifies desired behavior
- **Replaces Reality (Simulation)**
  A description that has some predictive value about the behavior of the real thing



DEMO

What is new in Simulink?

- Simulink Projects
- Data Inspector
- Comparing XML
- Concurrent execution
- Modeling Task
- Model Variants
- Subsystem Variants
- Model Explorer Improved
- Logging to Datasets
- Comparing Files/Folders
- Parallel Builds
- Model Advisor
- Export to Web

What is new in Simulink Blocksets?

- Formal Methods
  - Design Errors
  - Property Proving
  - Test Generation
- Code Generation
- Physical Modeling
- Fixed-Point word scaling
- SimEvents
- Simulink Code Inspector
- xPC Target
- SIL Performance

- Linking requirements to word via external file
- Export to web

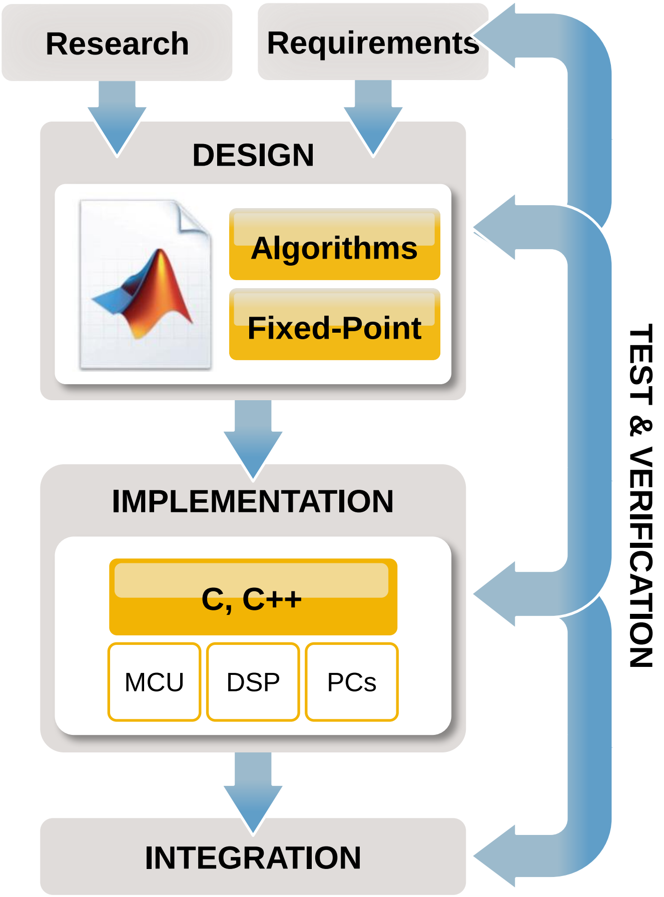
READ
Release
Notes!!!

MAB

- Open Video

A better workflow to implementation

- One language
  - No multiple copies of source code
  - Integrate real-world design constraints in MATLAB
- One integrated design environment
  - Integrated visualization, analysis & debugging
- Automatic code generation
  - Path to embedded software (Embedded C)
  - Path to FPGA/ASIC (HDL)



Introduction to Simscape:

Mechanic

Electric

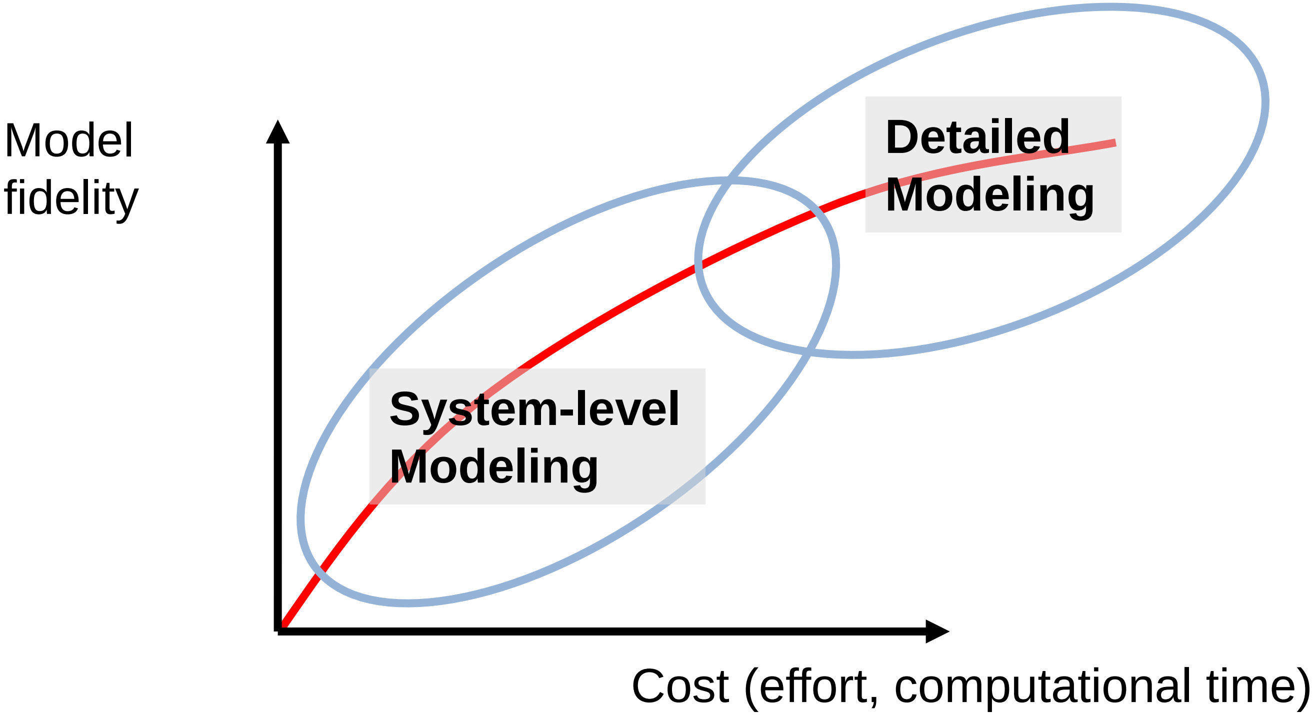
Hydraulic

Magnetic

Thermal

Pneumatic

System-level Modeling

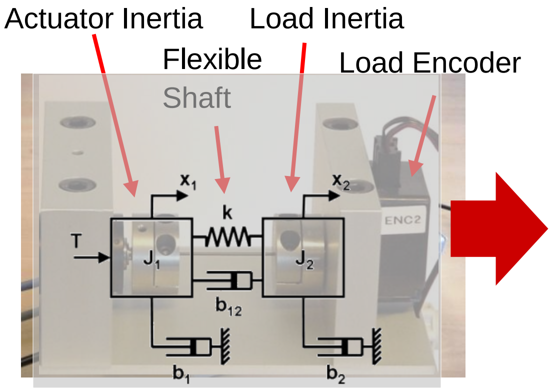


- Model the dynamics that matter for your analysis
- Balance cost and model fidelity

Modeling Dynamic Systems:
two approaches

First-Principles Modeling

Use an understanding of the system's physics to derive a mathematical representation

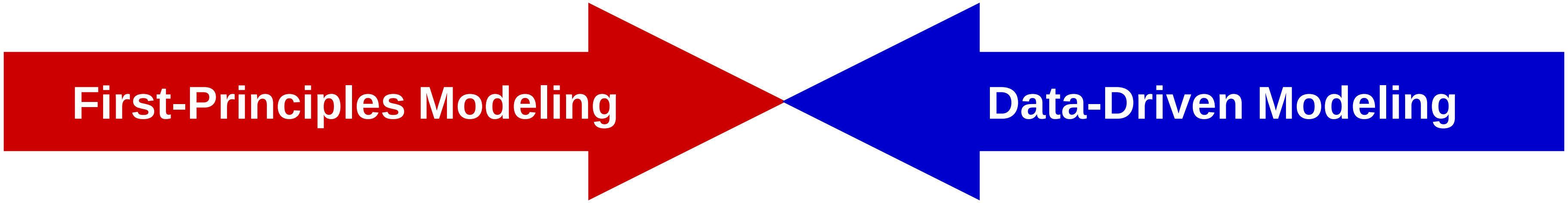


Sir Isaac Newton

$$J_1\ddot{x}_1 = \sum Torques = -b_1\dot{x}_1 - k(x_1-x_2) - b_{12}(\dot{x}_1-\dot{x}_2) + T$$

$$J_2\ddot{x}_2 = \sum Torques = -b_2\dot{x}_2 + k(x_1-x_2) + b_{12}(\dot{x}_1-\dot{x}_2)$$

Modeling Dynamic Systems:
two approaches



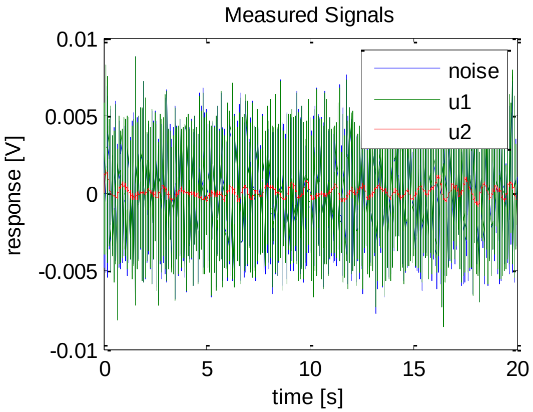
First-Principles Modeling

Data-Driven Modeling

Use an understanding of the system's physics to derive a mathematical representation

Use system test data to derive a mathematical representation

$$H(z) = \begin{bmatrix} \dfrac{1.22z^3 - 0.62z^2 - 0.78z - 1.27}{z^4 - 3.55z^3 + 5.08z^2 - 3.52z + 0.98} \\[2ex] \dfrac{0.85z^3 - 1.14z^2 + 2.37z - 0.52}{z^4 - 3.55z^3 + 5.08z^2 - 3.52z + 0.98} \end{bmatrix}$$



Both have Advantages & Disadvantages

Complete Modeling Environment

Advantages:

- Insight in behavior
- Physical parameters

Disadvantages:

- Friction and turbulence?
- Time consuming
- Requires expertise

Advantages:

- Fast
- Accurate

Disadvantages:

- Requires plant
- Requires data acquisition system

Tools that span both modeling approaches

Enhance Advantages, Reduce Disadvantages

Complete Modeling Environment

First-Principles

Simulink
Stateflow
Simscape

SimMechanics
SimDriveline
SimHydraulics
SimElectronics
SimPowersystems

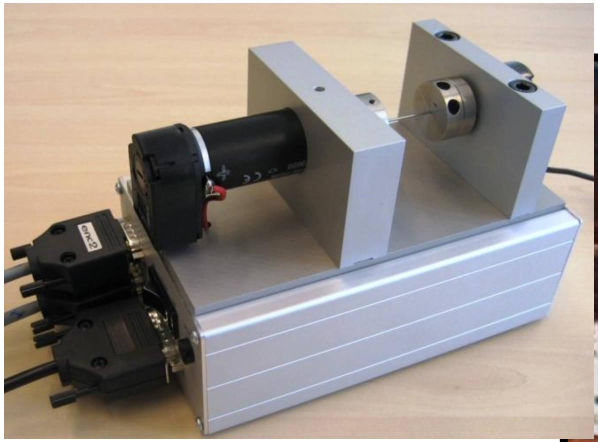
Simulink
Design
Optimization

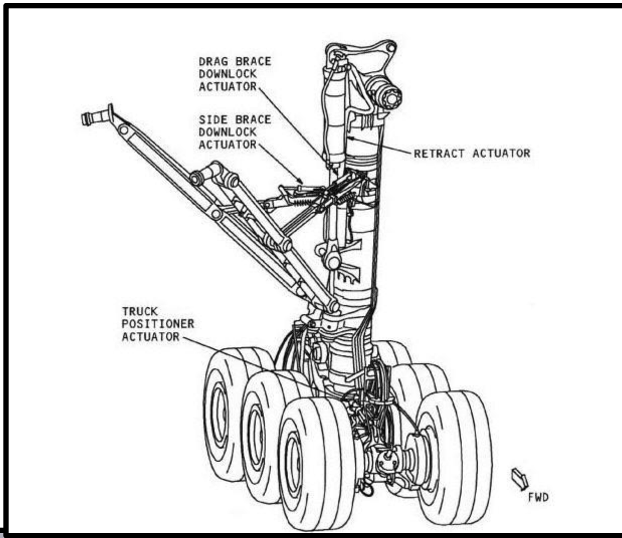
Data-Driven

System
Identification
Toolbox

Test &
Measurement
Tools

SimMechanics: accurate modeling of 3D mechanical systems



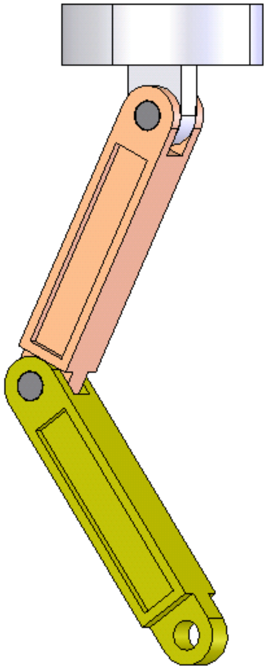

DRAG BRACE DOWNLOCK ACTUATOR
SIDE BRACE DOWNLOCK ACTUATOR
RETRACT ACTUATOR
TRUCK POSITIONER ACTUATOR
FWD

✓ ***3D Multi-Body Dynamics***

✓ ***Bodies and Joints***

✓ ***CAD Translation***

‘Simple’ example: double pendulum



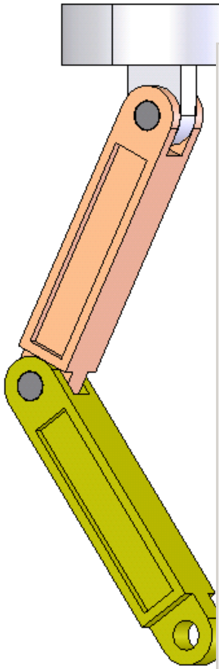
Traditional approach:

Derivation of the equations of motion requires knowledge and effort.

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

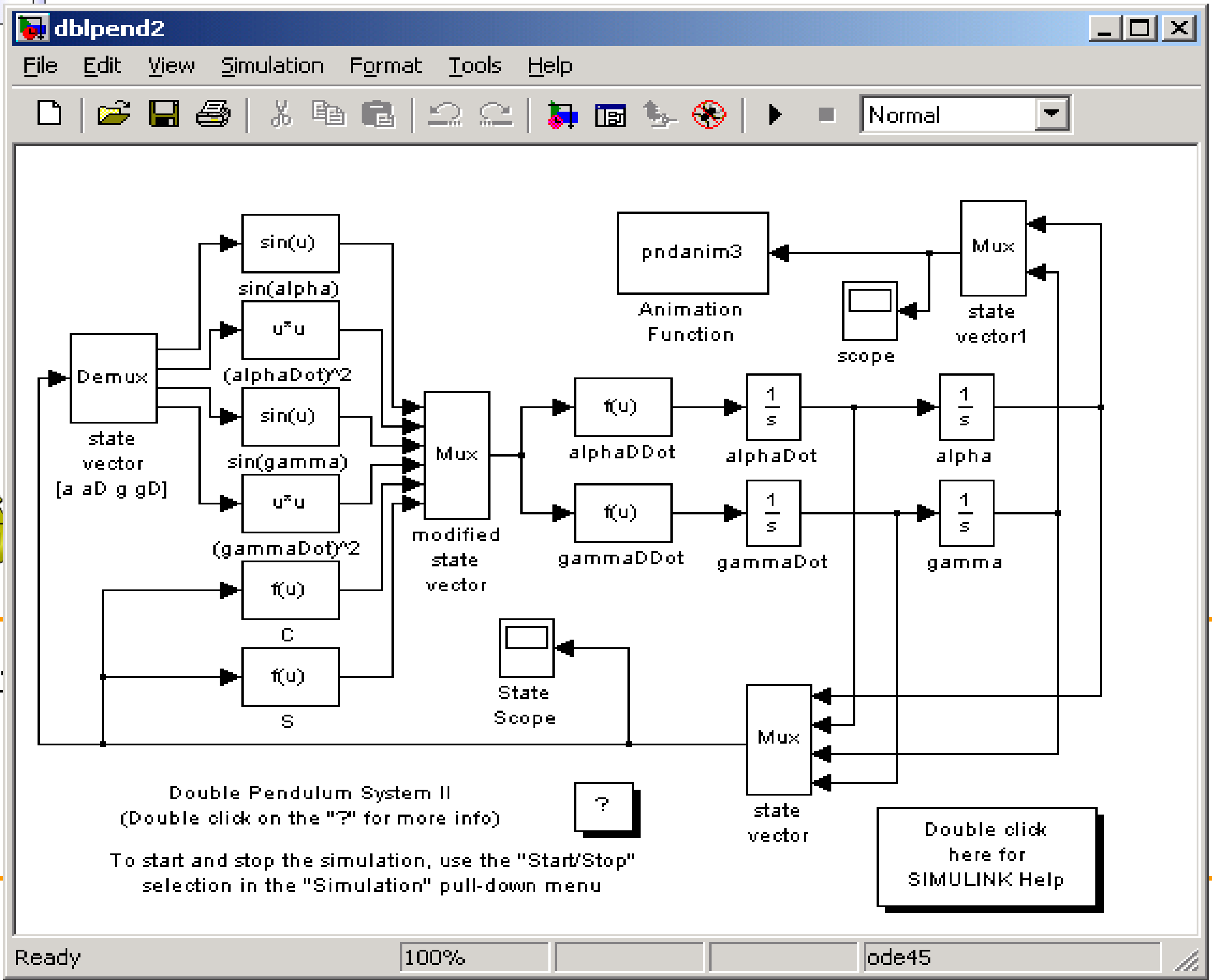
$$\theta_2'' = \frac{2\,\sin(\theta_1 - \theta_2)\,(\theta_1'^2\,L_1\,(m_1 + m_2) + g(m_1 + m_2)\cos\theta_1 + \theta_2'^2\,L_2\,m_2\cos(\theta_1 - \theta_2))}{L_2\,(2\,m_1 + m_2 - m_2\cos(2\,\theta_1 - 2\,\theta_2))}$$

‘Simple’ example: double pendulum



$$\theta_1'' = \frac{-g\,(2}{}$$

$$\theta_2'' =$$

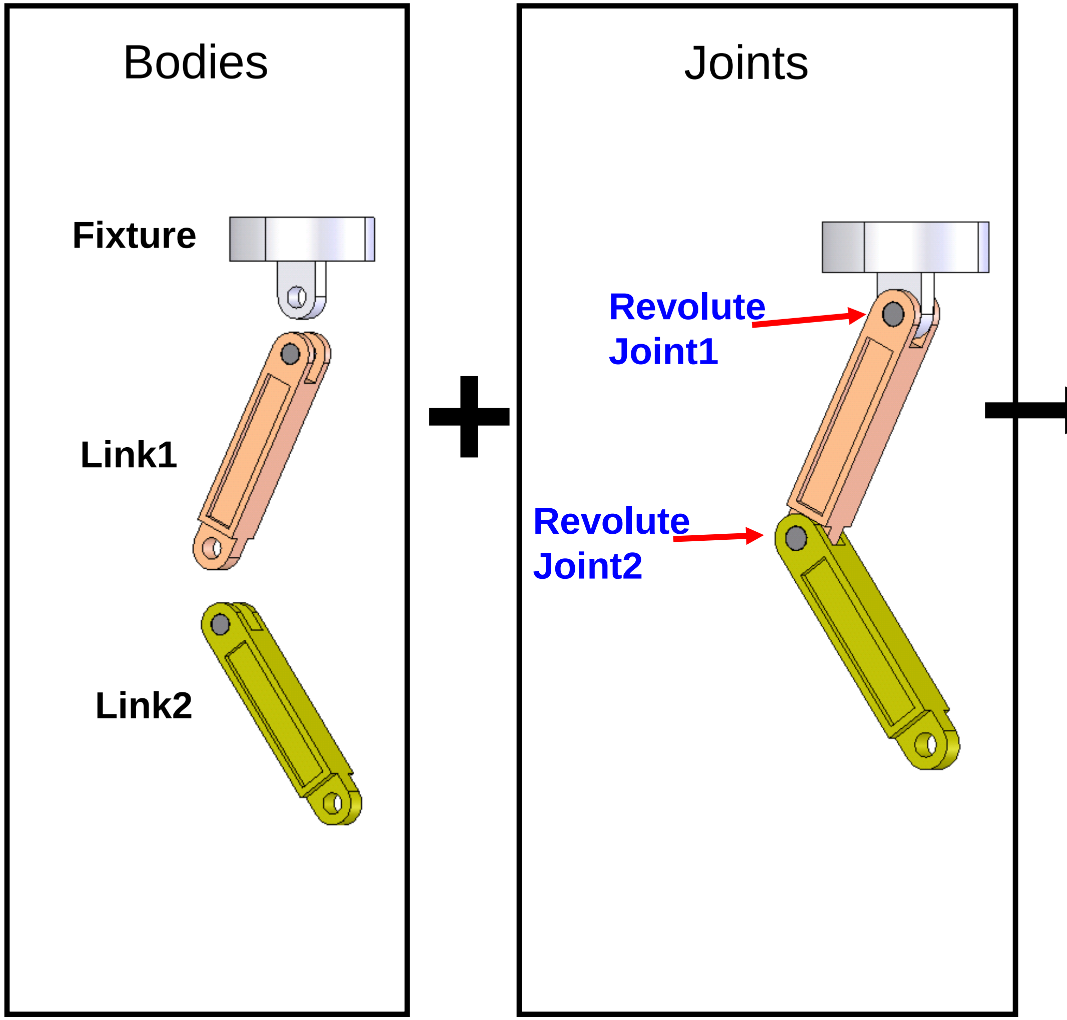


Double Pendulum System II
(Double click on the "?" for more info)

To start and stop the simulation, use the "Start/Stop" selection in the "Simulation" pull-down menu

With SimMechanics

DEMO



Bodies

Fixture

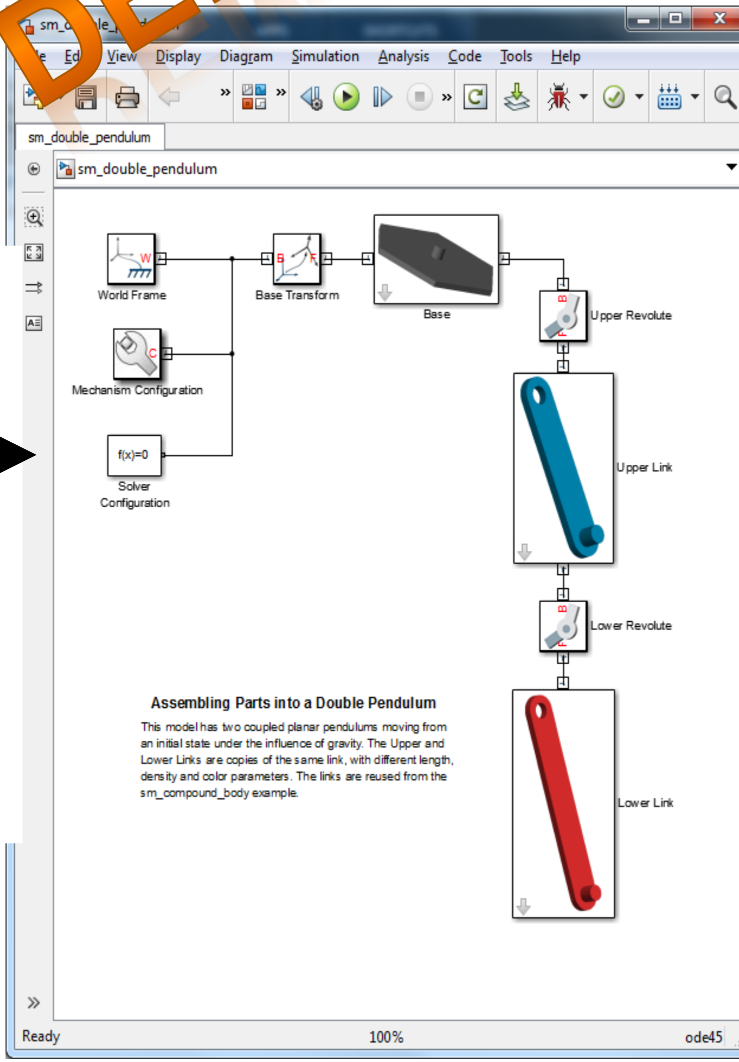
Link1

Link2

Joints

Revolute Joint1

Revolute Joint2



Assembling Parts into a Double Pendulum

This model has two coupled planar pendulums moving from an initial state under the influence of gravity. The Upper and Lower Links are copies of the same link, with different length, density and color parameters. The links are reused from the sm_compound_body example.

Introduction to Stateflow – Flow diagrams and State Machines

When should I use MATLAB?

- Next step for traditional programming (4th Generation)
- Quick and powerful (dedicated) visualization
- Simple C code Generation is possible
- Deployment
- Task Automation
- Data Analysis

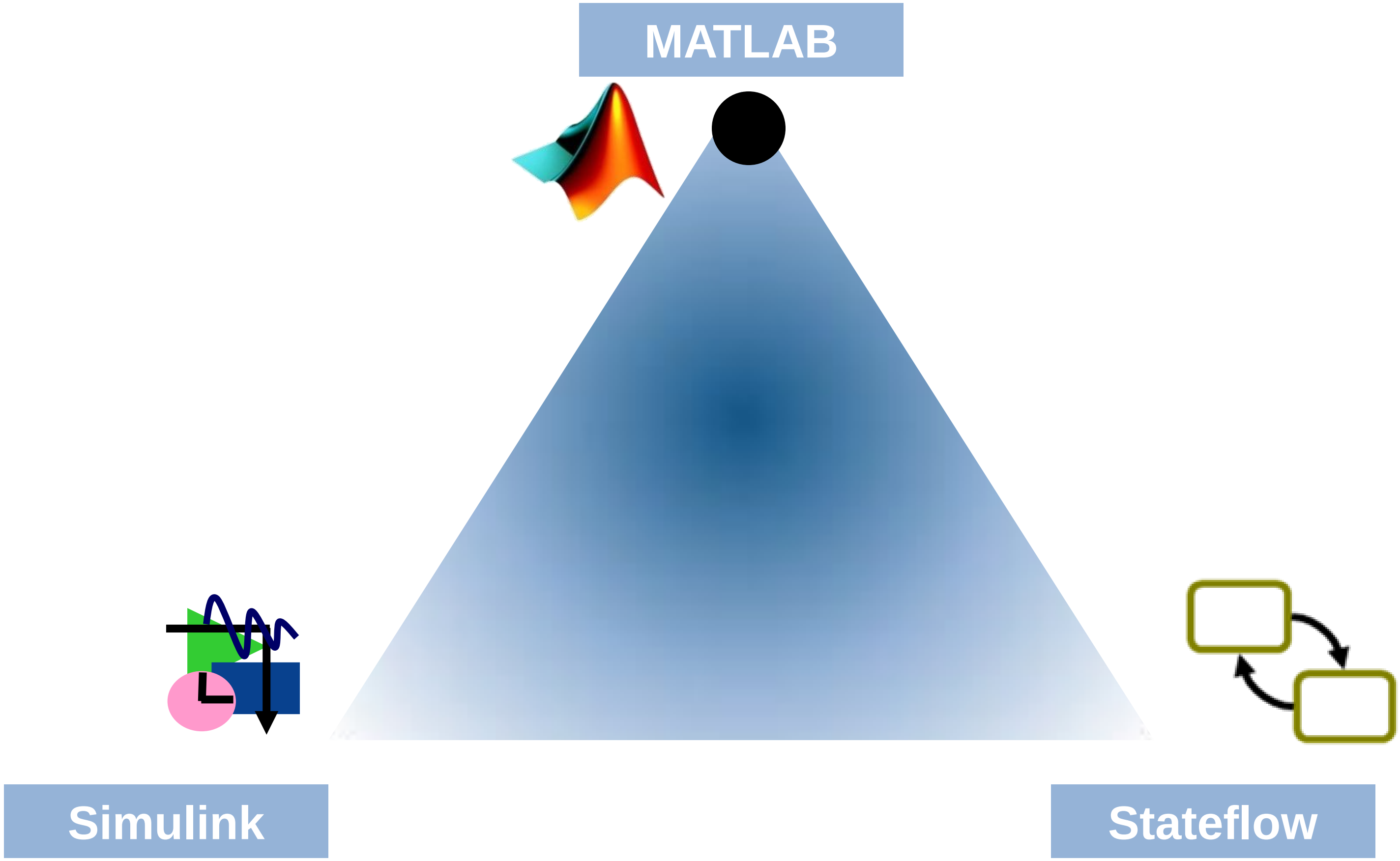
When should I use Simulink?

- System level overview
  - Signal flow/Block diagram representation
  - Architecture/Hierarchy definition
- Multi Rate/Multi Domain System
  - Mixed Signals
  - Physical Models
- Advanced Code Generation
- Certification – Model Based Design Support
  - Model Coverage, requirement traceability, formal proving, modelling standards....

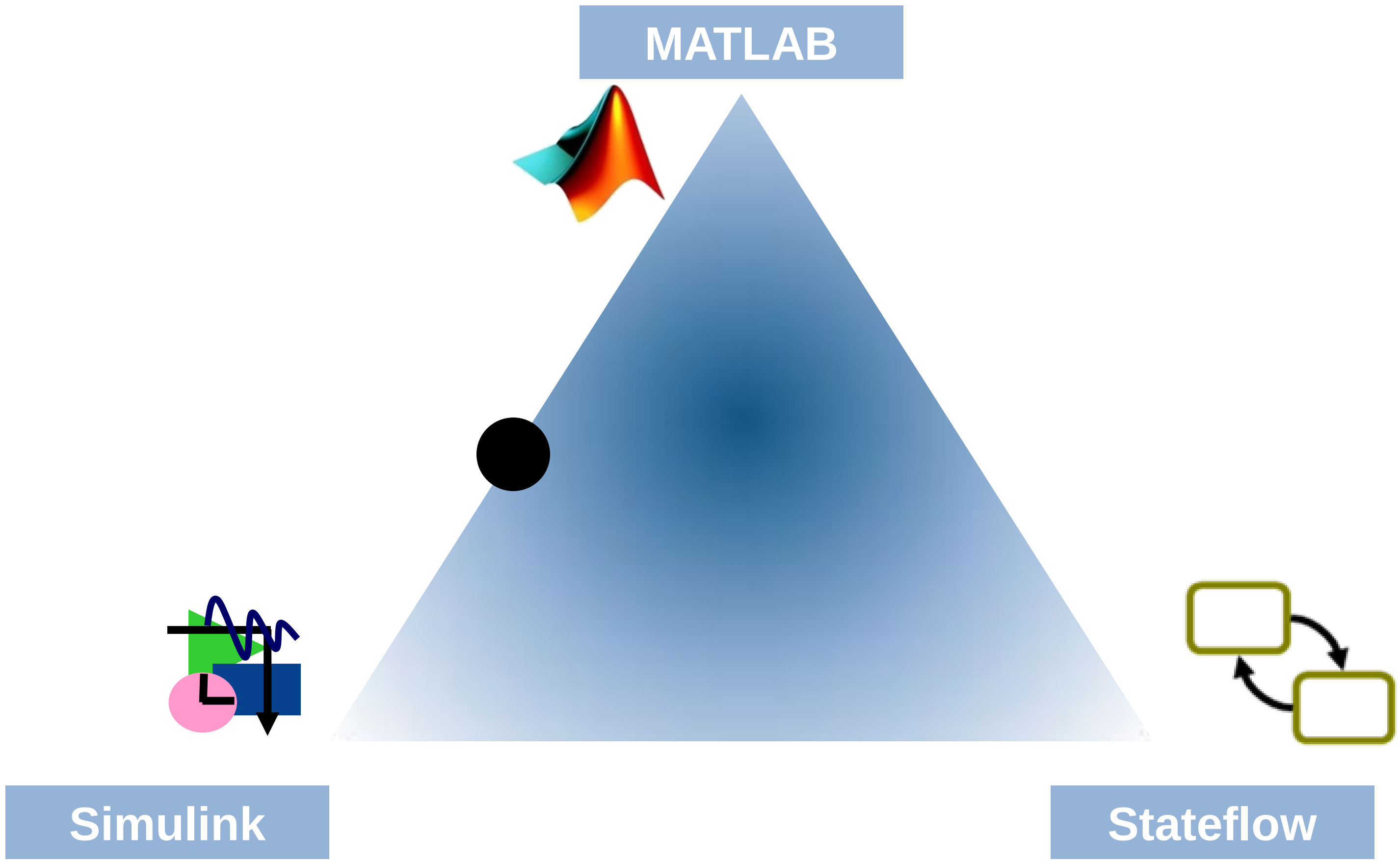
When should I use Stateflow?

- Control Logic
  - State Machine
  - Discrete Events
- Scheduling
  - Drive Simulink
  - Control flow Programming
- Mode Switching
- Fault Management

User Case – Simple filter, Image algorithm, GUI, y = A*x + B, visualization



User Case – Fixed Point Development

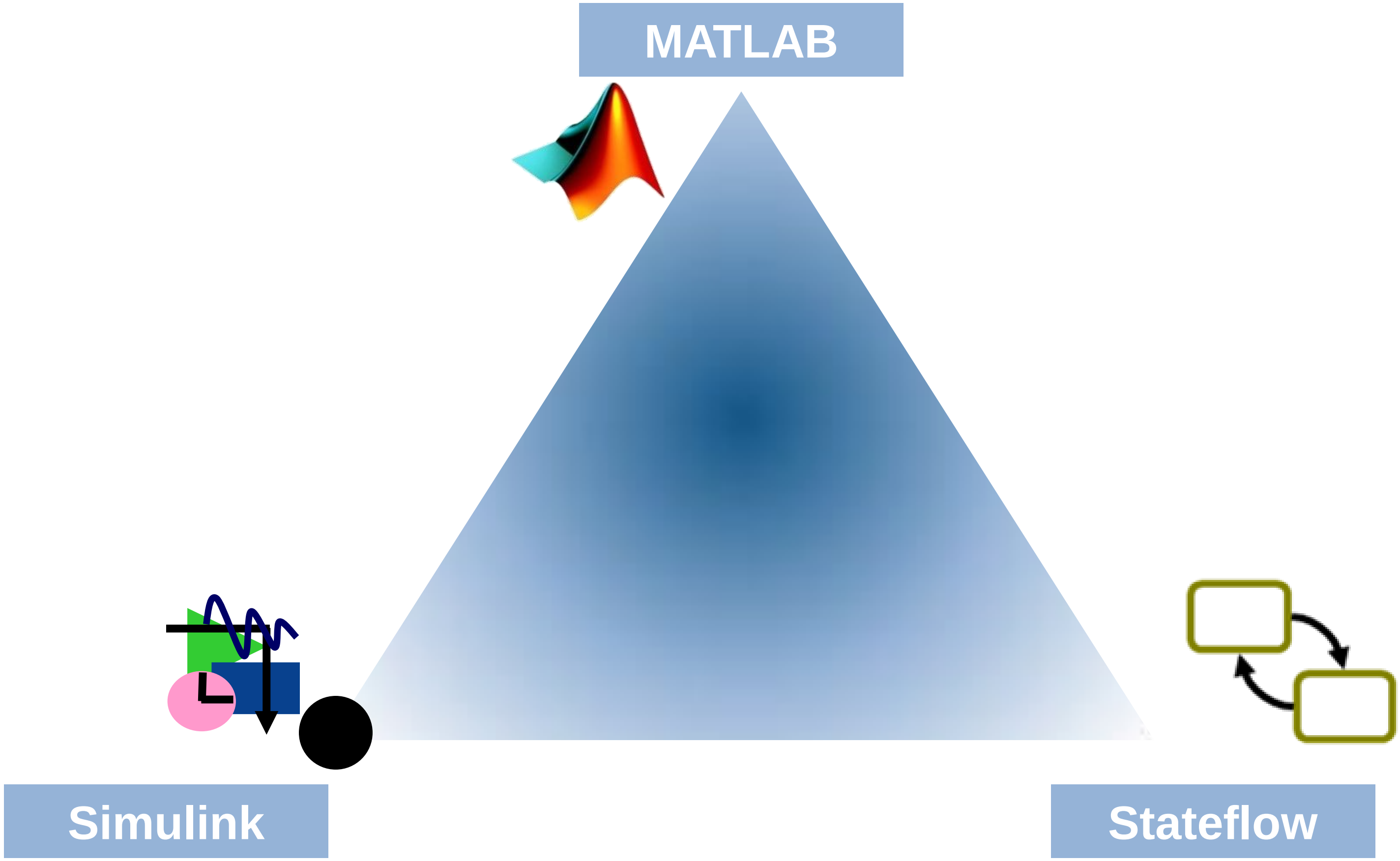


MATLAB

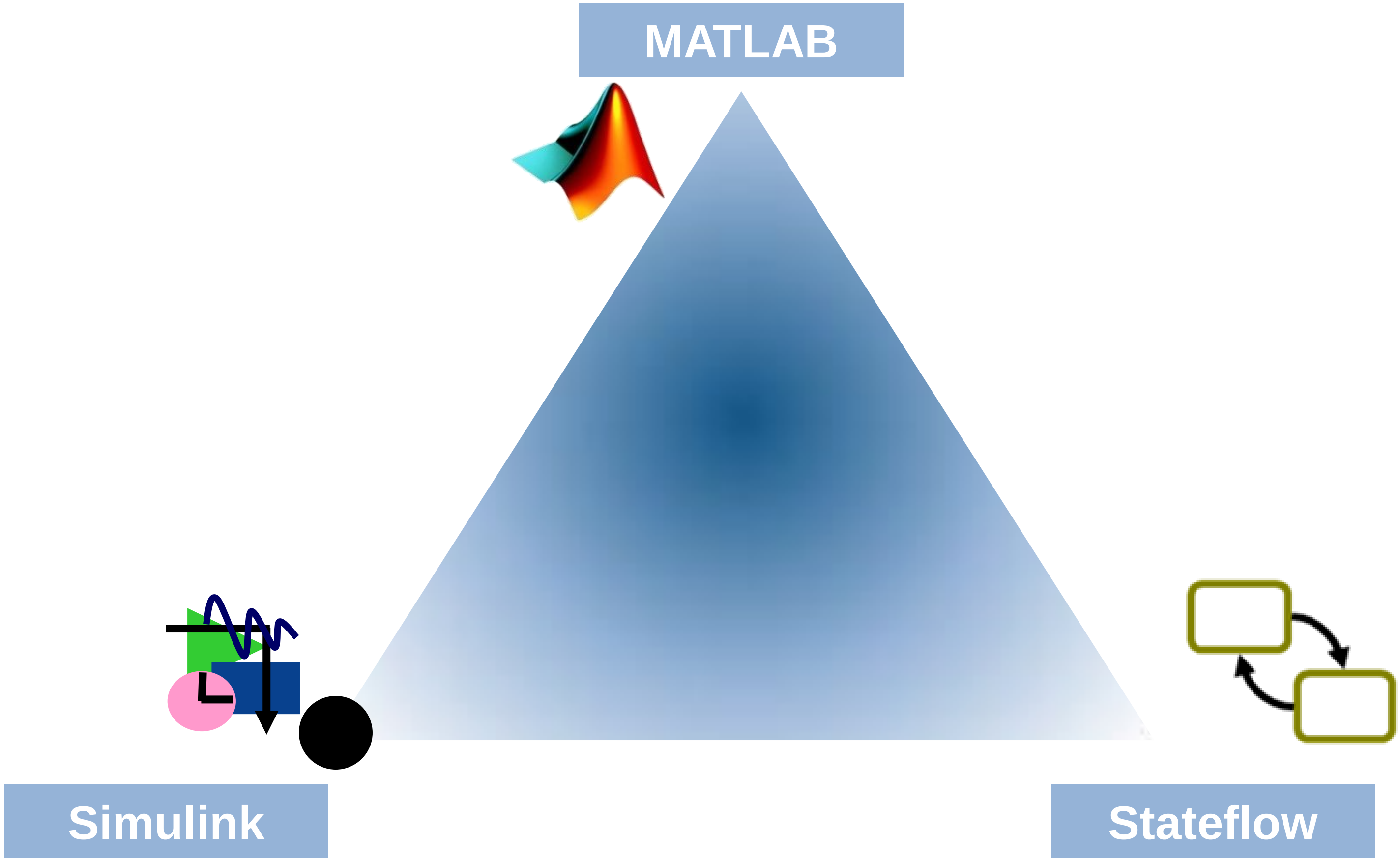
Simulink

Stateflow

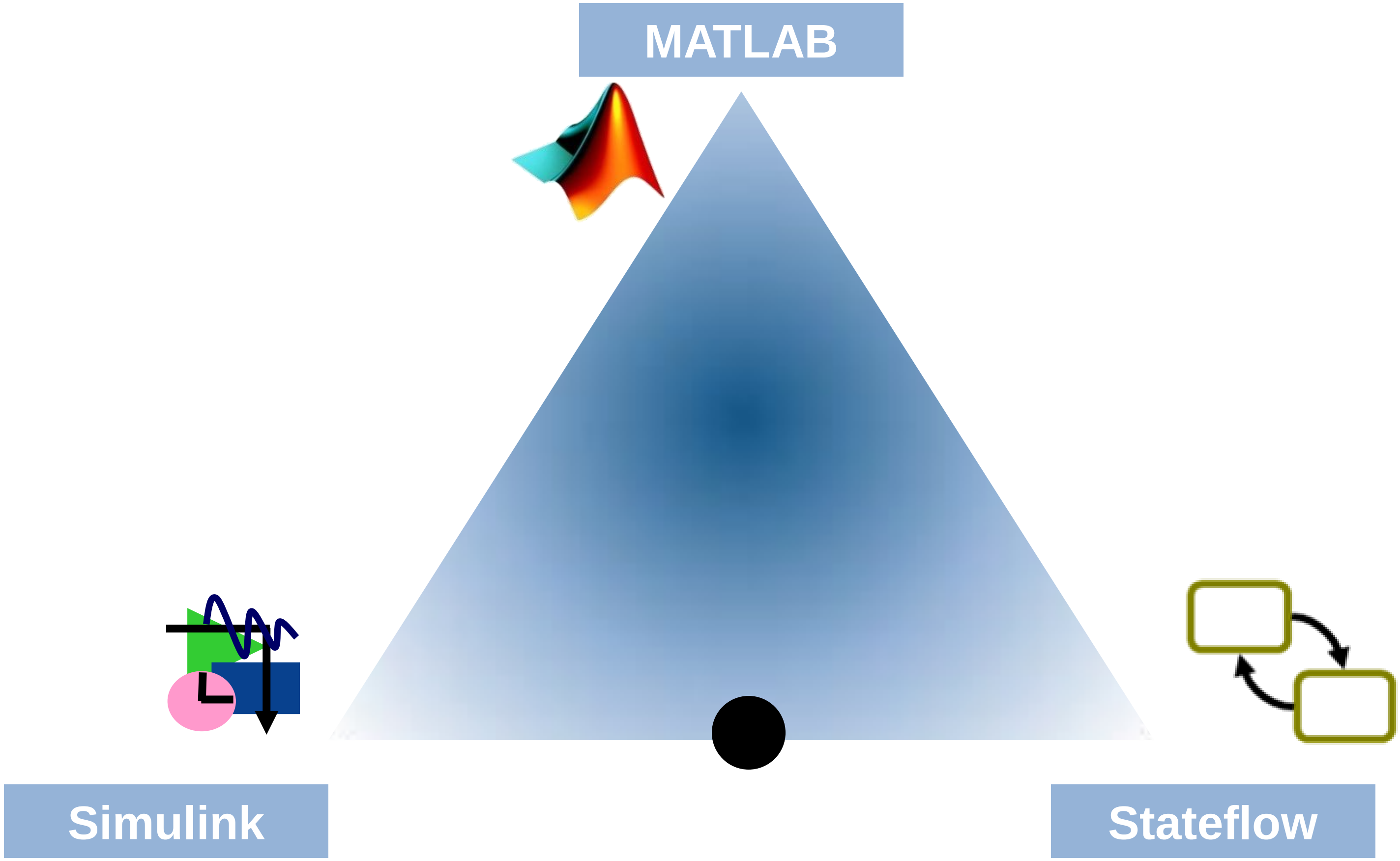
User Case –Certification, System overview



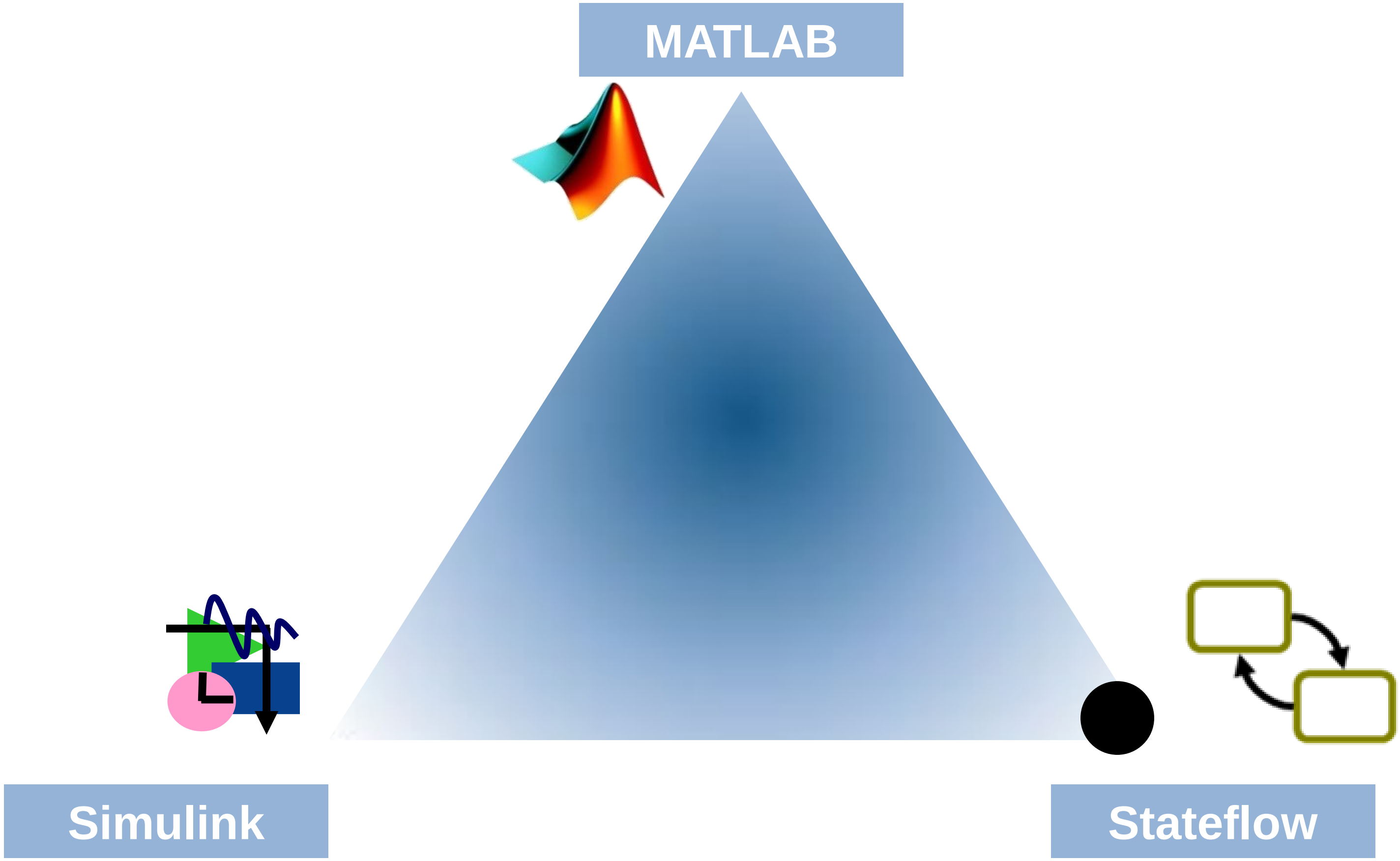
User Case – Multi-Domain, Mixed signals, Multi-Rate, ...



User Case – Concurrent States



User Case – Nested if then else

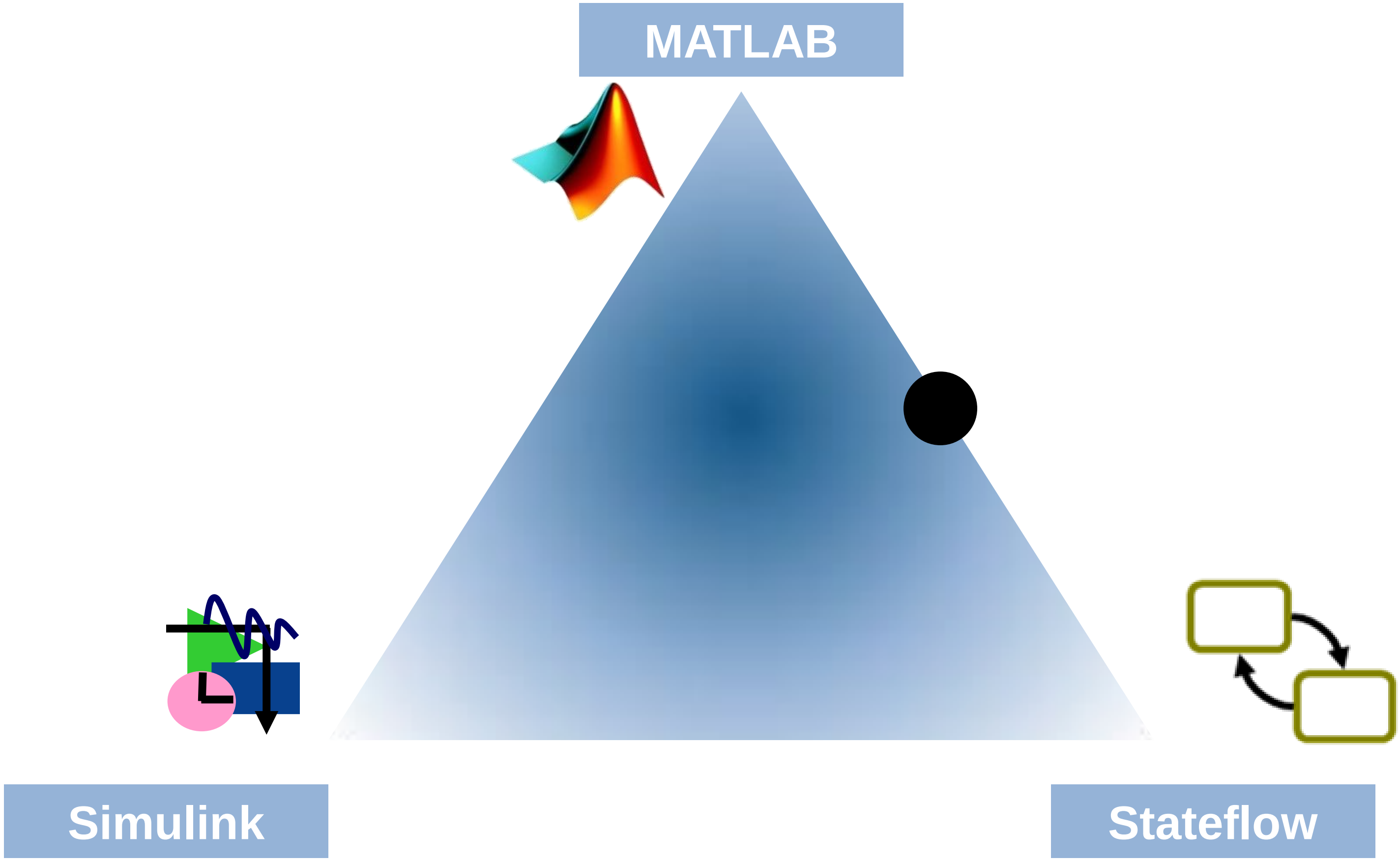


MATLAB

Simulink

Stateflow

User Case – if then else

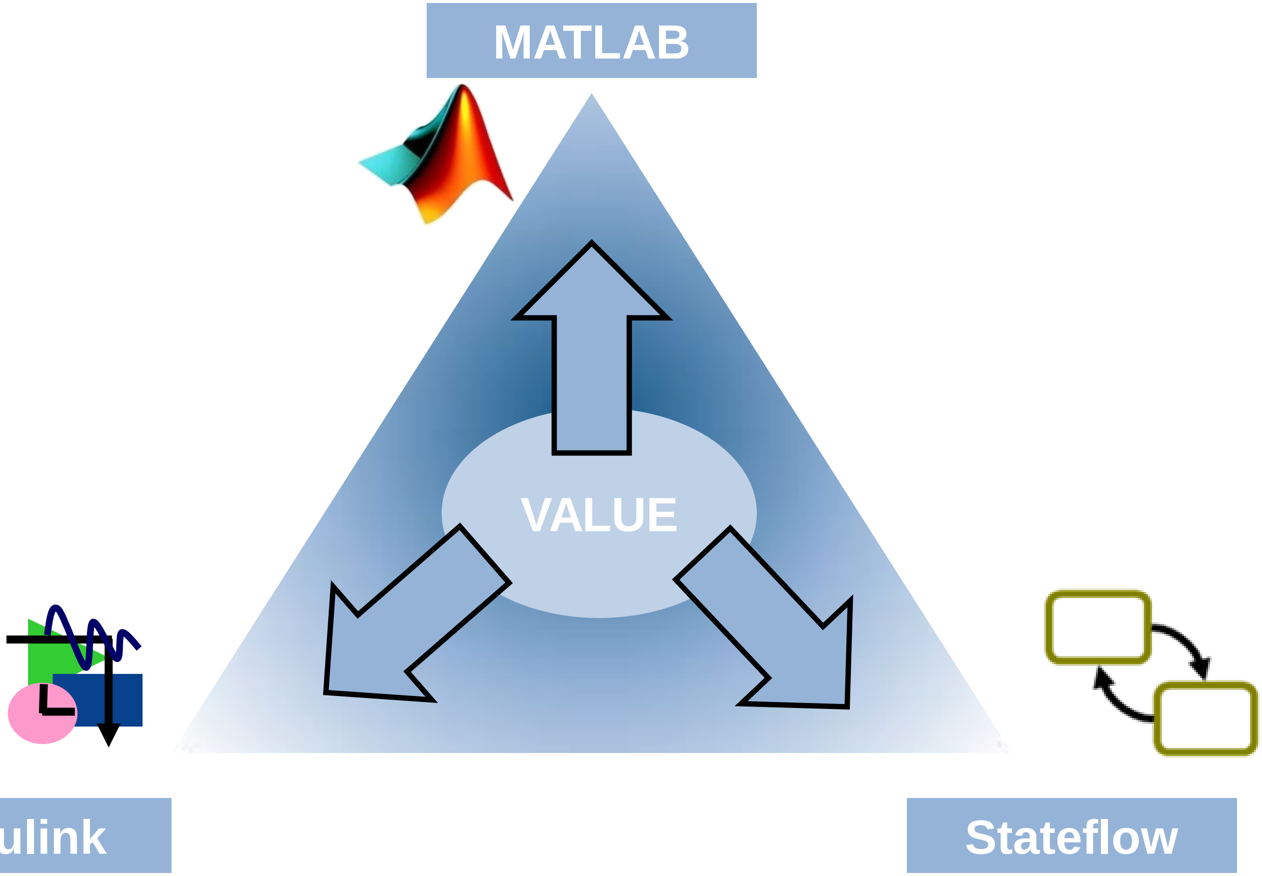


MATLAB

Simulink

Stateflow

Having the Choice is the Real Value!


MATLAB

VALUE

Simulink

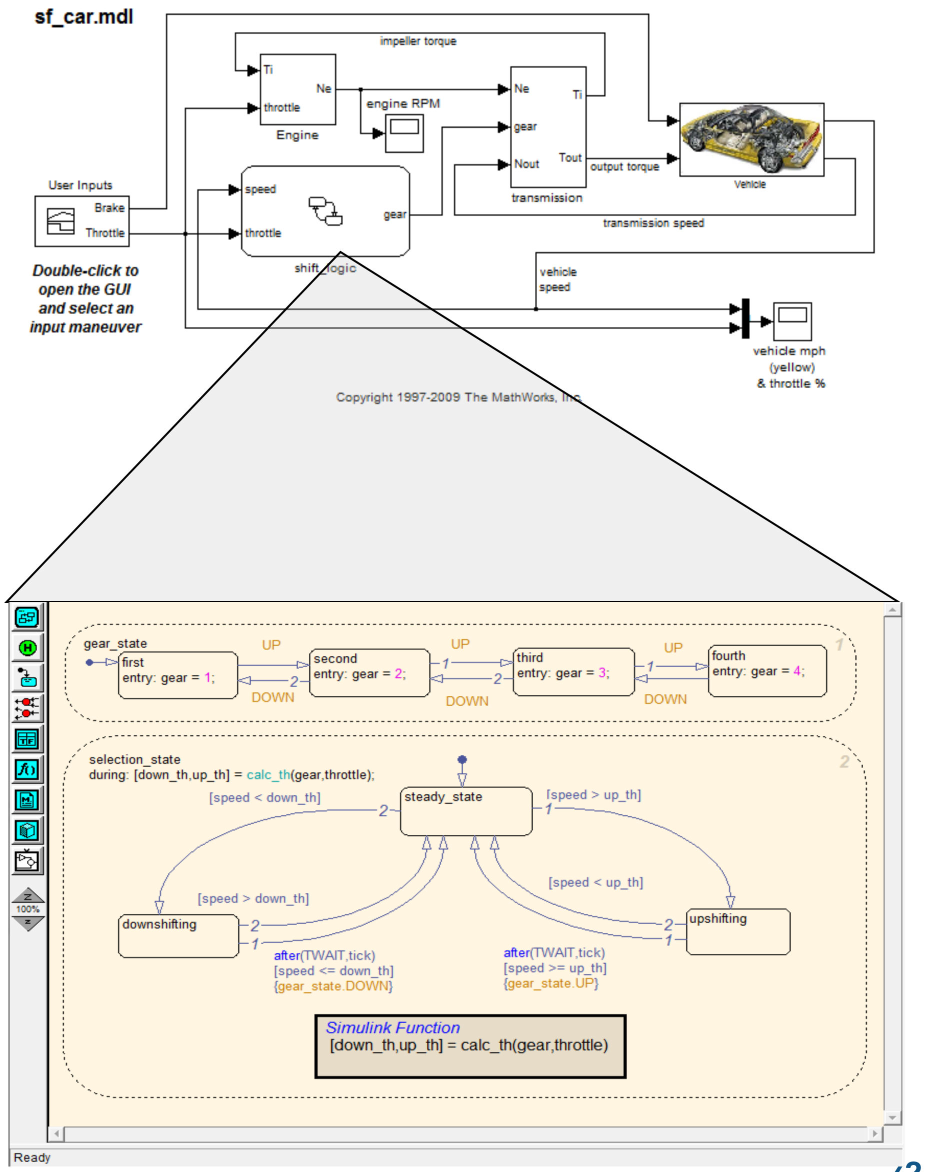
Stateflow

What is Stateflow?

Extend Simulink with state charts and flow graphs

Design supervisory control, scheduling, and mode logic

Model state discontinuities and instantaneous events



How Does Stateflow Work with Simulink?

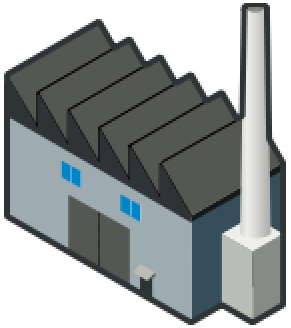
Simulink models **continuous** changes in dynamic systems.

Stateflow models **instantaneous** changes in dynamic systems.

Real-world systems have to respond to both continuous and instantaneous changes.

suspension dynamics
gear changes

propulsion system
liftoff stages



manufacturing robot
operation modes

Use both Simulink and Stateflow so that you can use the right tool for the right job.

Stateflow Concepts

States

- Exclusive
- Hierarchical
- Parallel

Transitions

- Default
- Conditions
- Condition Actions
- Event Triggers

Functions

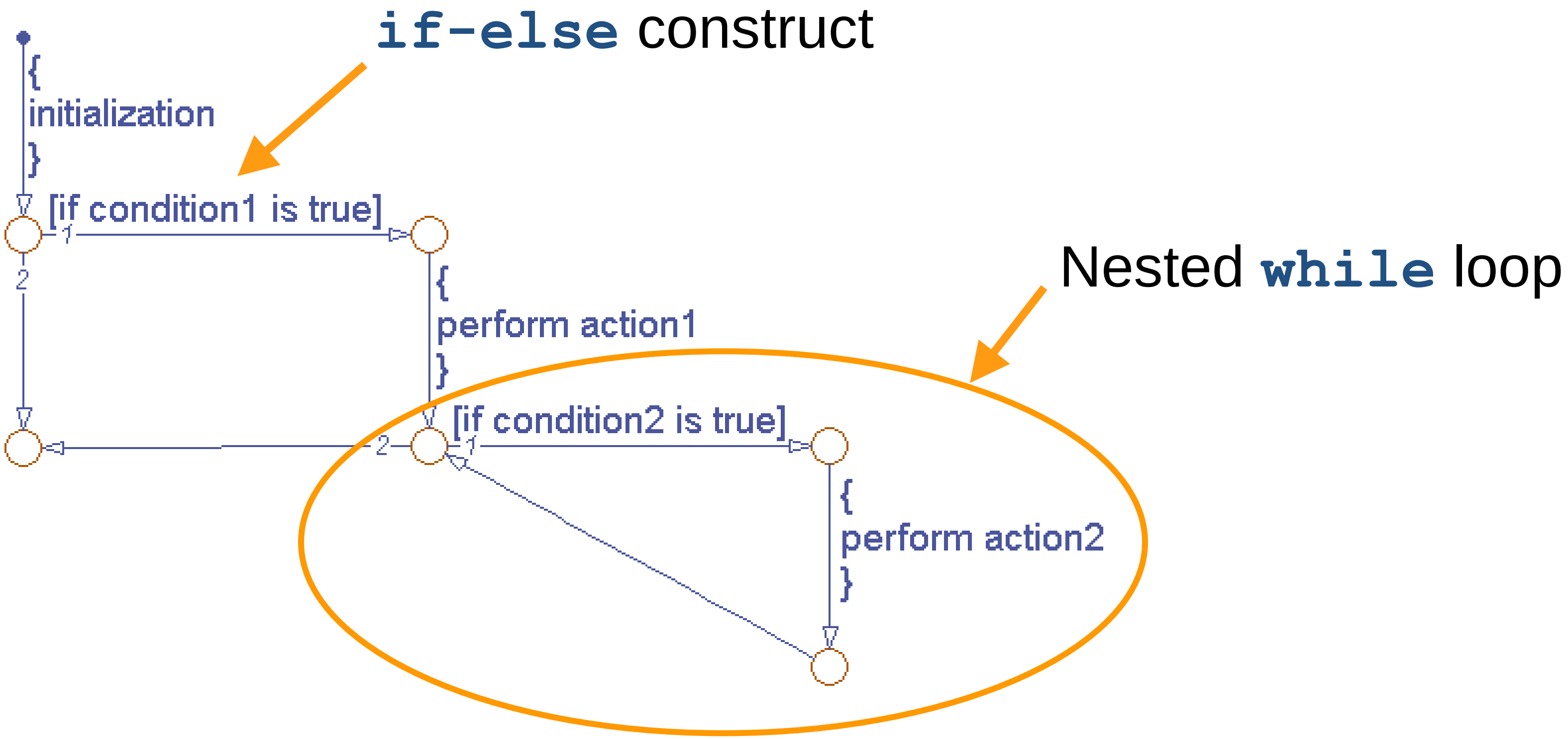
- Graphical
- Truth Tables
- MATLAB

Data

- Input/Output
- Local
- Model Explorer
- Add Menu
- Symbol Wizard

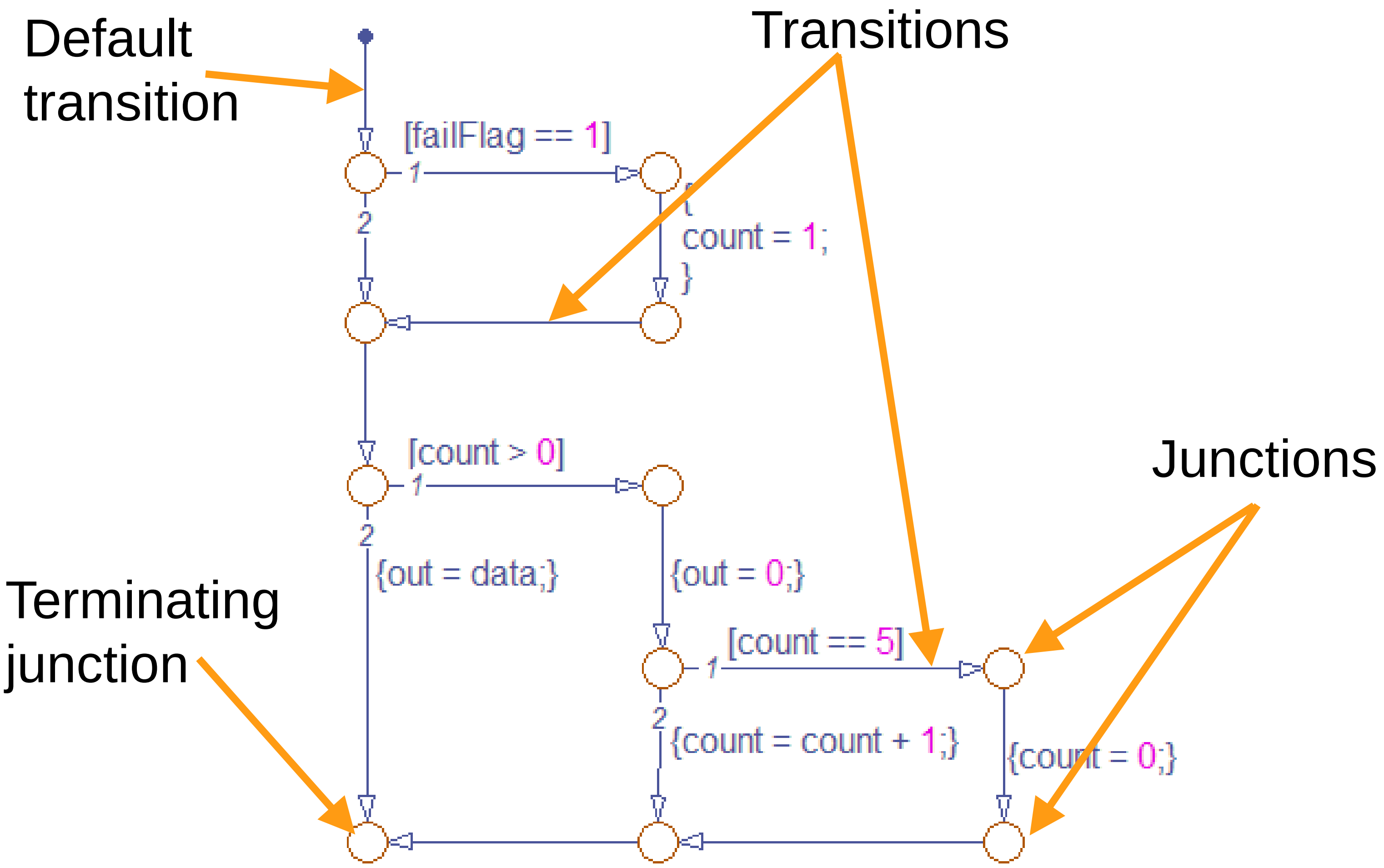
What Is a Flow Graph?

A chain of logical patterns that implement a series of decision flows

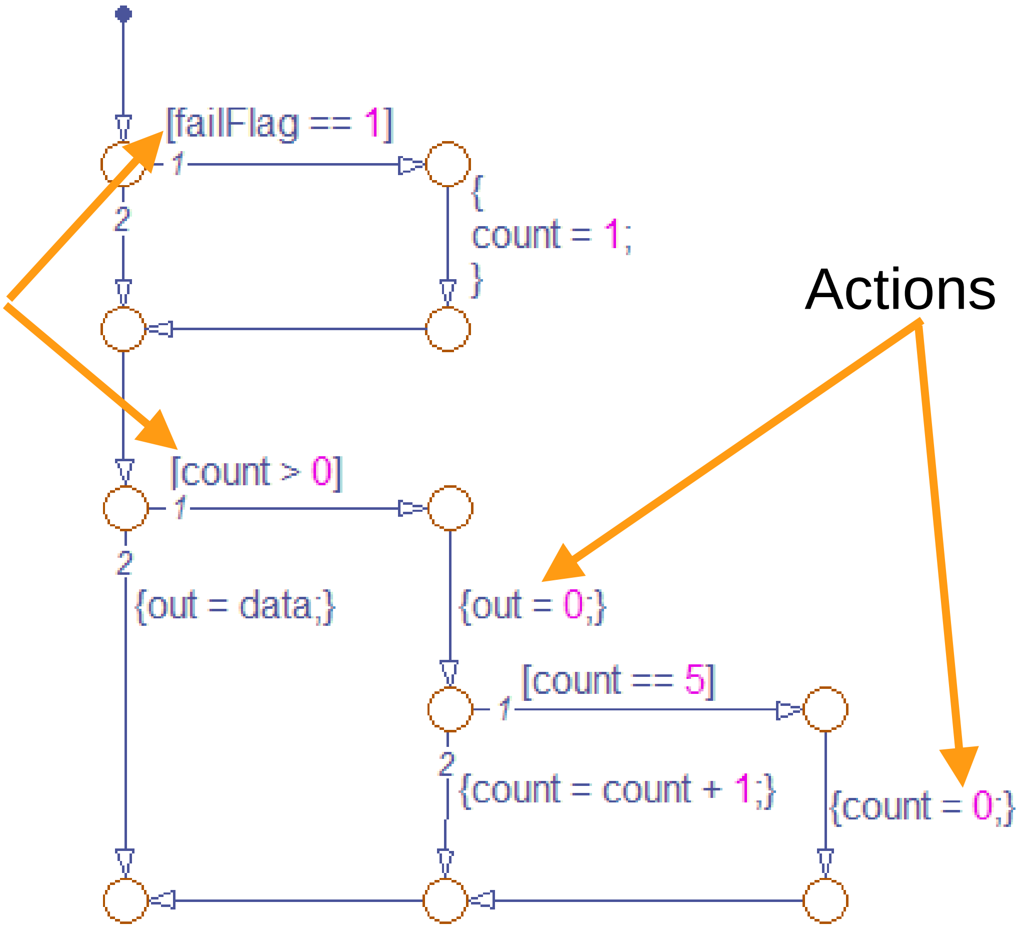


Can implement sequential, nested, and iterative flows

Junctions and Transitions



Conditions and Actions

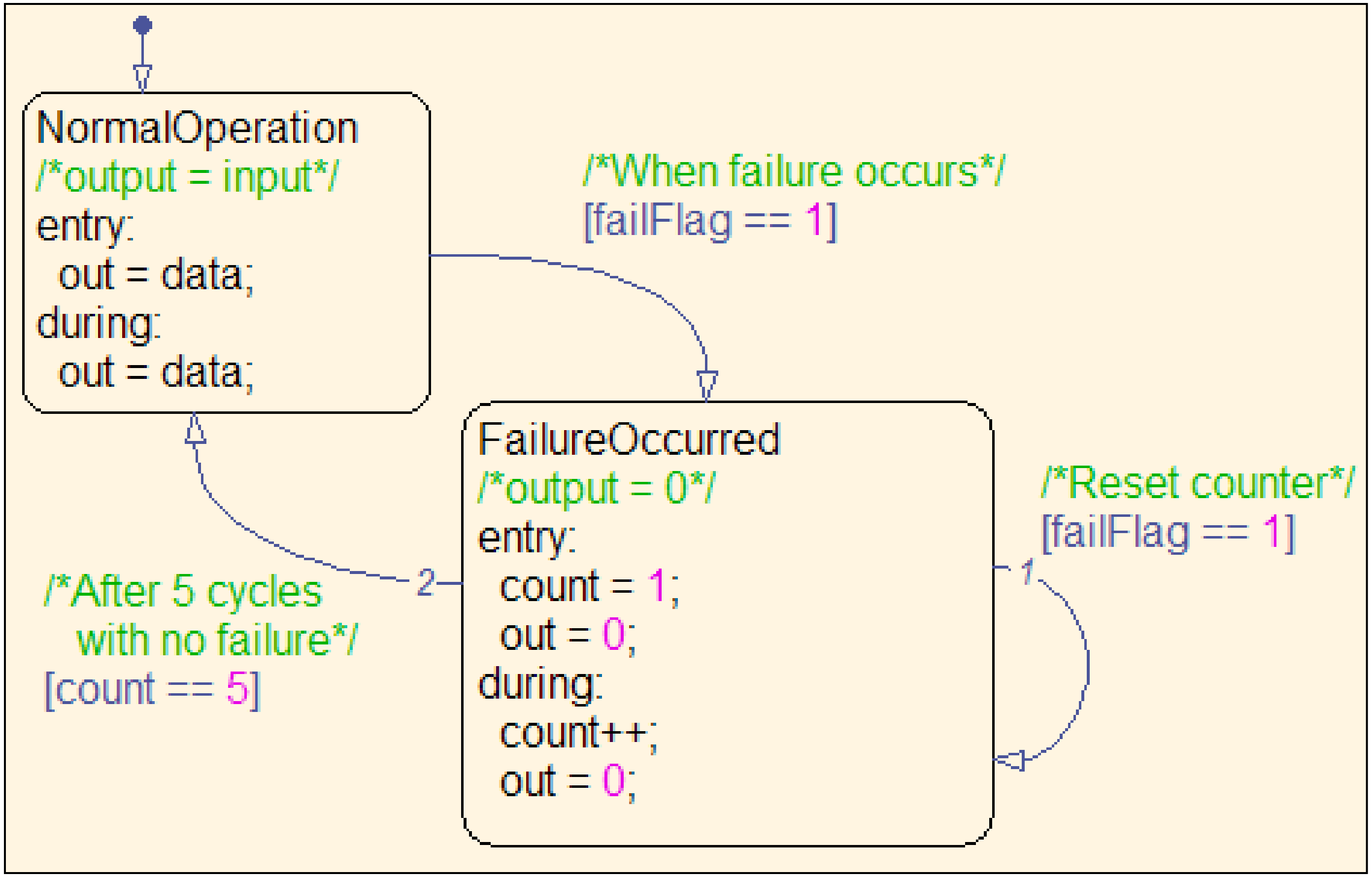


Conditions

Actions

What Is a State Machine?

- A system that can only exist in a finite number of modes
- Can only behave in a predefined number of ways



```
NormalOperation
/*output = input*/
entry:
  out = data;
during:
  out = data;
```

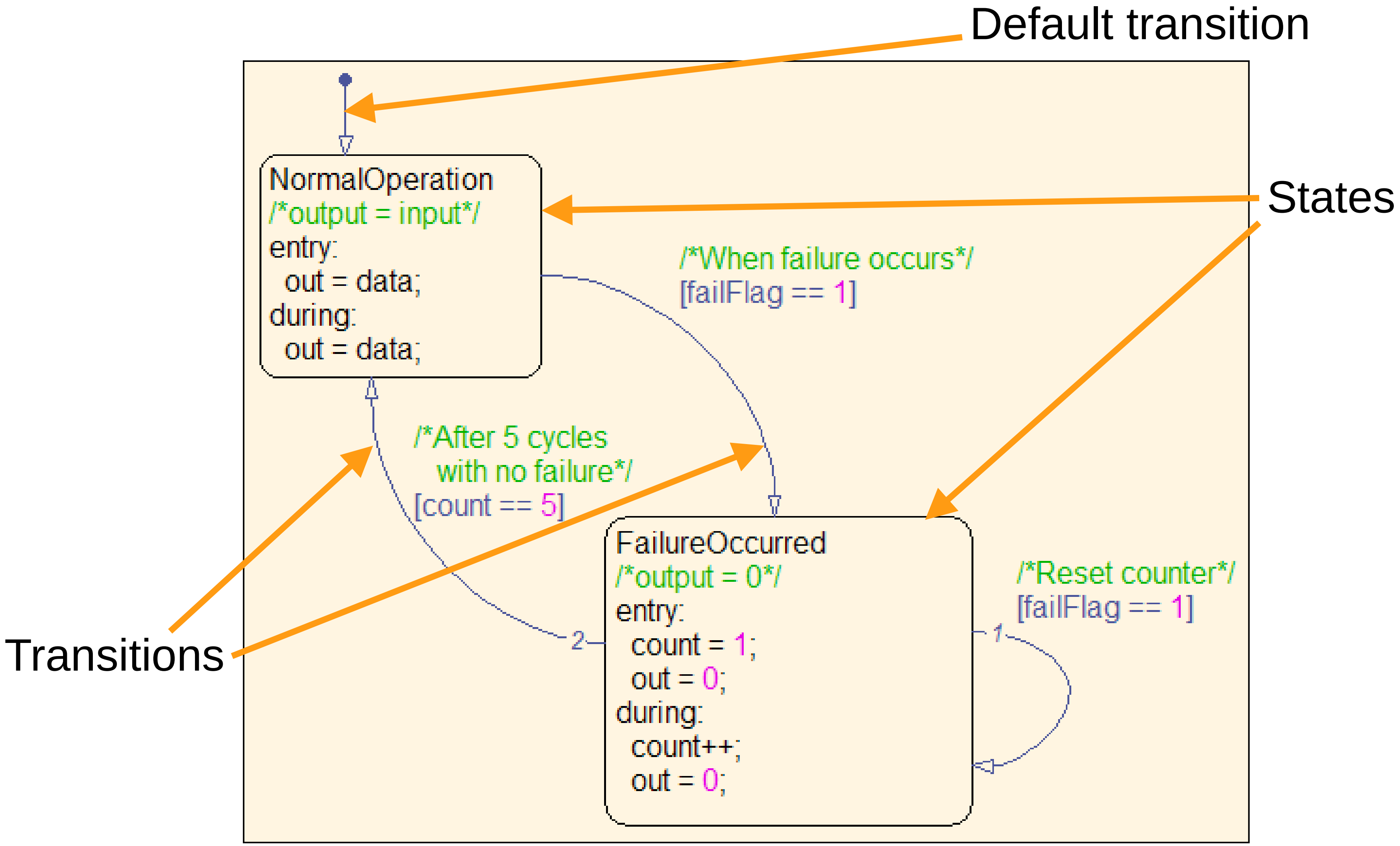
/*When failure occurs*/
[failFlag == 1]

```
FailureOccurred
/*output = 0*/
entry:
  count = 1;
  out = 0;
during:
  count++;
  out = 0;
```

/*Reset counter*/
[failFlag == 1]

/*After 5 cycles
with no failure*/
[count == 5]

States and Transitions



Default transition

States

Transitions

```
NormalOperation
/*output = input*/
entry:
  out = data;
during:
  out = data;
```

```
/*When failure occurs*/
[failFlag == 1]
```

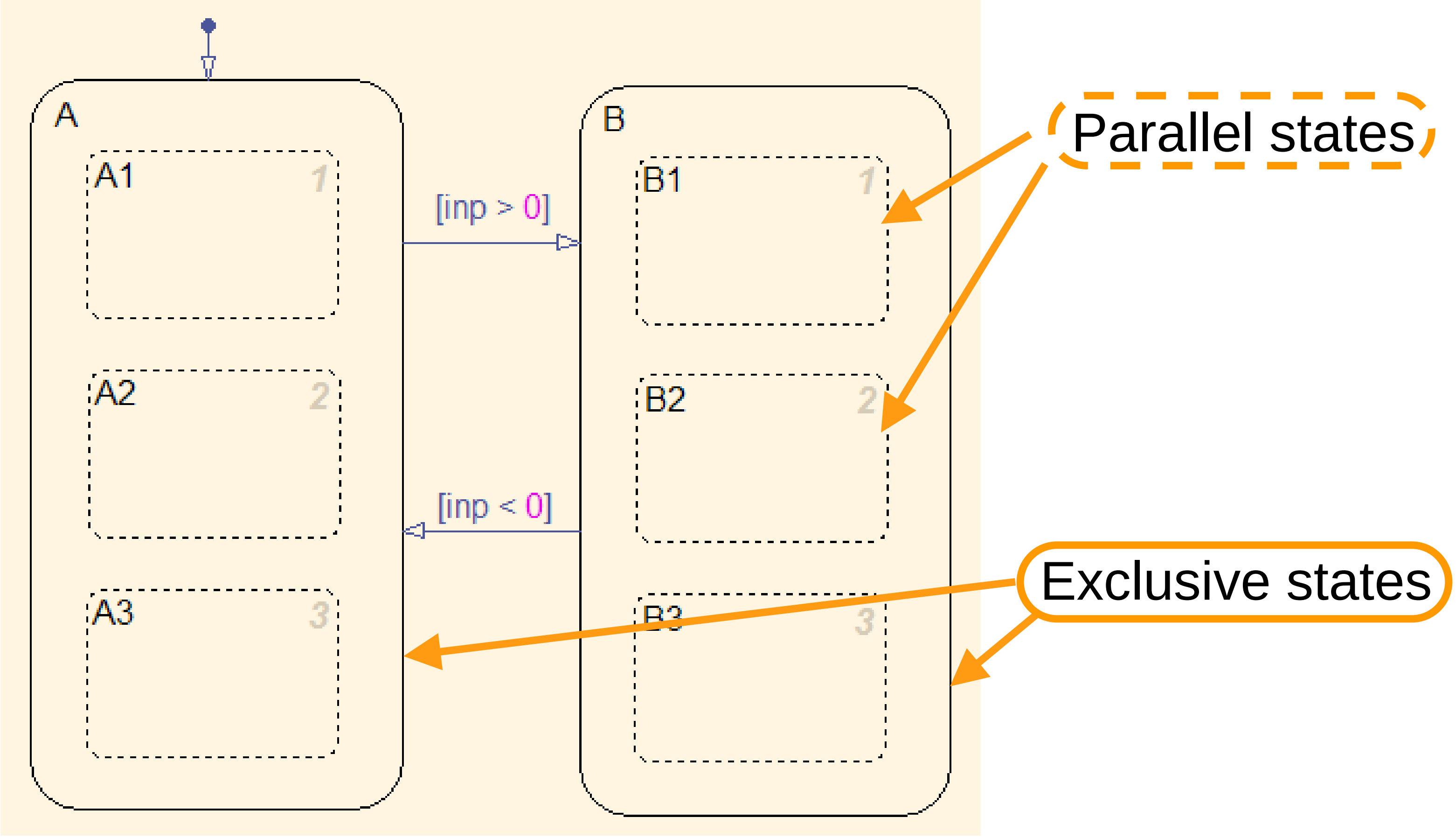
```
/*After 5 cycles
   with no failure*/
[count == 5]
```

```
FailureOccurred
/*output = 0*/
entry:
  count = 1;
  out = 0;
during:
  count++;
  out = 0;
```

```
/*Reset counter*/
[failFlag == 1]
```

The Concepts of Parallelism

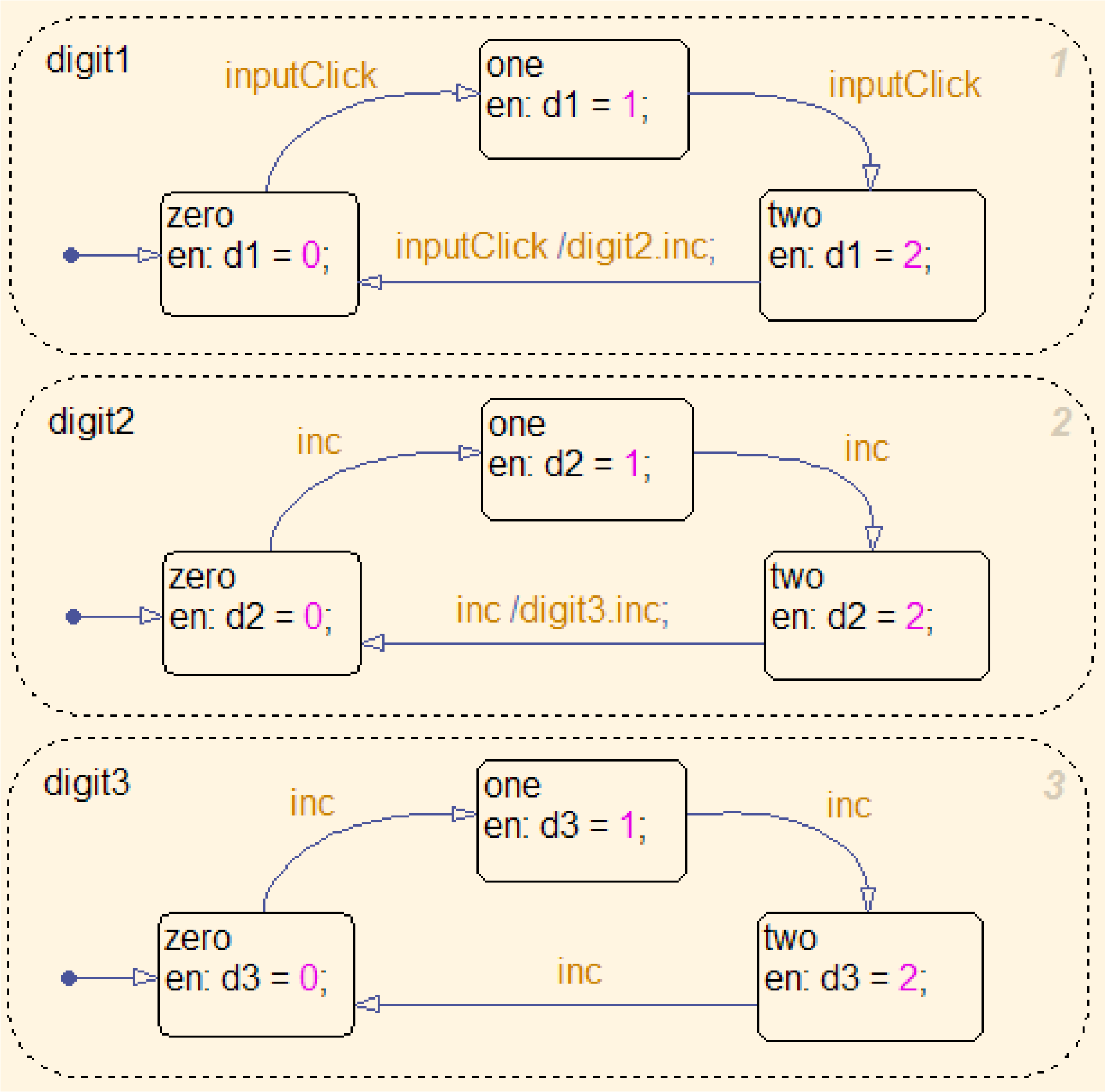
- Parallel states enter when their parent activates.
- Transitions from or to parallel states are prohibited.



Parallel states

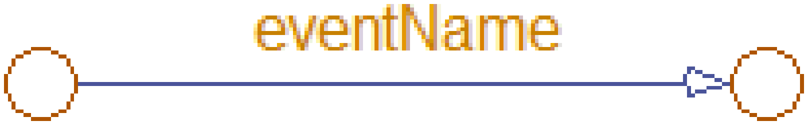
Exclusive states

An Example of Stateflow® Events

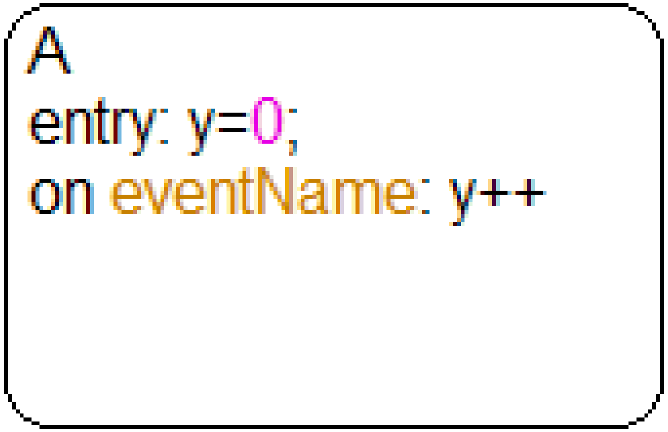


Using Events to Trigger Actions

- Guard transitions



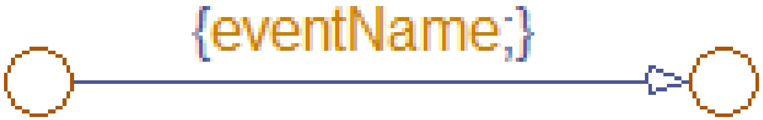
- Perform state actions (`on` keyword)



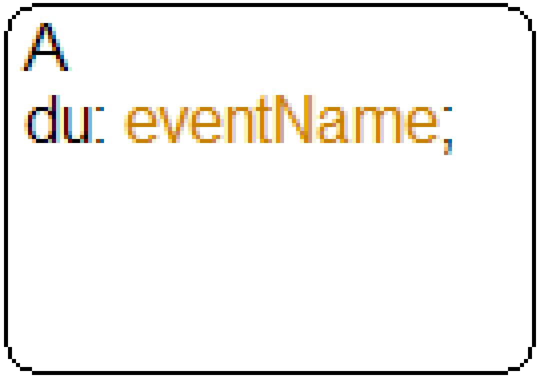
```
A
entry: y=0;
on eventName: y++
```

Broadcasting Events

- Use the event name to broadcast the event.
- This can be done anywhere that actions are specified (state actions, condition actions, and transition actions)

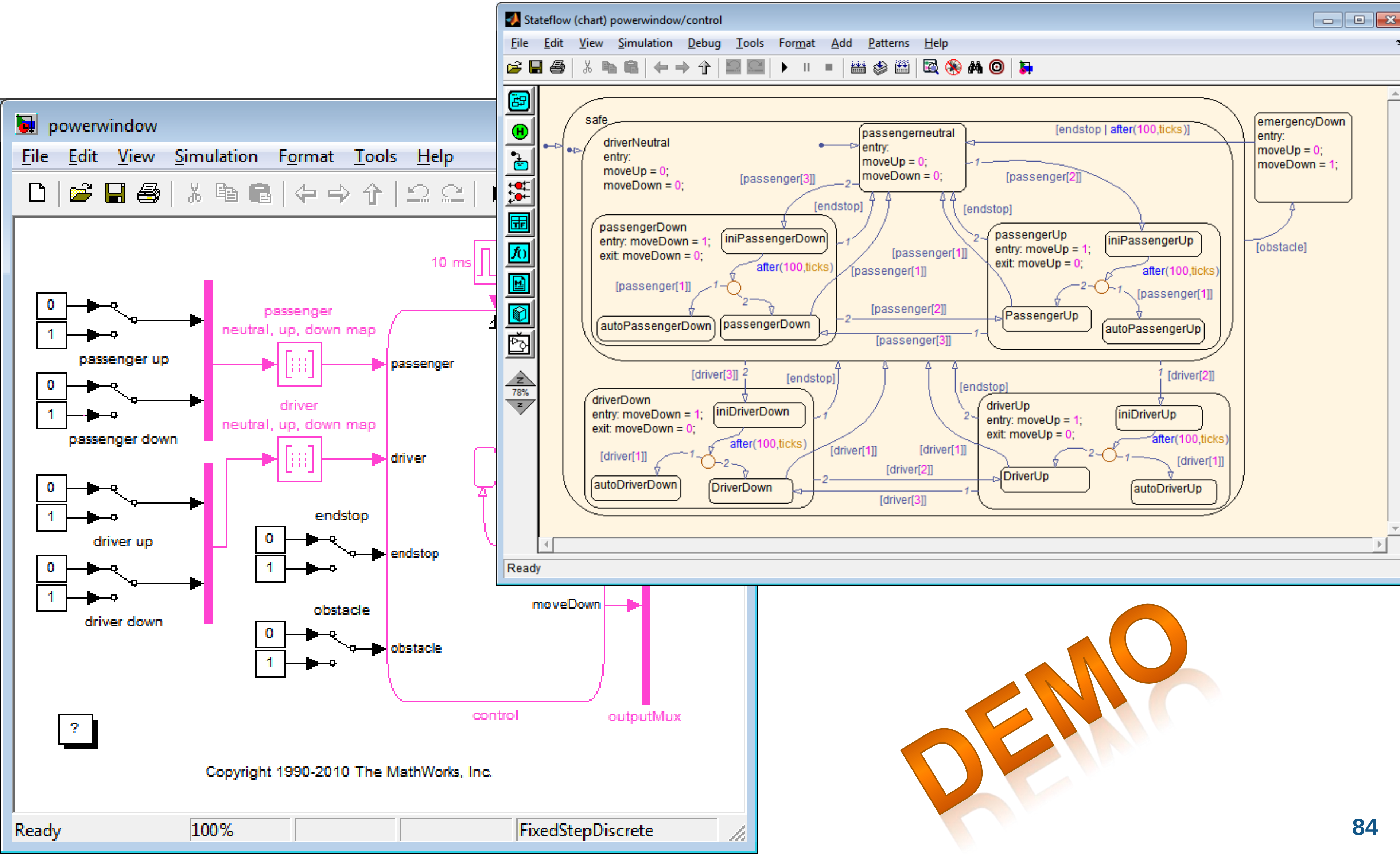


```
{eventName;}
```



```
A
du: eventName;
```

Power Window Example

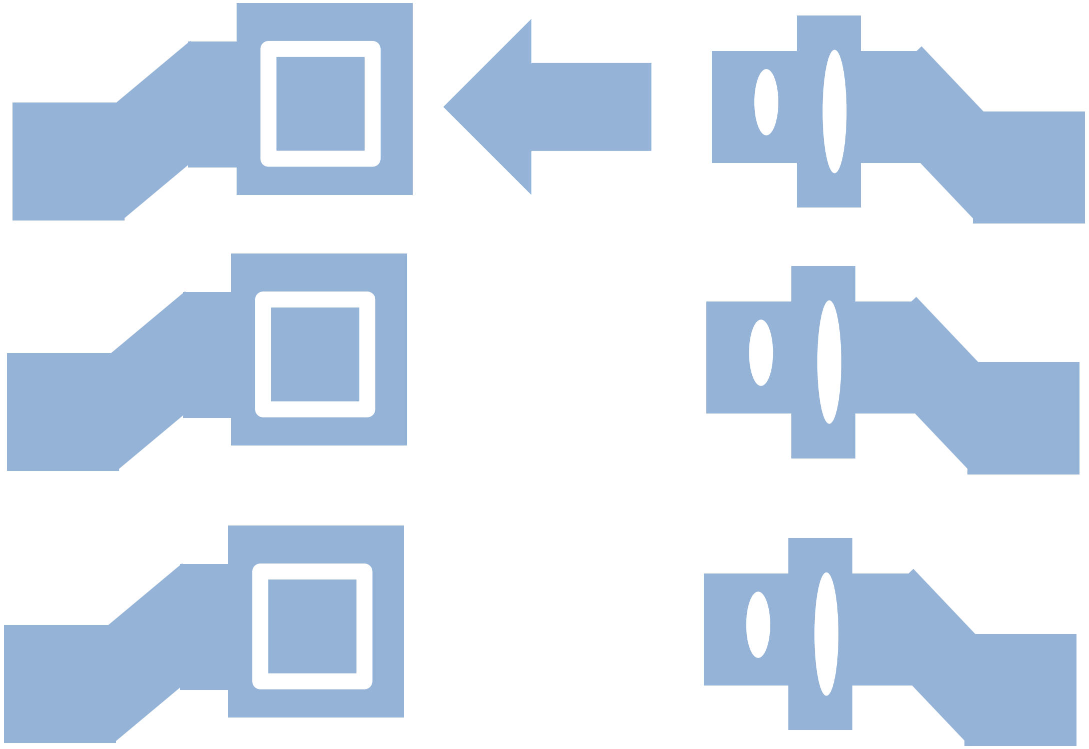


DEMO

Introduction to Verification and Validation

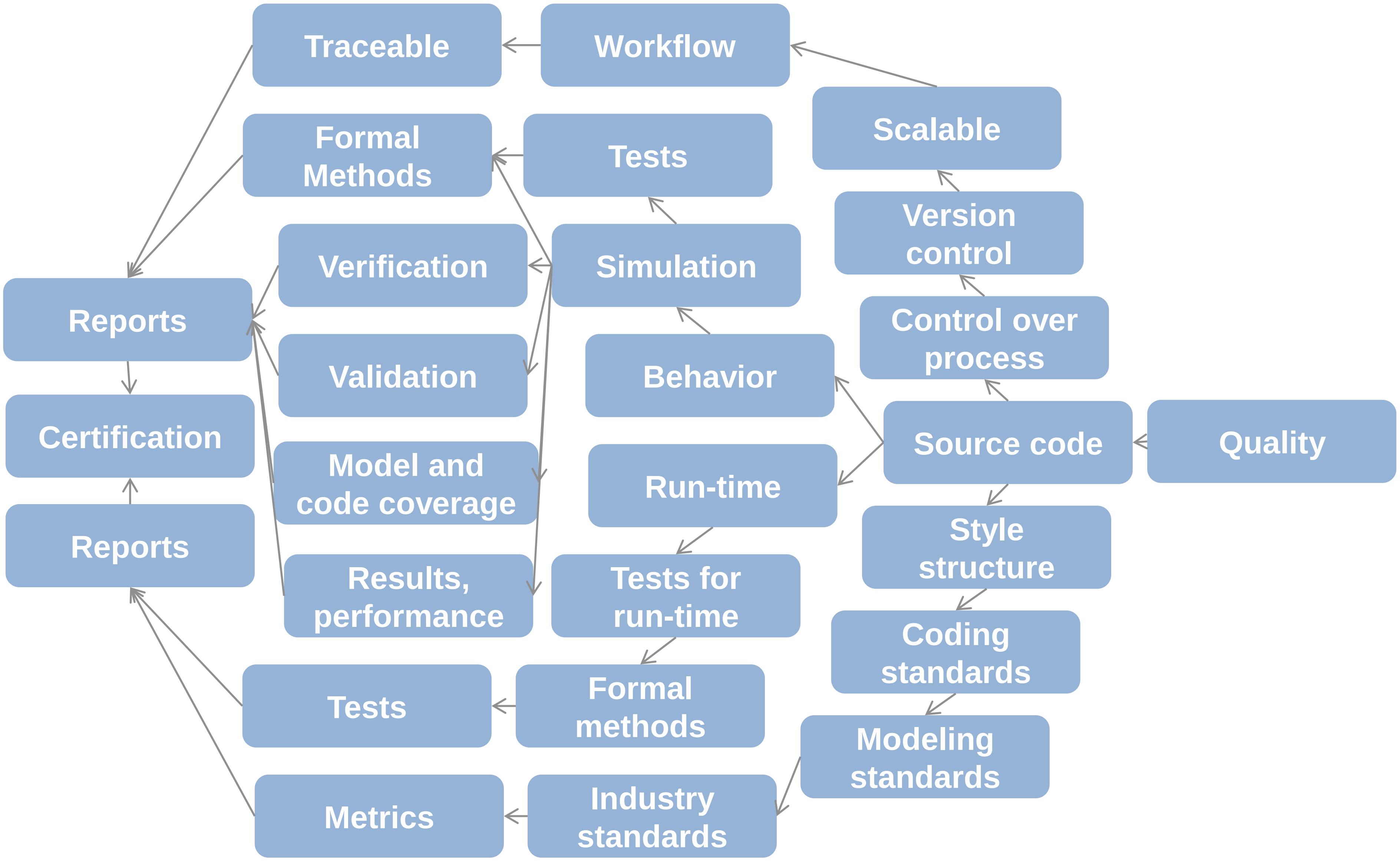
Where does engineering go wrong…

Design
Error?



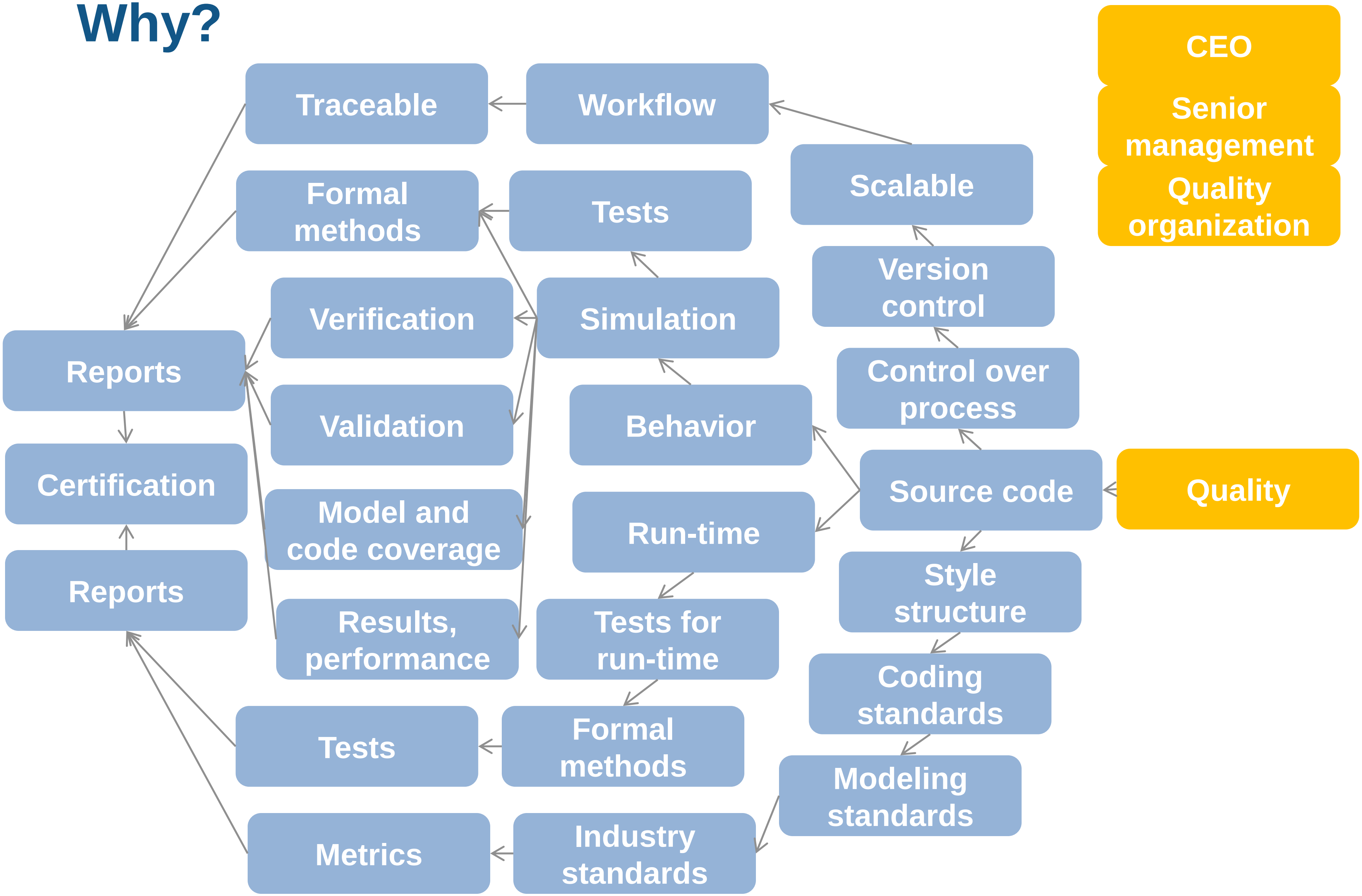
Missing
Test?

Why?



Traceable
Workflow
Scalable
Formal Methods
Tests
Version control
Verification
Simulation
Reports
Control over process
Validation
Behavior
Certification
Source code
Quality
Model and code coverage
Run-time
Reports
Style structure
Results, performance
Tests for run-time
Coding standards
Tests
Formal methods
Modeling standards
Metrics
Industry standards

Why?


CEO

Senior management

Quality organization

Traceable

Workflow

Formal methods

Tests

Scalable

Version control

Verification

Simulation

Reports

Control over process

Validation

Behavior

Certification

Source code

Quality

Model and code coverage

Run-time

Reports

Style structure

Results, performance

Tests for run-time

Coding standards

Tests

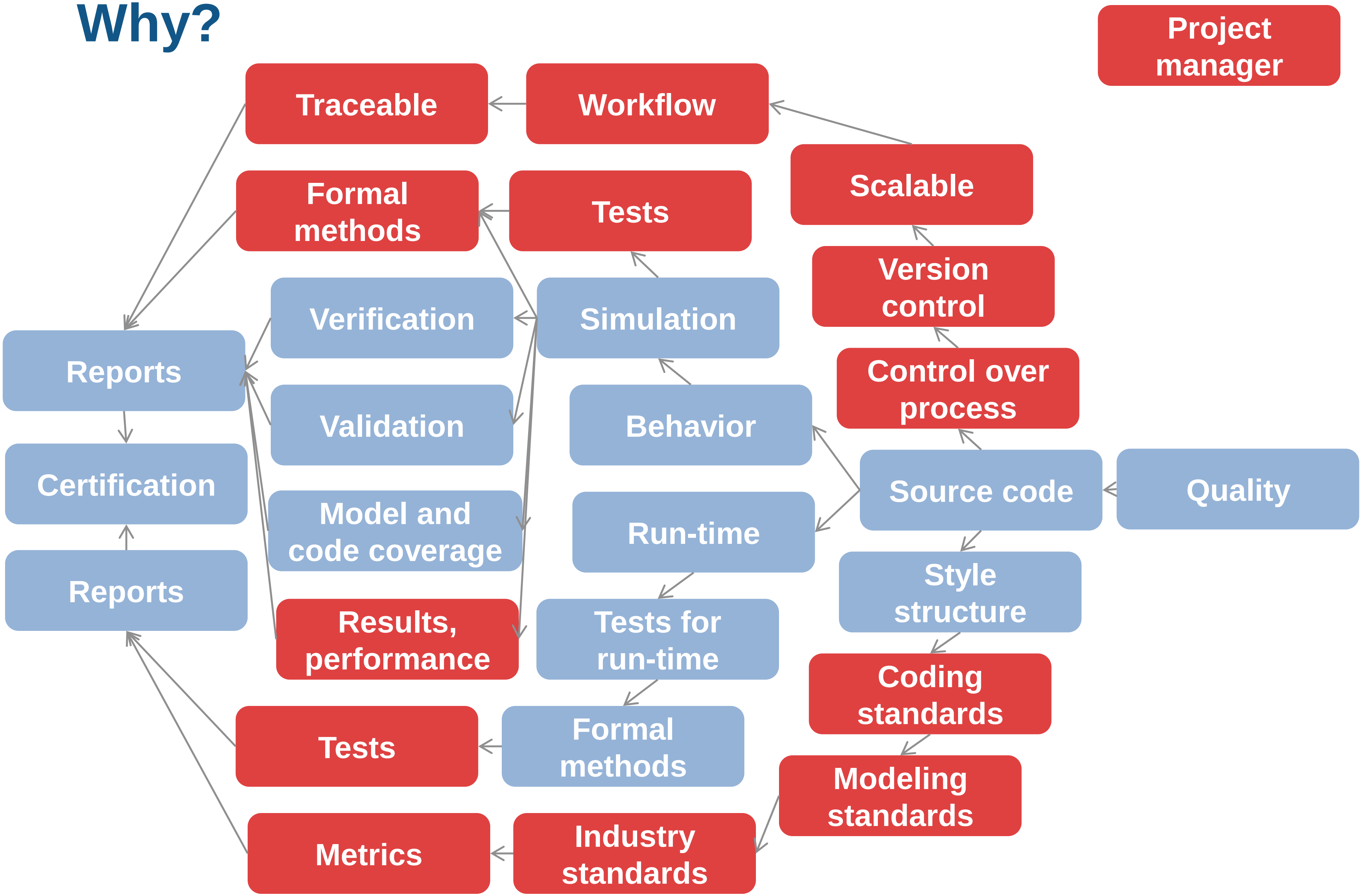
Formal methods

Modeling standards

Metrics

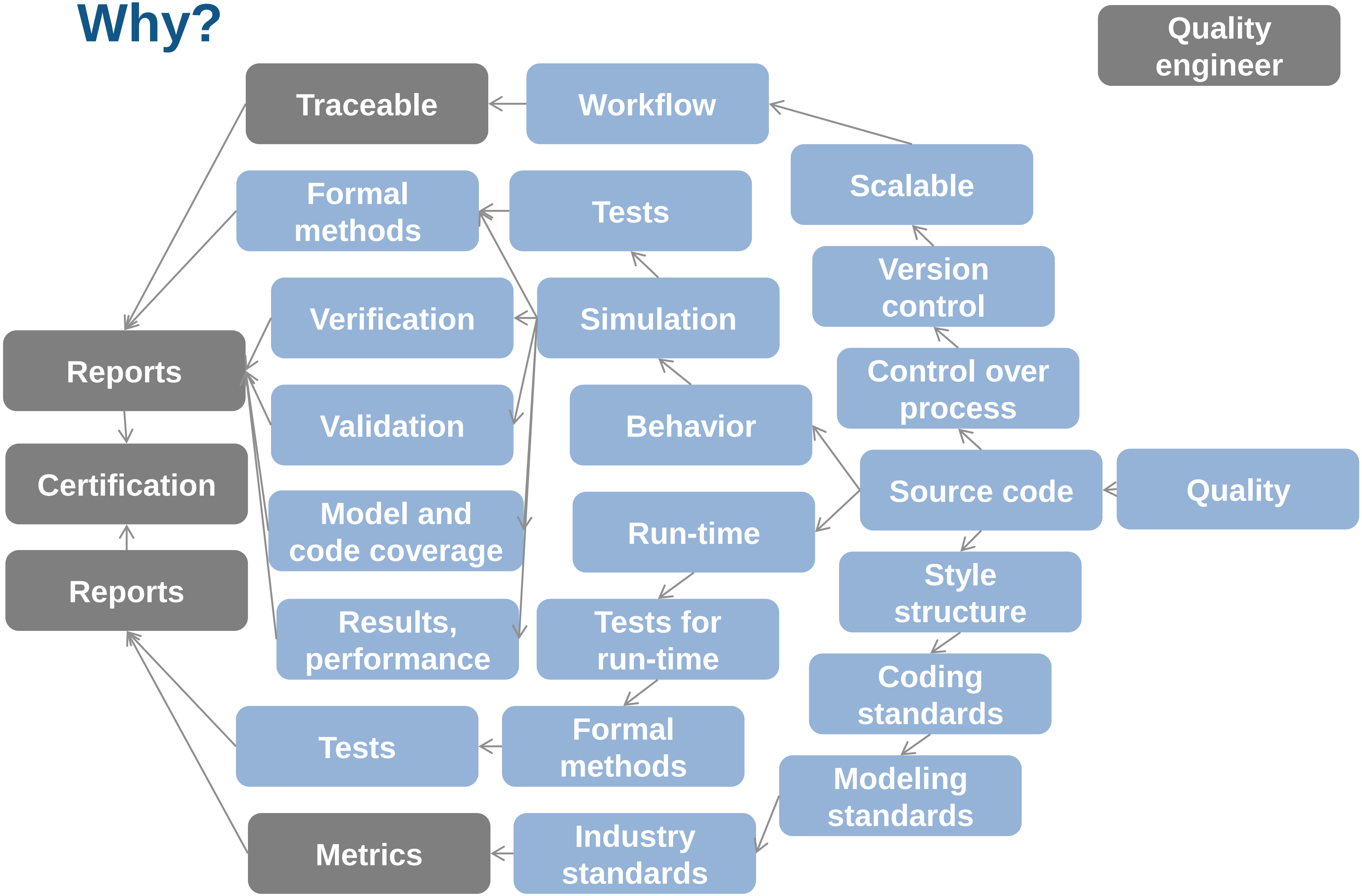
Industry standards

Why?

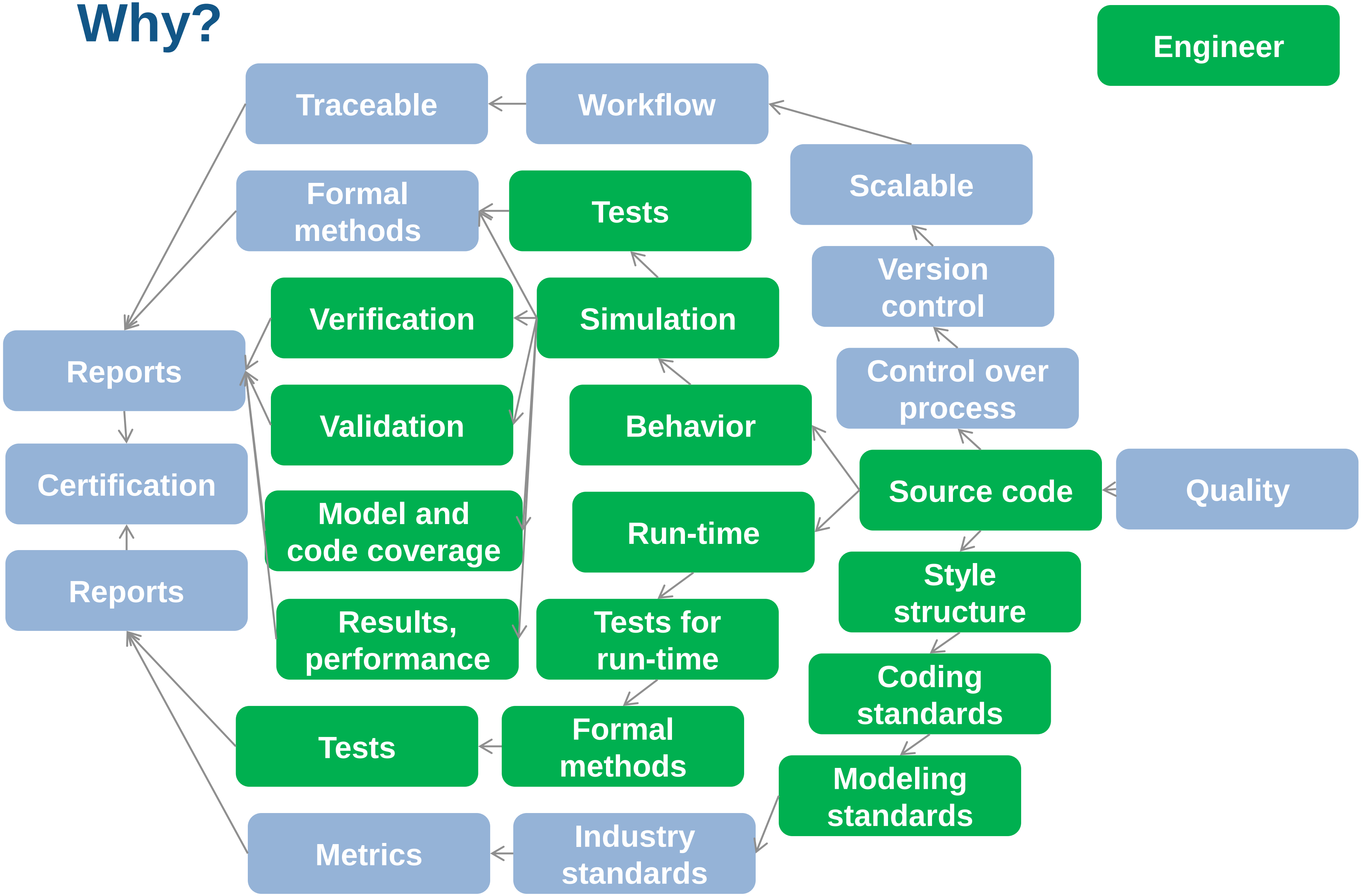


Project manager
Traceable
Workflow
Scalable
Formal methods
Tests
Version control
Verification
Simulation
Reports
Control over process
Validation
Behavior
Certification
Source code
Quality
Model and code coverage
Run-time
Reports
Style structure
Results, performance
Tests for run-time
Coding standards
Tests
Formal methods
Modeling standards
Metrics
Industry standards

Why?


Quality engineer
Traceable
Workflow
Scalable
Formal methods
Tests
Version control
Verification
Simulation
Control over process
Reports
Validation
Behavior
Certification
Model and code coverage
Run-time
Source code
Quality
Reports
Style structure
Results, performance
Tests for run-time
Coding standards
Tests
Formal methods
Modeling standards
Metrics
Industry standards

Why?


Engineer
Traceable
Workflow
Scalable
Formal methods
Tests
Version control
Verification
Simulation
Reports
Control over process
Validation
Behavior
Certification
Source code
Quality
Model and code coverage
Run-time
Reports
Style structure
Results, performance
Tests for run-time
Coding standards
Tests
Formal methods
Modeling standards
Metrics
Industry standards

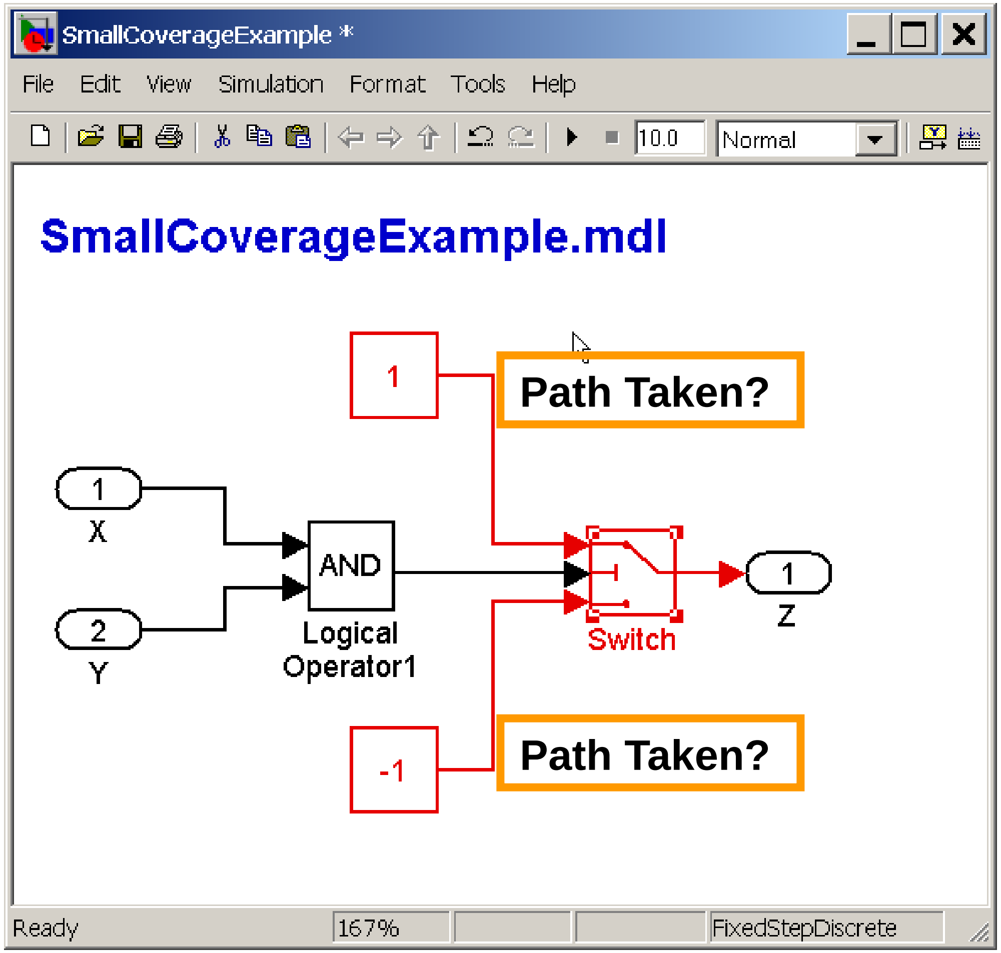
Why?

TO KEEP EVERYONE HAPPY!

- CEO, senior management
  - Want to improve quality for product
  - Yet need to remain competitive in price, time-to-market, feature content
- Project manager
  - Has to give reports to quality engineer
  - Has to make his team comply with standards
  - Creates more work
- Engineer
  - Has more work
  - Pushes back because need is not clear
- Quality engineer
  - Needs to work with everyone and bring them on board!
  - Finds it hard

Decision Coverage (DC)

Percentage of paths taken through decision point

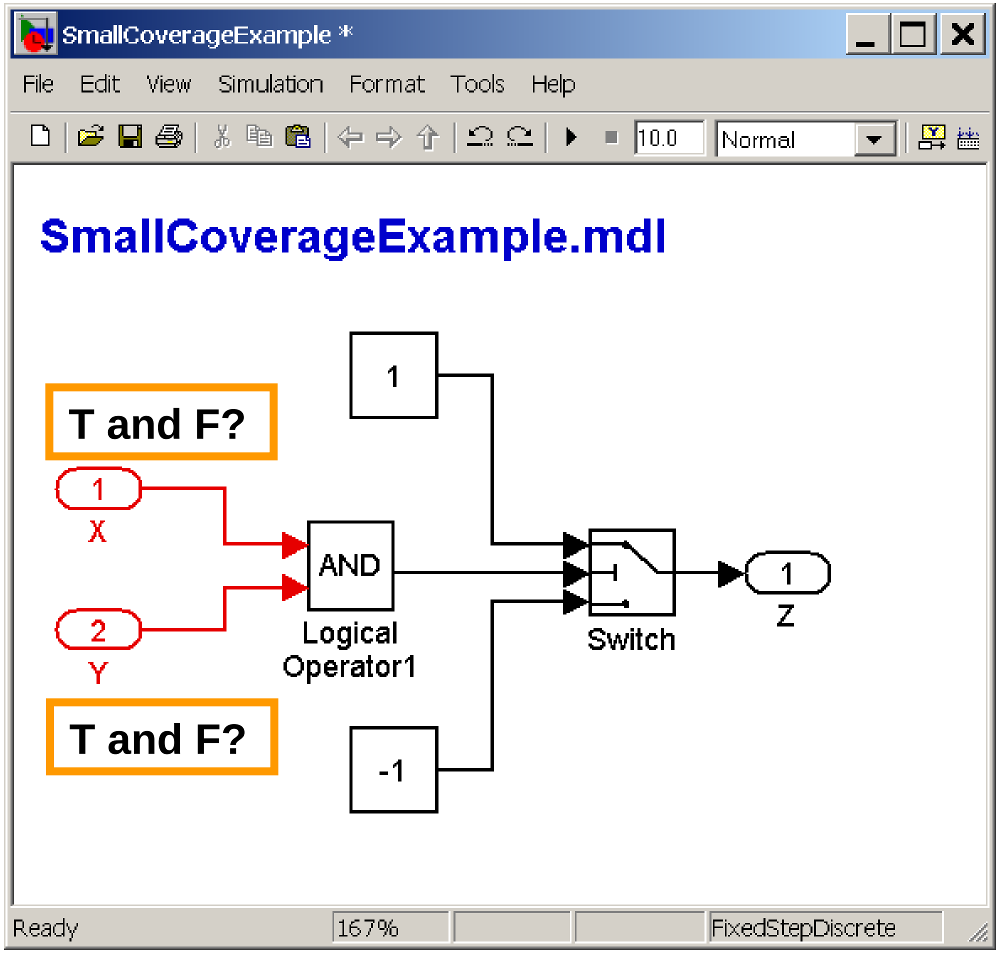


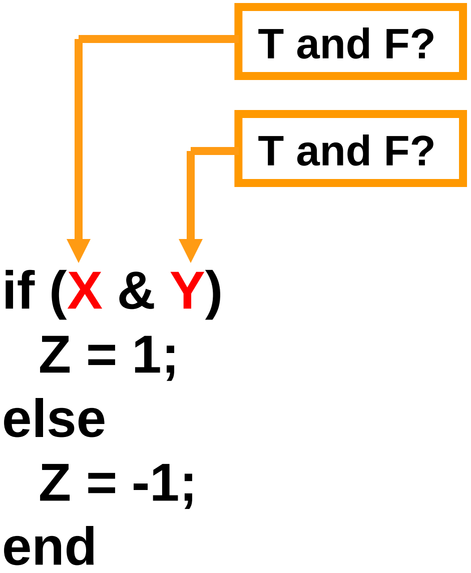
T and F?

```
if (X & Y)
  Z = 1;
else
  Z = -1;
end
```

Condition Coverage (CC)

Percentage of conditions exercised

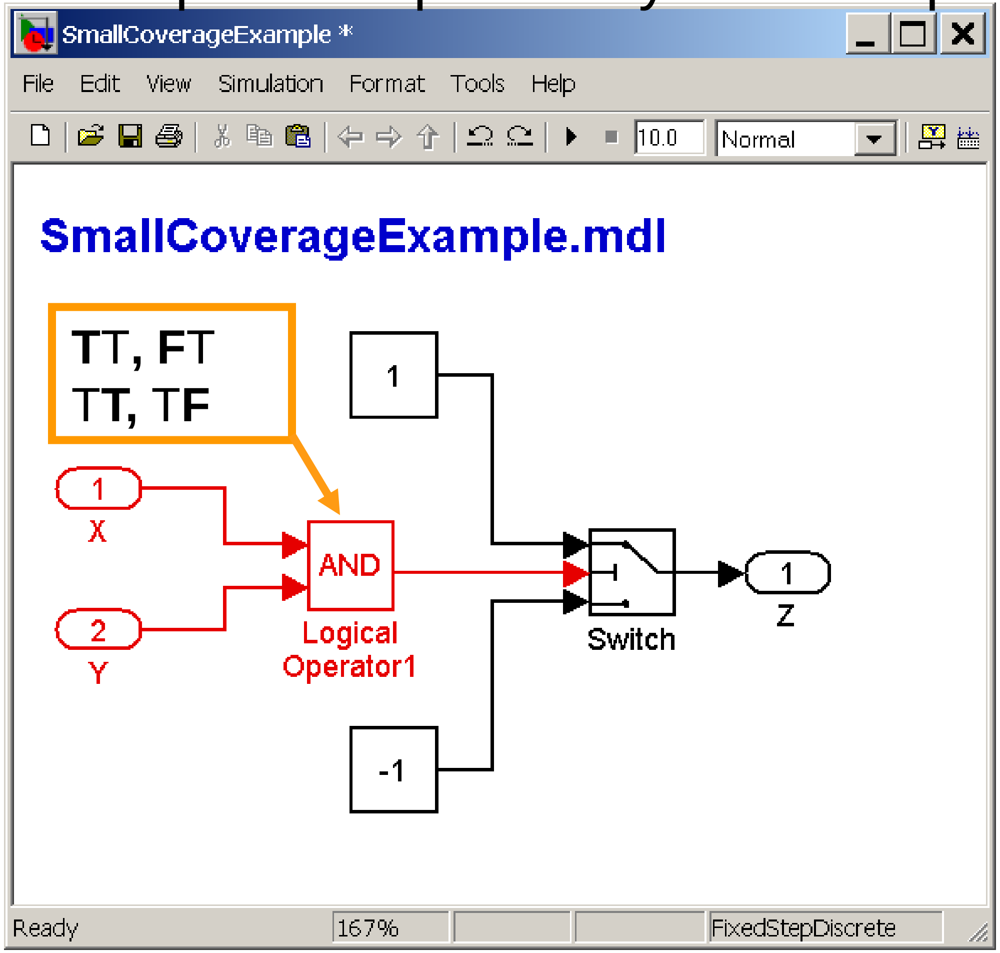




```
if (X & Y)
  Z = 1;
else
  Z = -1;
end
```

Modified Condition/Decision Coverage (MC/DC)

Checks inputs independently affect output



Stop time: 10.0

Simulation mode: Normal

SmallCoverageExample.mdl

TT, FT
TT, TF

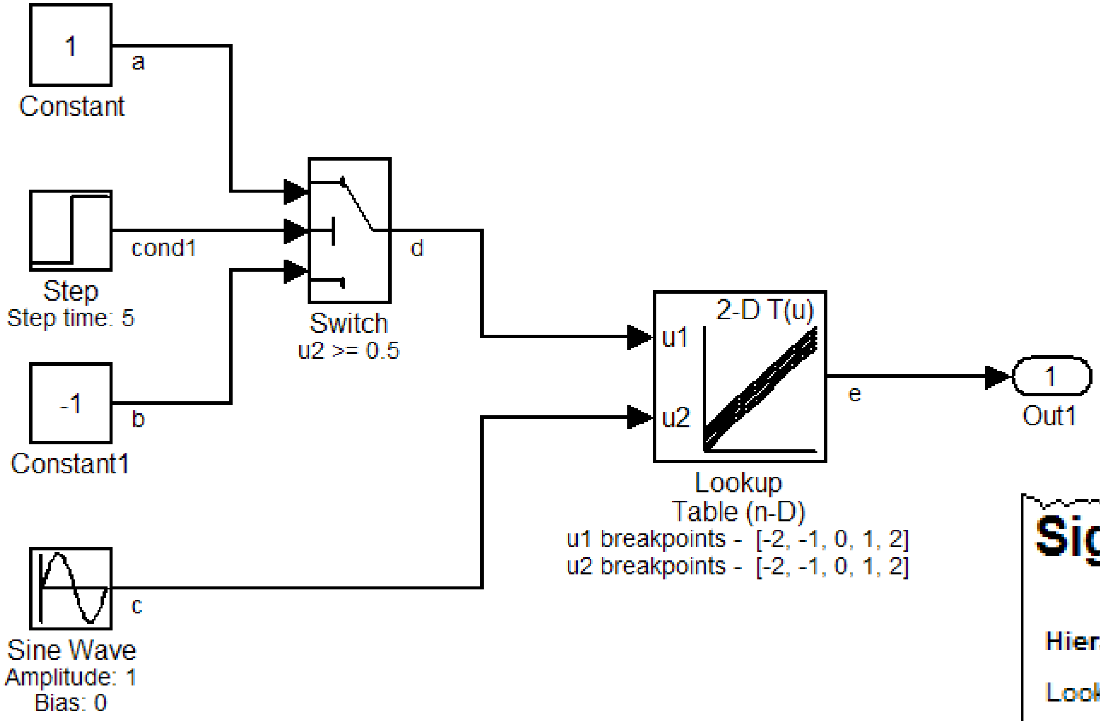
Ready | 167% | FixedStepDiscrete

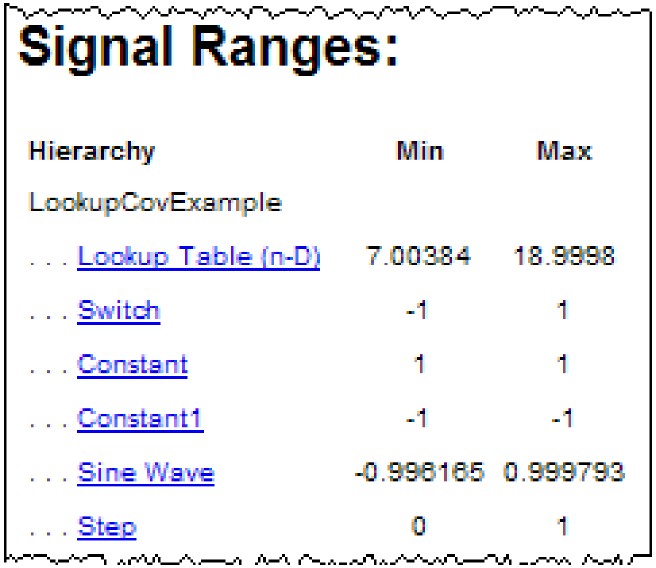
Affects (X & Y) to be T and F?

Affects (X & Y) to be T and F?

```
if (X & Y)
  Z = 1;
else
  Z = -1;
end
```

Signal Range Coverage

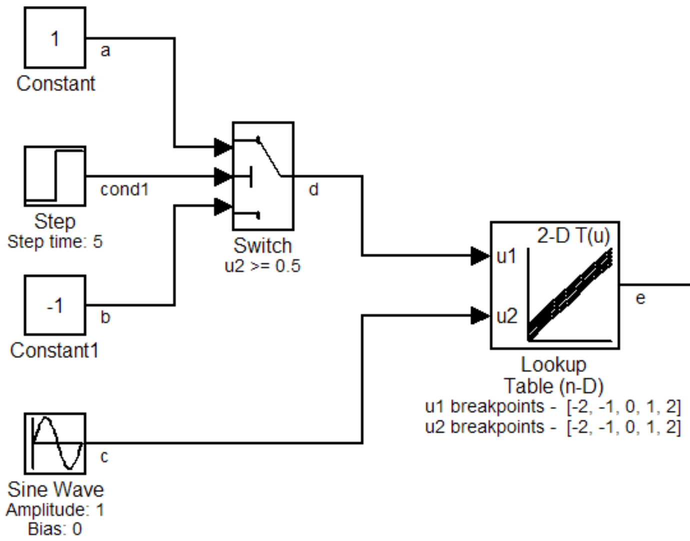




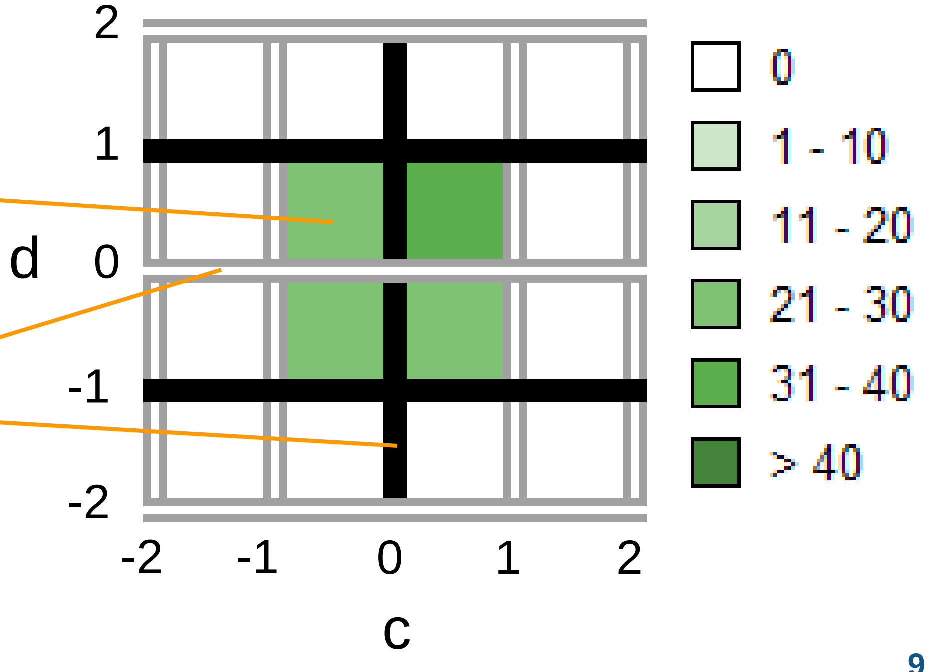
Signal Ranges:

Hierarchy	Min	Max
LookupCovExample		
. . . Lookup Table (n-D)	7.00384	18.9998
. . . Switch	-1	1
. . . Constant	1	1
. . . Constant1	-1	-1
. . . Sine Wave	-0.996165	0.999793
. . . Step	0	1

Lookup Table Coverage (LUT)



Click graph for range information



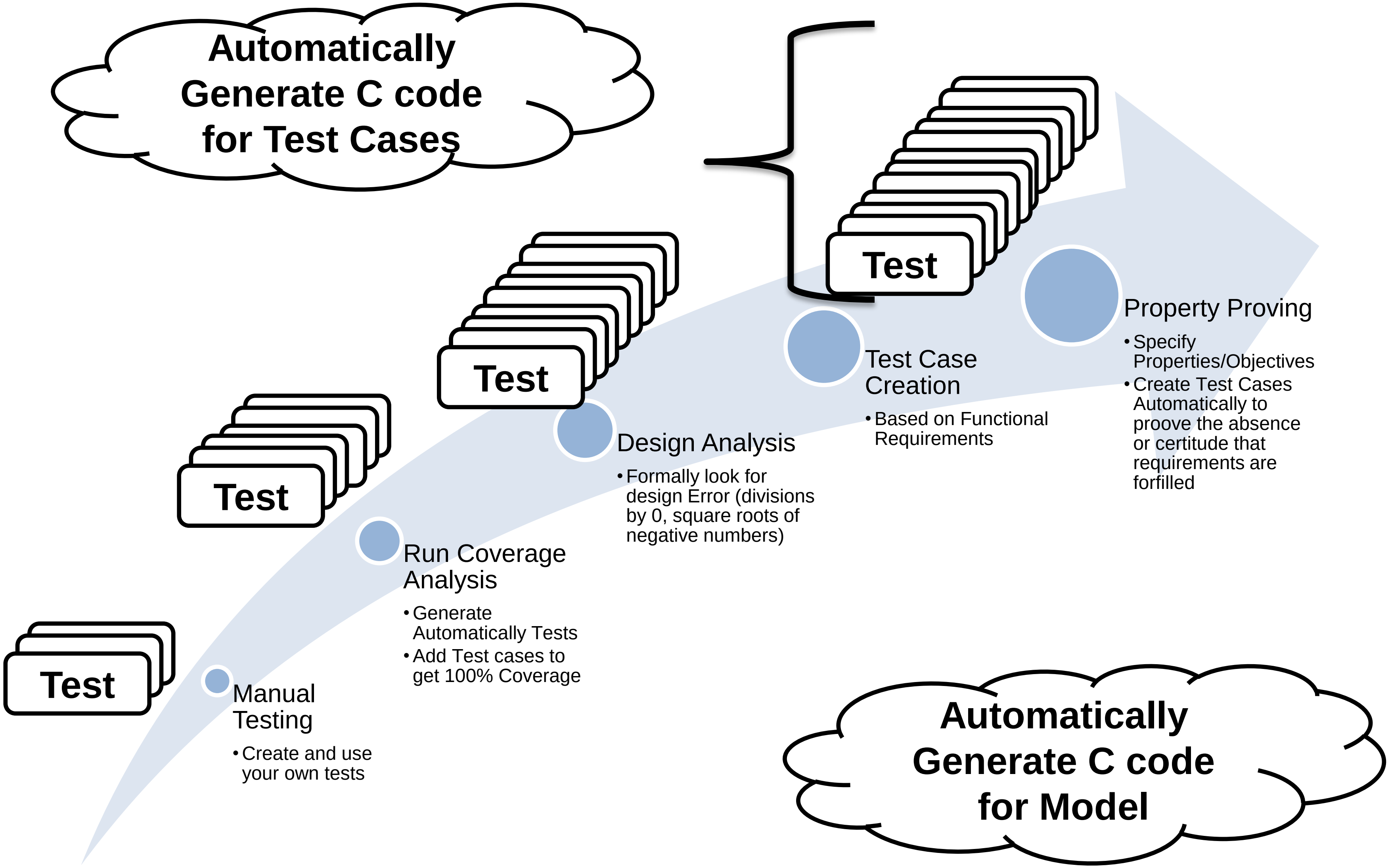
Interpolation interval

Exact values

What Does Coverage Tell You

- Useful information:
  - How much of my system did testing explore?
  - How complete are someone else’s tests?
  - How much testing has a team done on a model?
  - Is there a part of the model that is hard or impossible to reach?
  - If using code generation, what tests are needed for the final code?
  - If I know what the expected behaviour is, did I see any violations whilst achieving coverage?
- What it isn’t:
  - Coverage testing helps find unintended function, but doesn’t test for correct function on its own.
  - A good starting point, but additional tests needed for full source and object code coverage.

Verification and Validation @ Model Level



Automatically Generate C code for Test Cases

Manual Testing
- Create and use your own tests

Run Coverage Analysis
- Generate Automatically Tests
- Add Test cases to get 100% Coverage

Design Analysis
- Formally look for design Error (divisions by 0, square roots of negative numbers)

Test Case Creation
- Based on Functional Requirements

Property Proving
- Specify Properties/Objectives
- Create Test Cases Automatically to proove the absence or certitude that requirements are forfilled

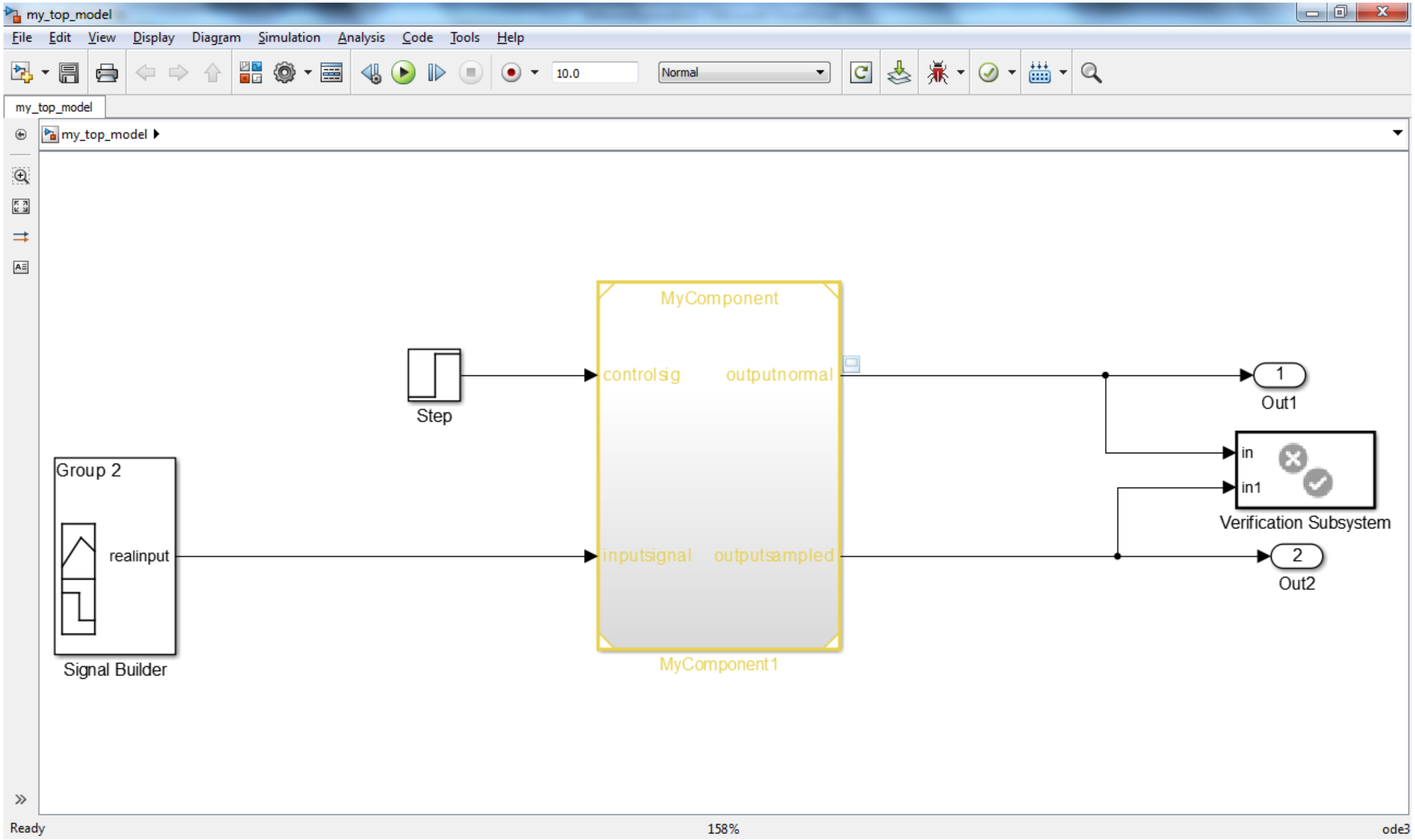
Automatically Generate C code for Model

Simple Example

- Generate Test Cases Based on Coverage
- Generate Counter Examples for division by zero
- Generate Test Cases based on Conditions and Objectives
- Generate Counter Examples Test Cases Based on Properties

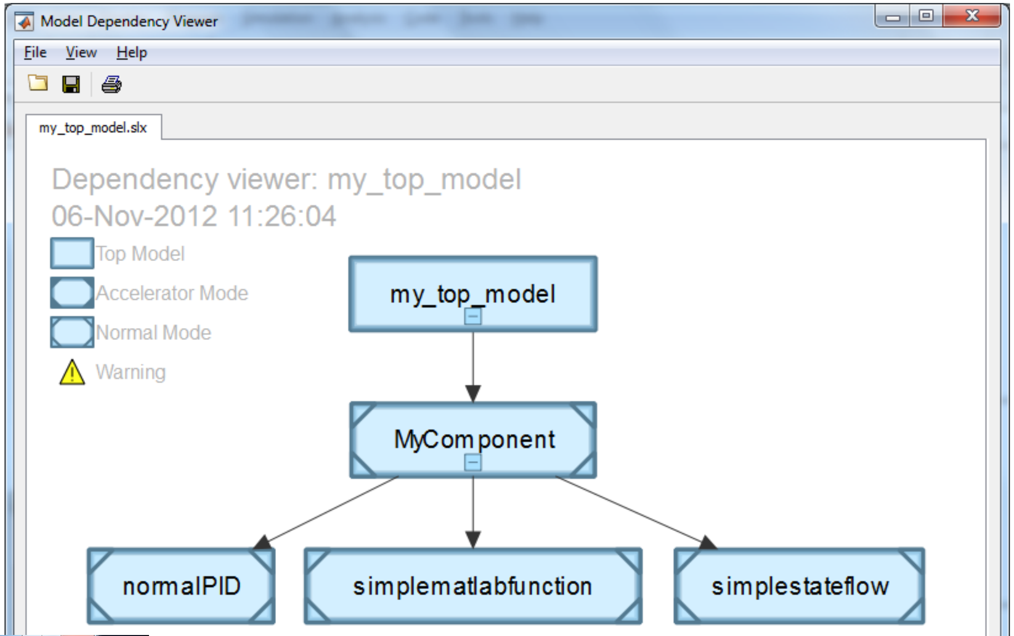
DEMO

System Modeling with Simulink

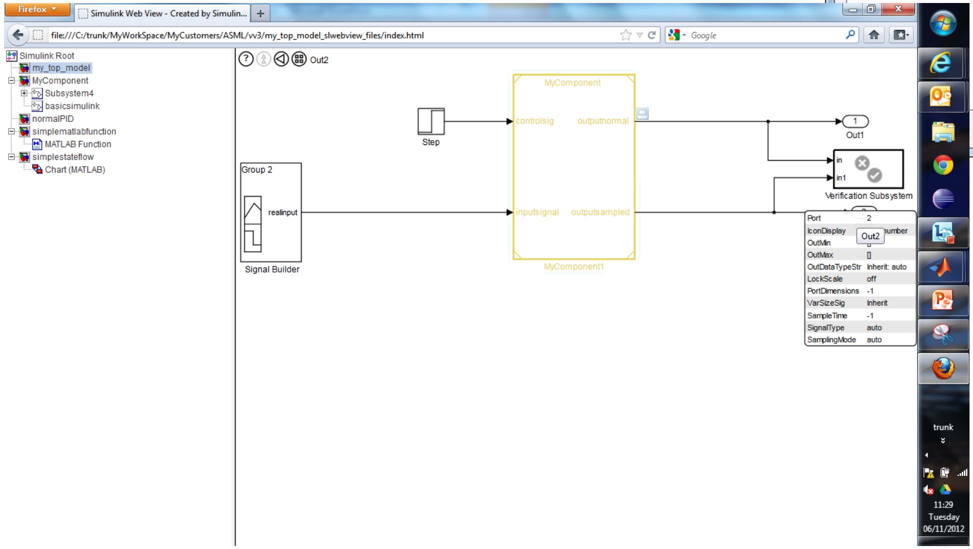


System Modeling with Simulink

- See Dependencies
- Share Model via HTML
- Generate Reports Automatically



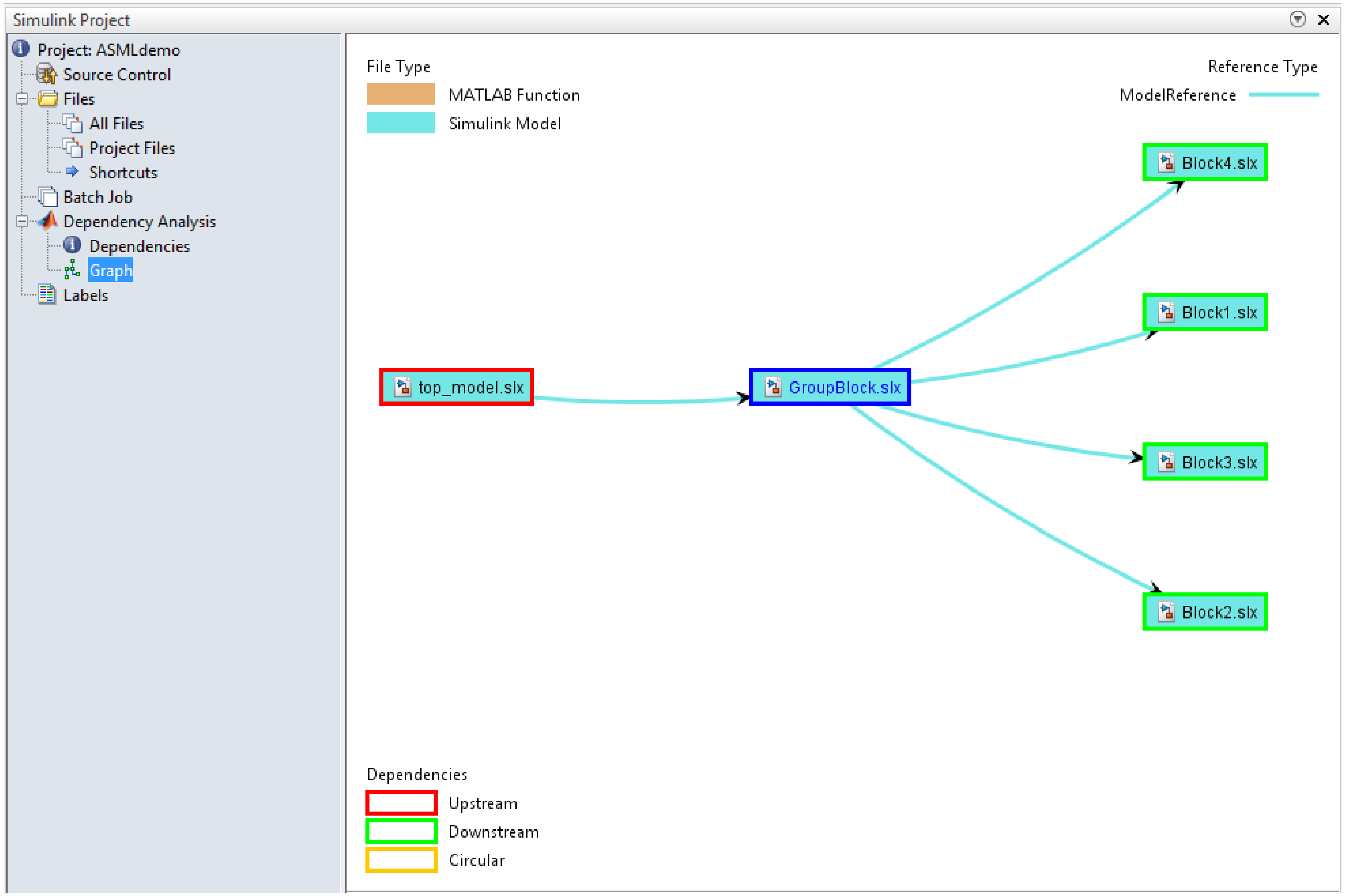
Dependency viewer: my_top_model
06-Nov-2012 11:26:04



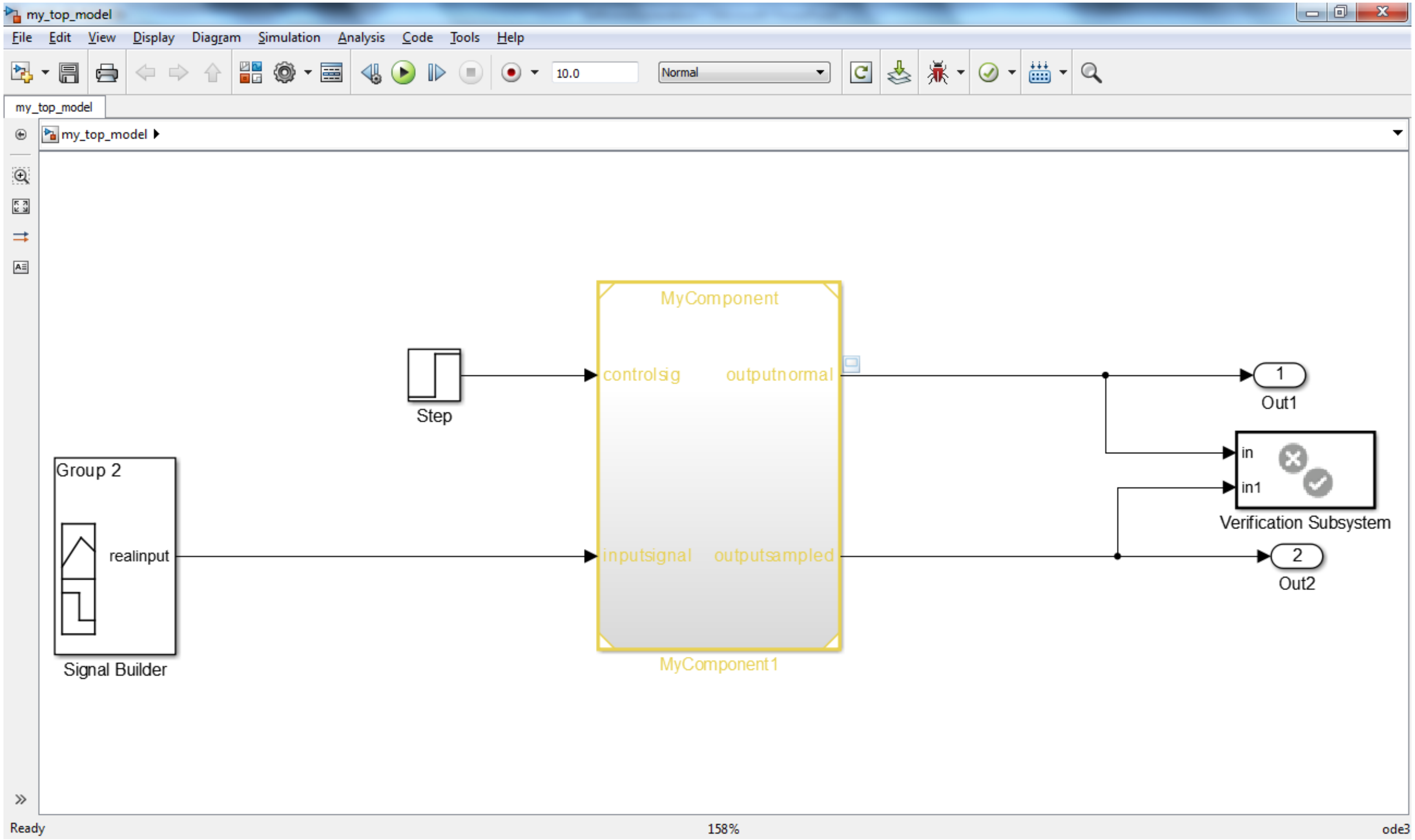
Port	2
IconDisplay	number
OutMin	[]
OutMax	[]
OutDataTypeStr	Inherit: auto
LockScale	off
PortDimensions	-1
VarSizeSig	Inherit
SampleTime	-1
SignalType	auto
SamplingMode	auto

Workflow in Simulink

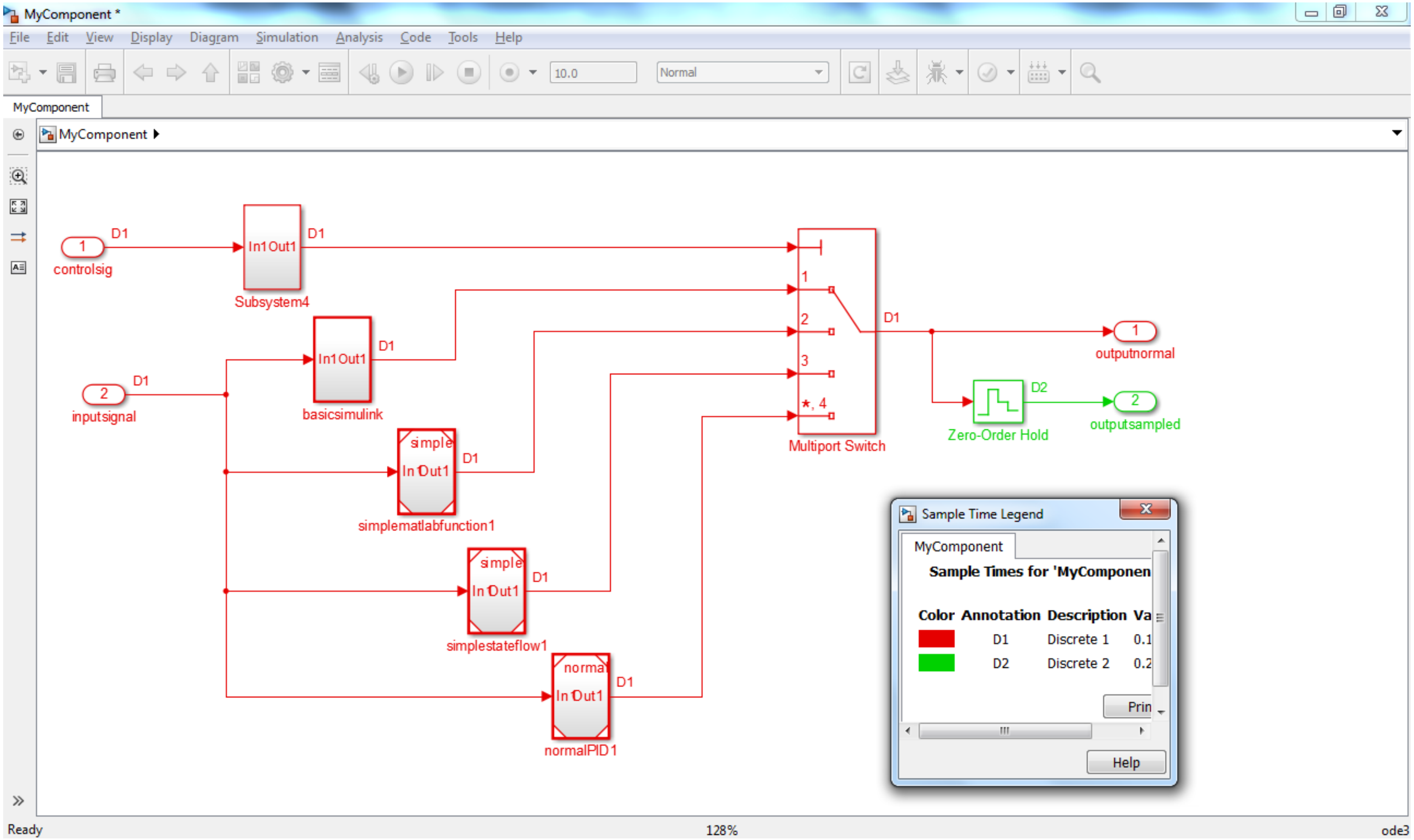
Integrates into Version Control



System Modeling with Simulink



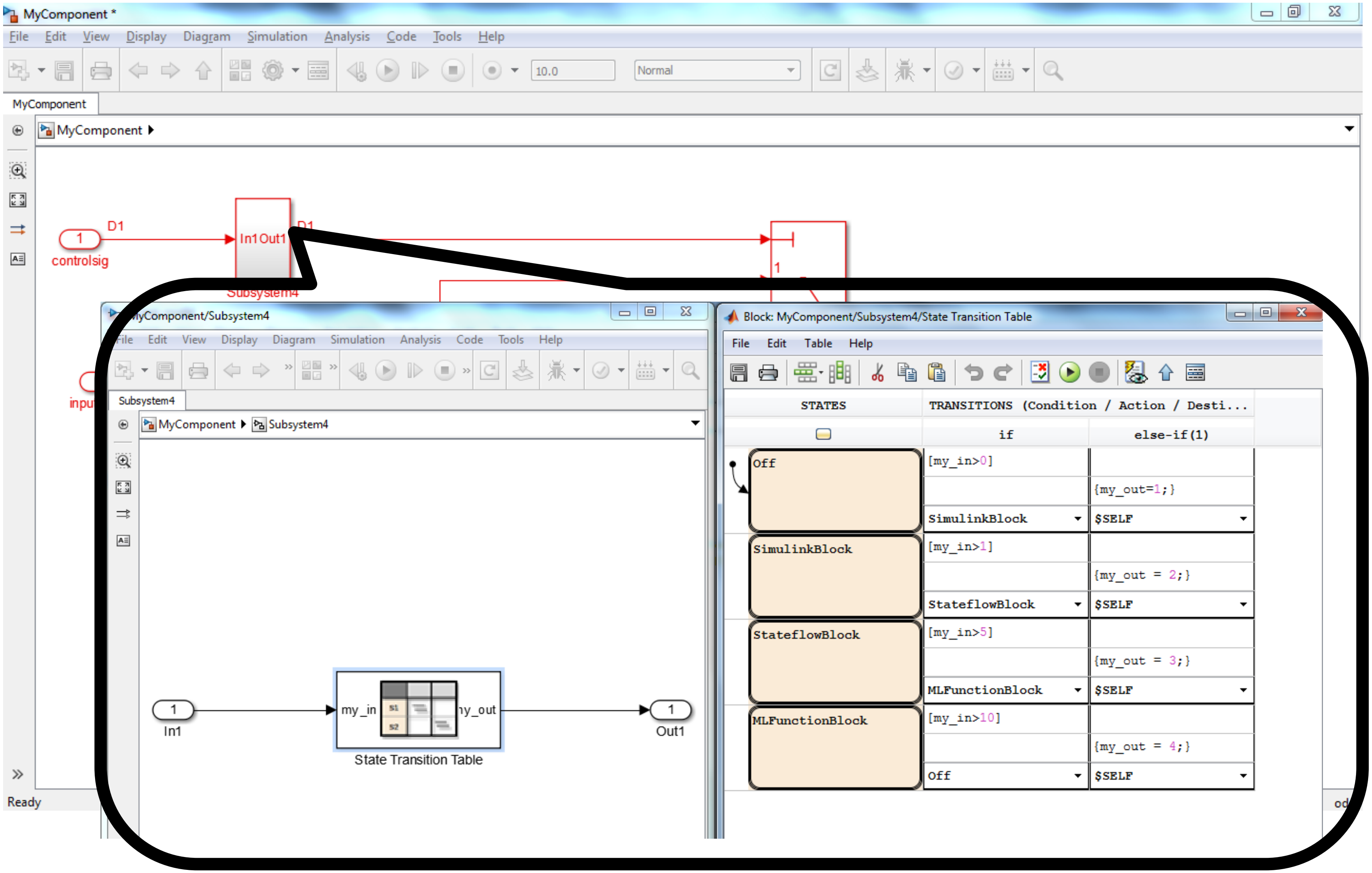
System Modeling with Simulink
MultiRate



Sample Times for 'MyComponen

Color	Annotation	Description	Va
🟥	D1	Discrete 1	0.1
🟩	D2	Discrete 2	0.2

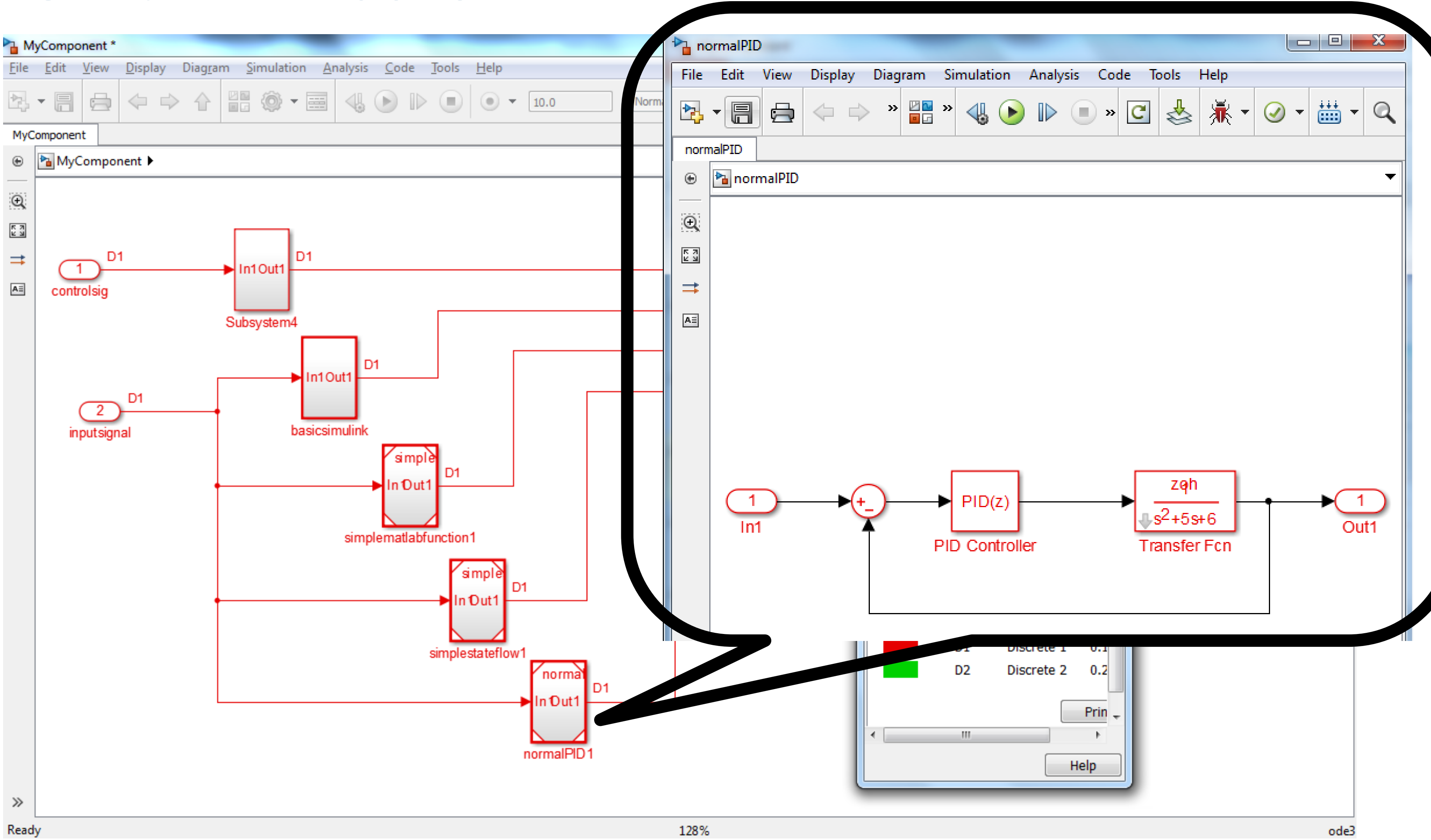
System Modeling with Simulink
State Transition Tables



STATES	TRANSITIONS (Condition / Action / Desti...	
	if	else-if(1)
Off	[my_in>0]	
		{my_out=1;}
	SimulinkBlock	$SELF
SimulinkBlock	[my_in>1]	
		{my_out = 2;}
	StateflowBlock	$SELF
StateflowBlock	[my_in>5]	
		{my_out = 3;}
	MLFunctionBlock	$SELF
MLFunctionBlock	[my_in>10]	
		{my_out = 4;}
	Off	$SELF

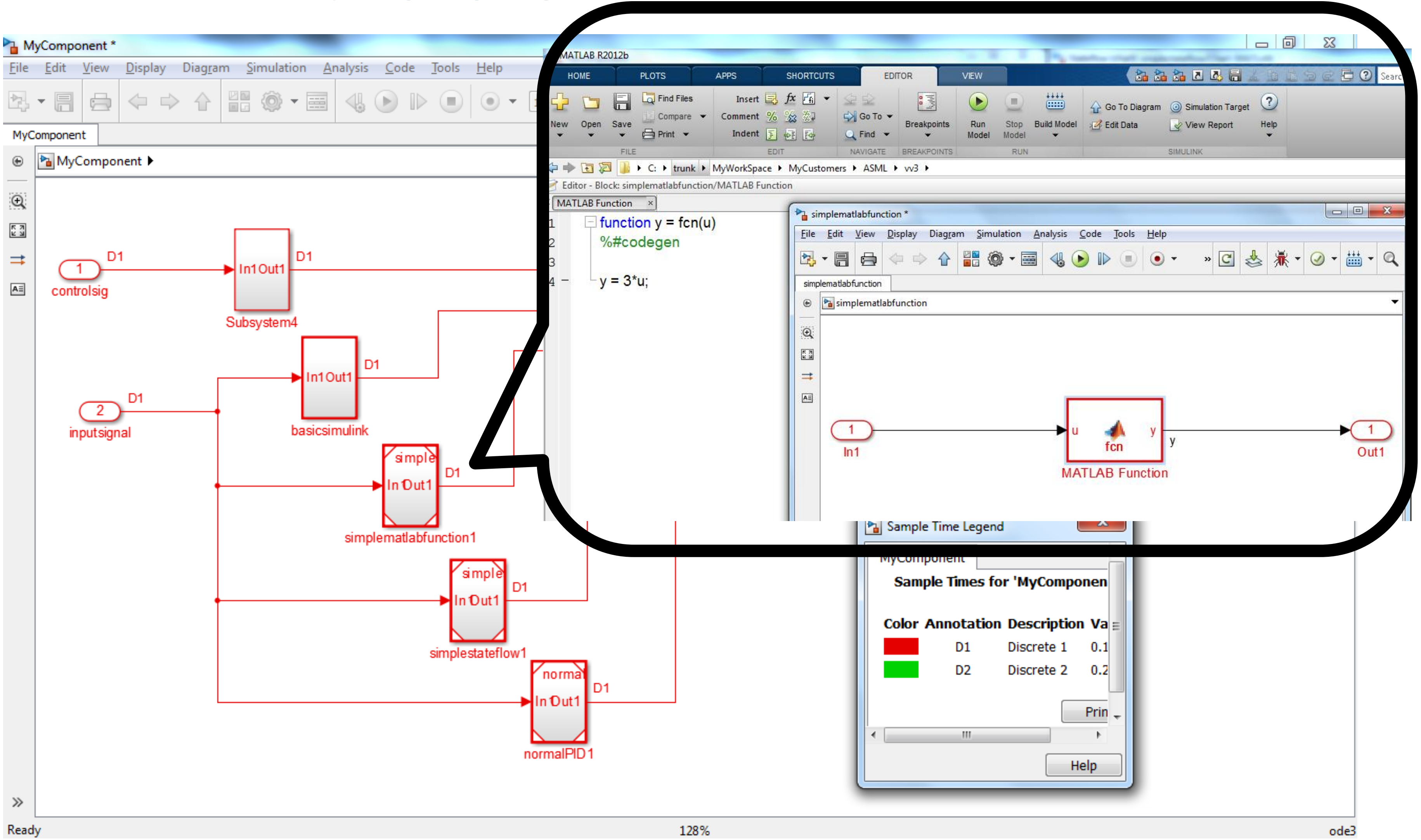
System Modeling with Simulink

Simulink Blocks



System Modeling with Simulink

MATLAB Functions



MyComponent *
File Edit View Display Diagram Simulation Analysis Code Tools Help
MyComponent
1
controlsig
D1
In1 Out1
Subsystem4
2
inputsignal
basicsimulink
simple
simplematlabfunction1
simplestateflow1
normal
normalPID1
Ready
128%
ode3
MATLAB R2012b
HOME PLOTS APPS SHORTCUTS EDITOR VIEW
New Open Save Find Files Compare Print
Insert Comment Indent
Go To Find
Breakpoints
Run Model Stop Model Build Model
Go To Diagram Edit Data Simulation Target View Report Help
FILE EDIT NAVIGATE BREAKPOINTS RUN SIMULINK
C: trunk MyWorkSpace MyCustomers ASML vv3
Editor - Block: simplematlabfunction/MATLAB Function
MATLAB Function
simplematlabfunction *
File Edit View Display Diagram Simulation Analysis Code Tools Help
simplematlabfunction
In1
u
fcn
y
MATLAB Function
Out1
Sample Time Legend
MyComponent
Print
Help

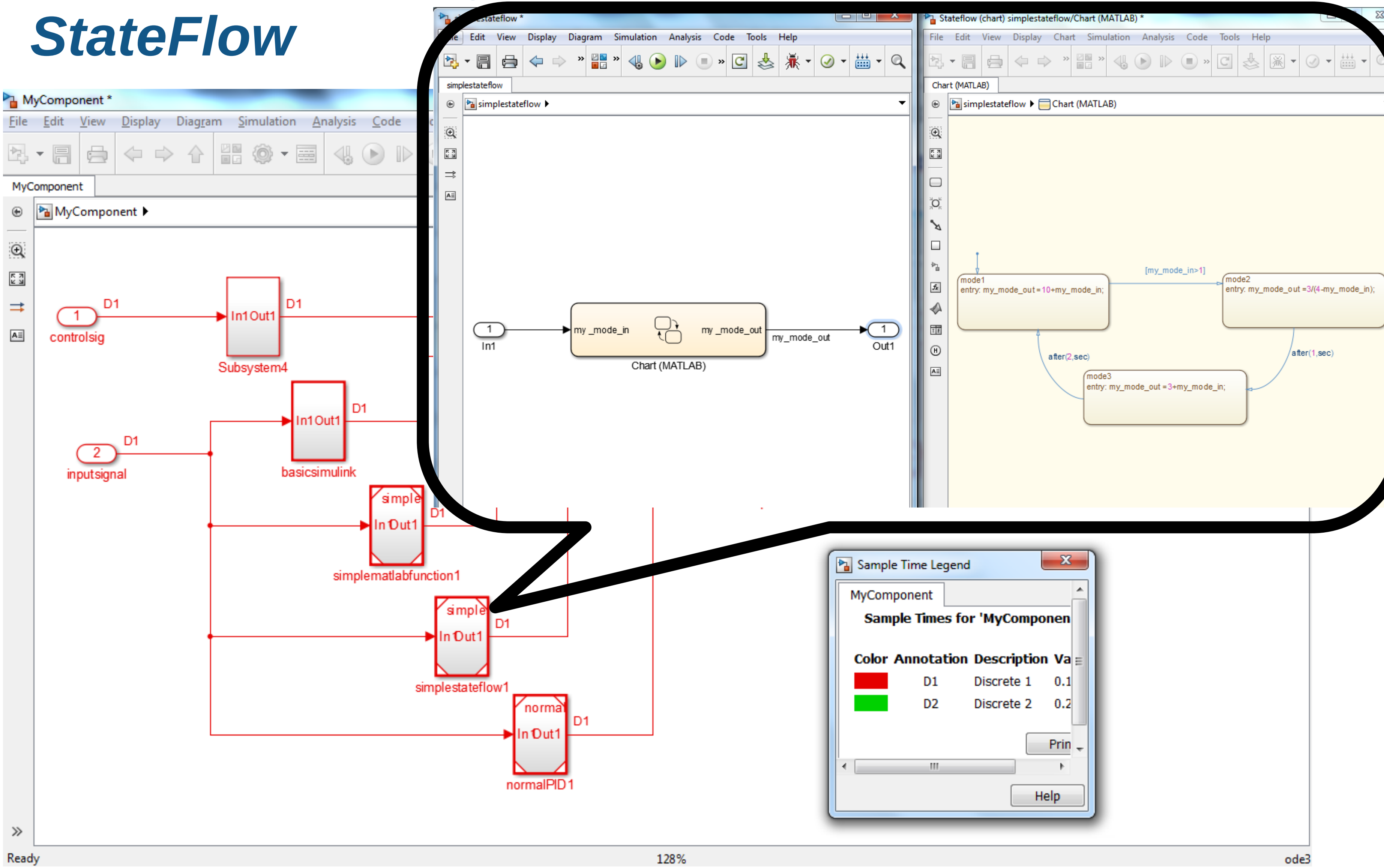
```
function y = fcn(u)
%#codegen

y = 3*u;
```

Sample Times for 'MyComponen

Color	Annotation	Description	Va
🟥	D1	Discrete 1	0.1
🟩	D2	Discrete 2	0.2

System Modeling with Simulink

StateFlow



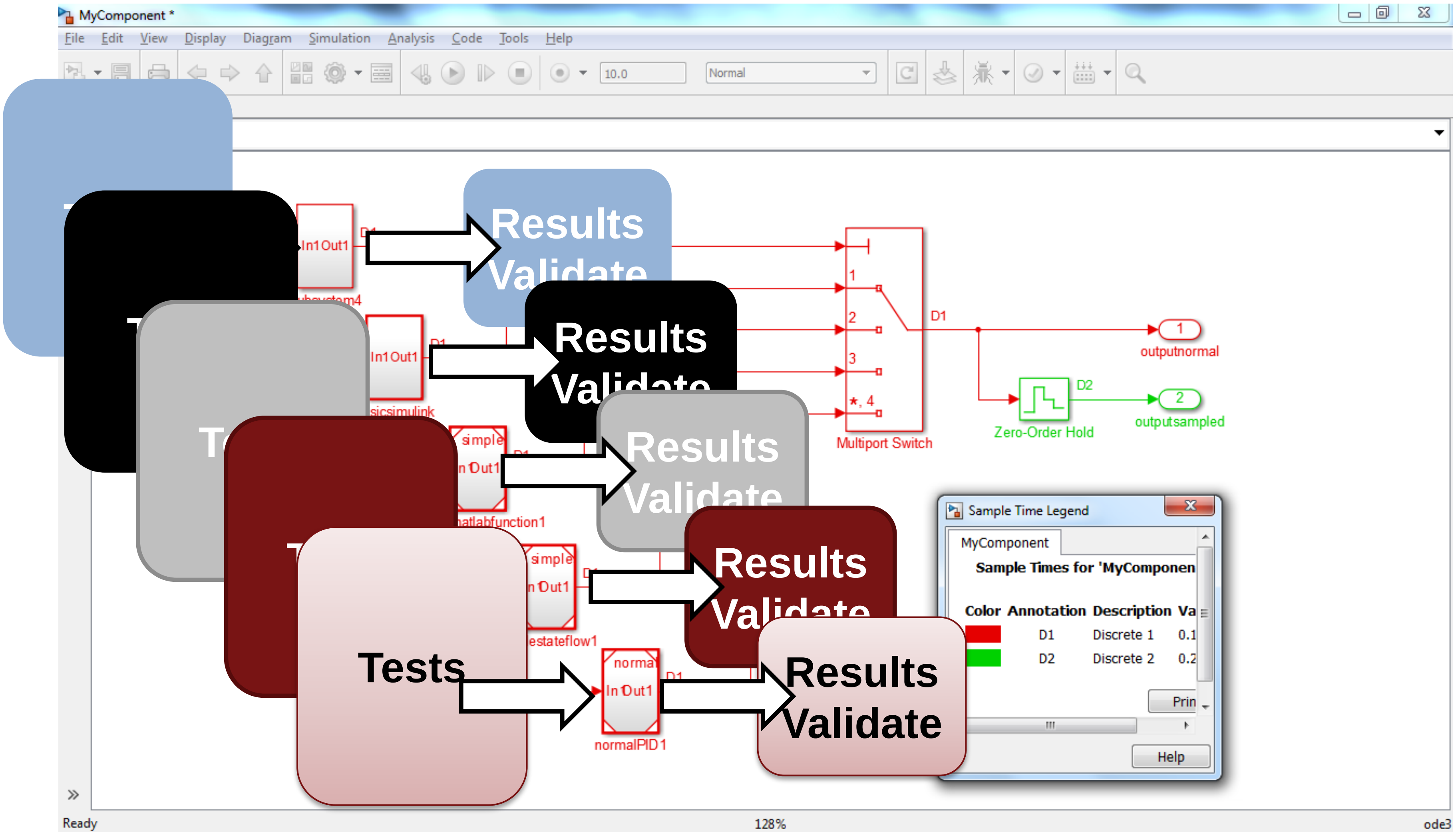
MyComponent *
File Edit View Display Diagram Simulation Analysis Code
MyComponent
controlsig
D1
In1 Out1
Subsystem4
inputsignal
basicsimulink
simple
simplematlabfunction1
simplestateflow1
norma
normalPID1
Ready
128%
ode3
simplestateflow *
File Edit View Display Diagram Simulation Analysis Code Tools Help
simplestateflow
In1
my_mode_in
my_mode_out
Chart (MATLAB)
Out1
Stateflow (chart) simplestateflow/Chart (MATLAB) *
File Edit View Display Chart Simulation Analysis Code Tools Help
Chart (MATLAB)
mode1
entry: my_mode_out=10+my_mode_in;
[my_mode_in>1]
mode2
entry: my_mode_out=3/(4-my_mode_in);
after(2,sec)
after(1,sec)
mode3
entry: my_mode_out=3+my_mode_in;
Sample Time Legend
MyComponent
Prin
Help

Sample Times for 'MyComponen

Color	Annotation	Description	Va
🟥	D1	Discrete 1	0.1
🟩	D2	Discrete 2	0.2

System Modeling with Simulink

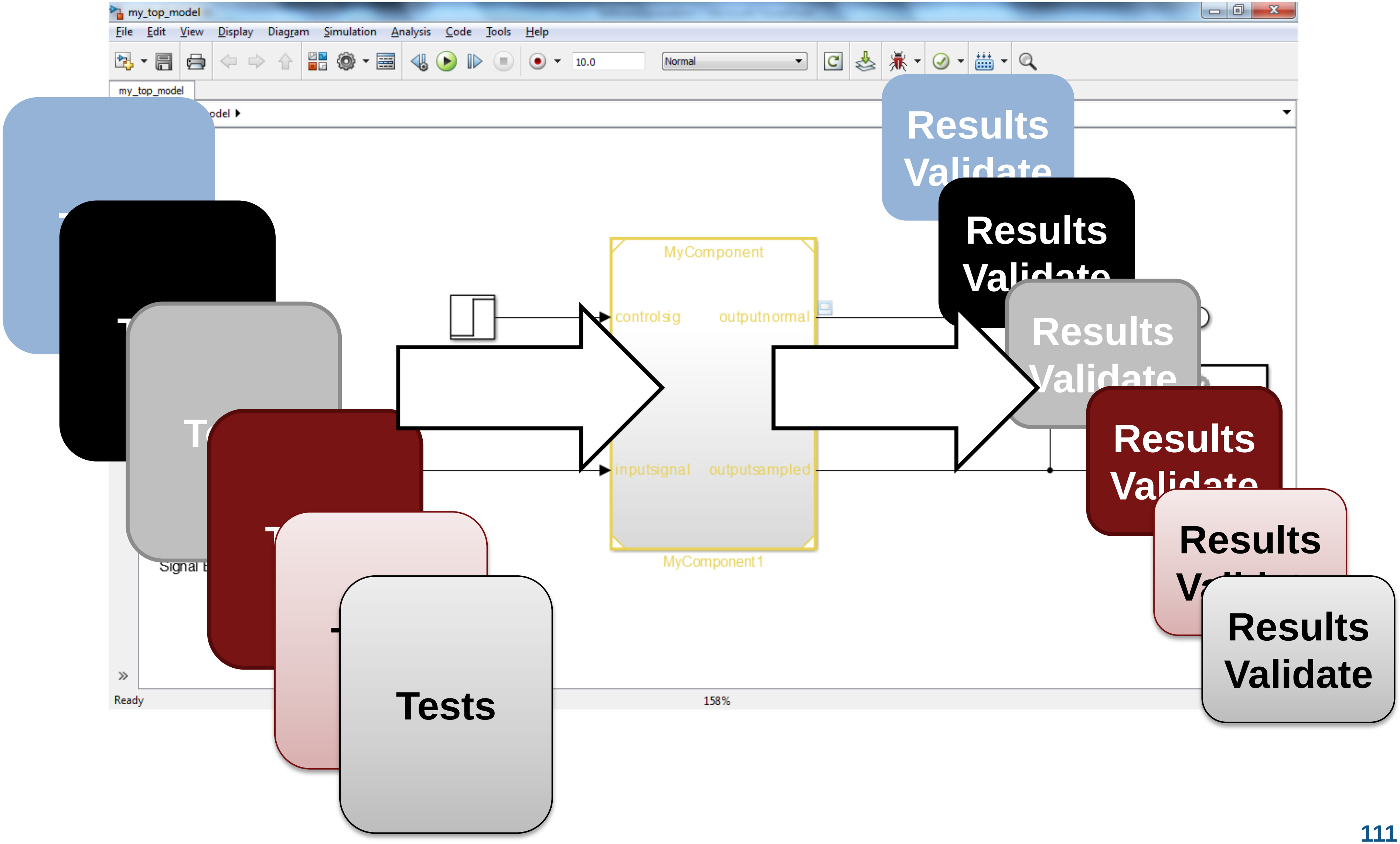
MultiRate



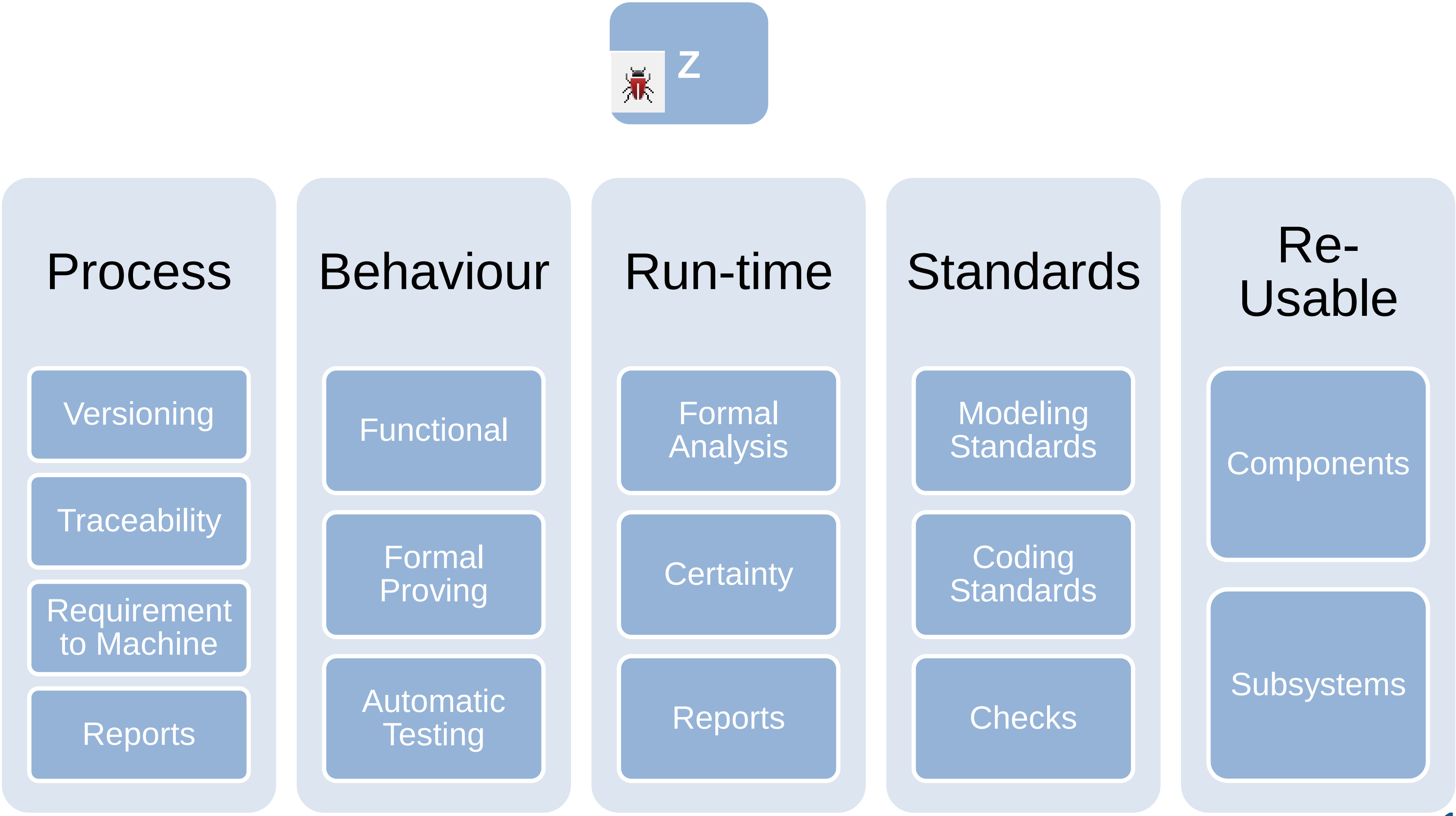
Sample Times for 'MyComponen

Color	Annotation	Description	Va
🟥	D1	Discrete 1	0.1
🟩	D2	Discrete 2	0.2

System Modeling with Simulink



How do I know how "good" Z is? What to do when there is a problem?



Z

Process
- Versioning
- Traceability
- Requirement to Machine
- Reports

Behaviour
- Functional
- Formal Proving
- Automatic Testing

Run-time
- Formal Analysis
- Certainty
- Reports

Standards
- Modeling Standards
- Coding Standards
- Checks

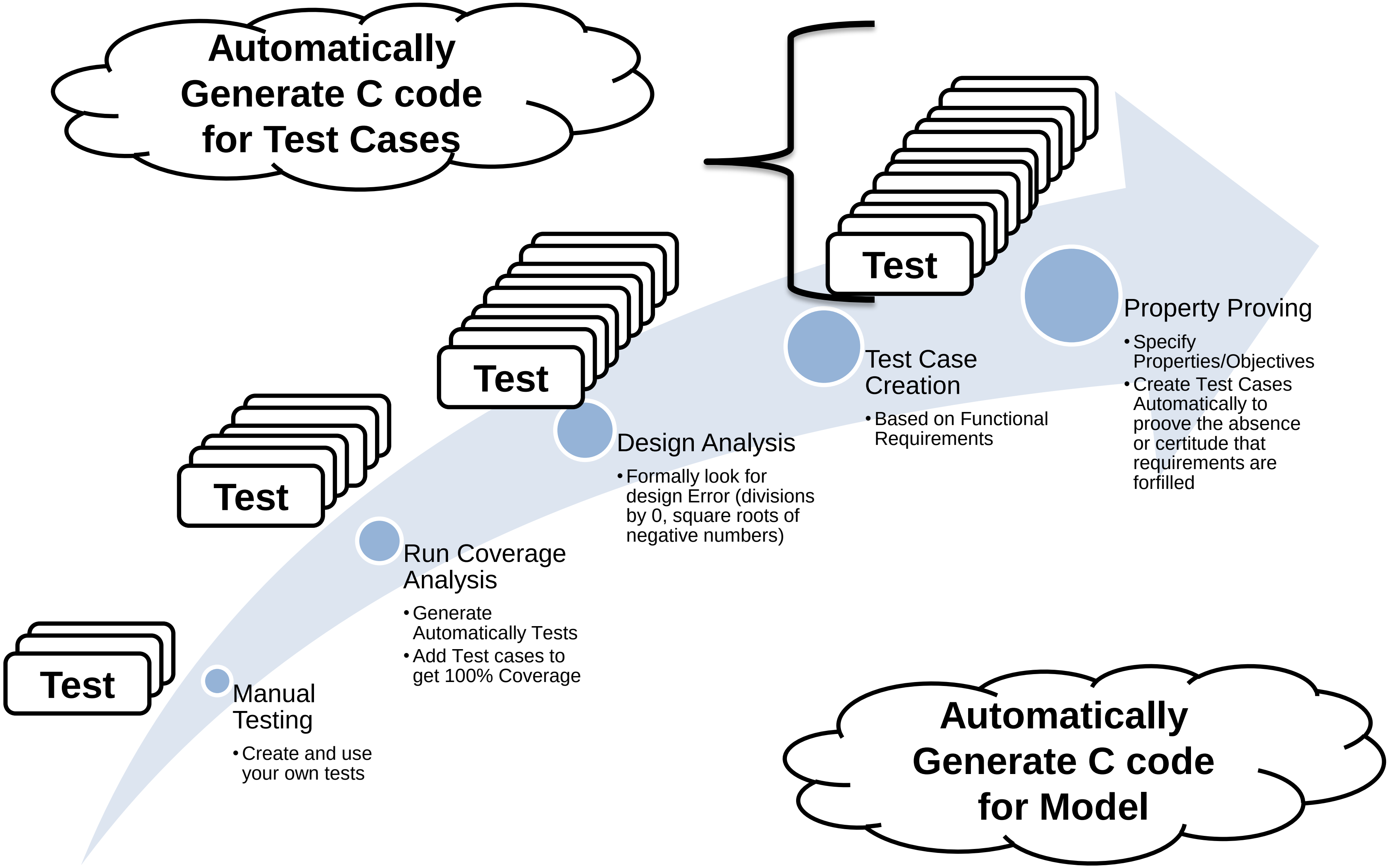
Re-Usable
- Components
- Subsystems

How do I know how “good” Z is? What to do when there is a problem?

Z

Process	Behaviour	Run-time	Standards	Re-Usability
Versioning	Functional	Formal Analysis	Modeling Standards	Components
Traceability	Formal Proving	Certainty	Coding Standards	Subsystems
Requirement to Machine	Automatic Testing	Fault Detection, Isolation and Recovery	Checks	
Reports	Scheduling	Reports		

Verification and Validation @ Model Level



Automatically Generate C code for Test Cases

Manual Testing

- Create and use your own tests

Run Coverage Analysis

- Generate Automatically Tests
- Add Test cases to get 100% Coverage

Design Analysis

- Formally look for design Error (divisions by 0, square roots of negative numbers)

Test Case Creation

- Based on Functional Requirements

Property Proving

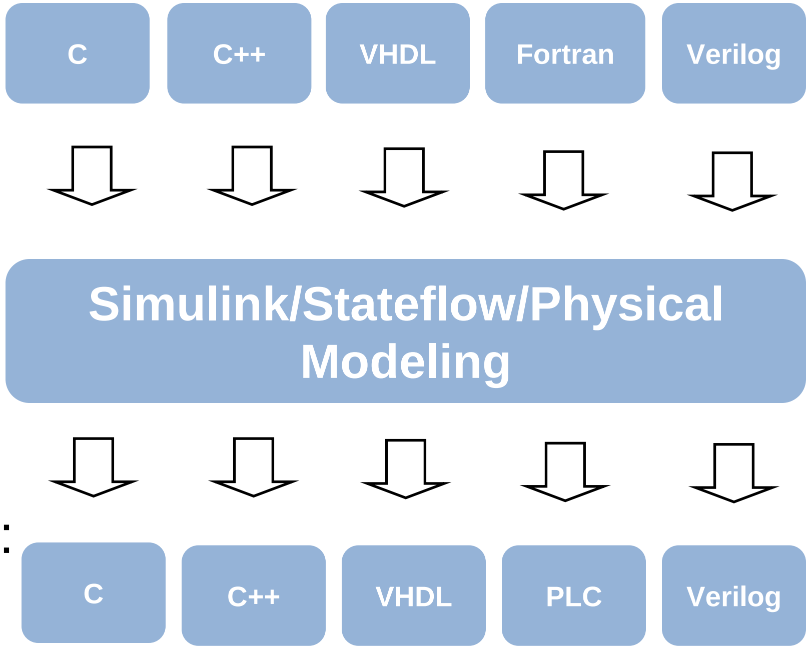
- Specify Properties/Objectives
- Create Test Cases Automatically to proove the absence or certitude that requirements are forfilled

Automatically Generate C code for Model

Model Transformation is Key

Model Transformation is Key

- Automatic Code Generation
  - From Simulink to:
    - C
    - C++
    - Structured Text
    - Verilog
    - VHDL
  - From C/C++/ADA/Verilog/VHDL to:
    - Simulink

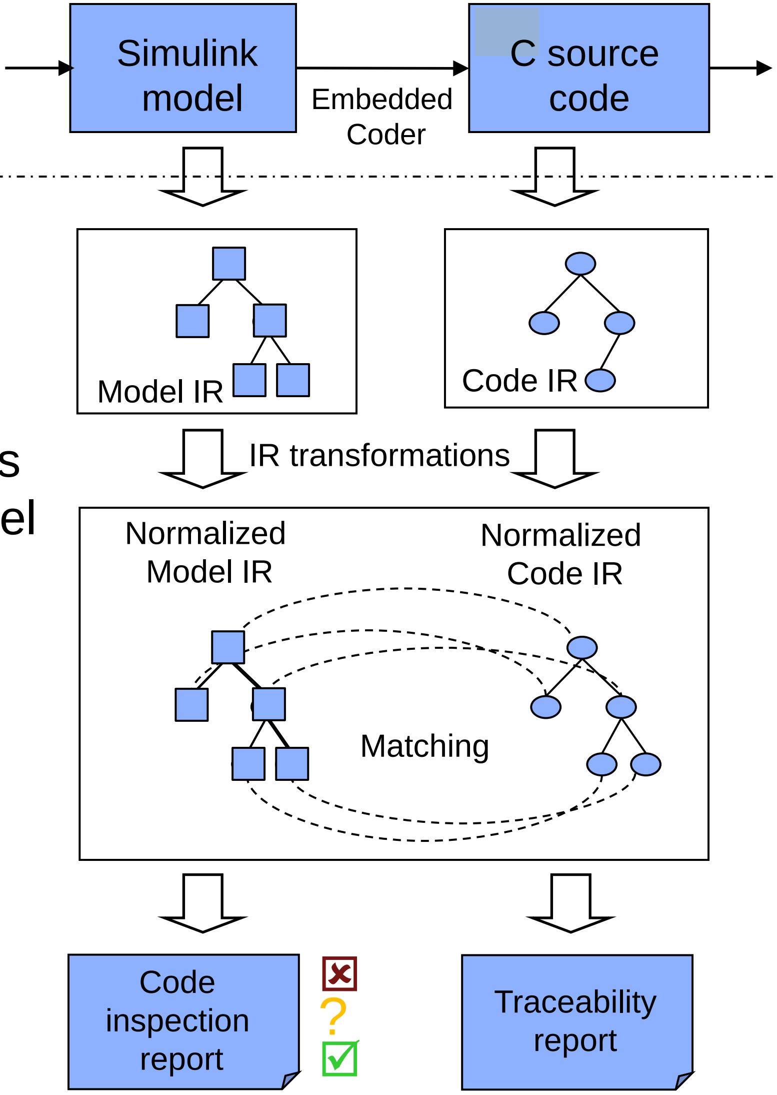


Simulink Code Inspector

Model and code development

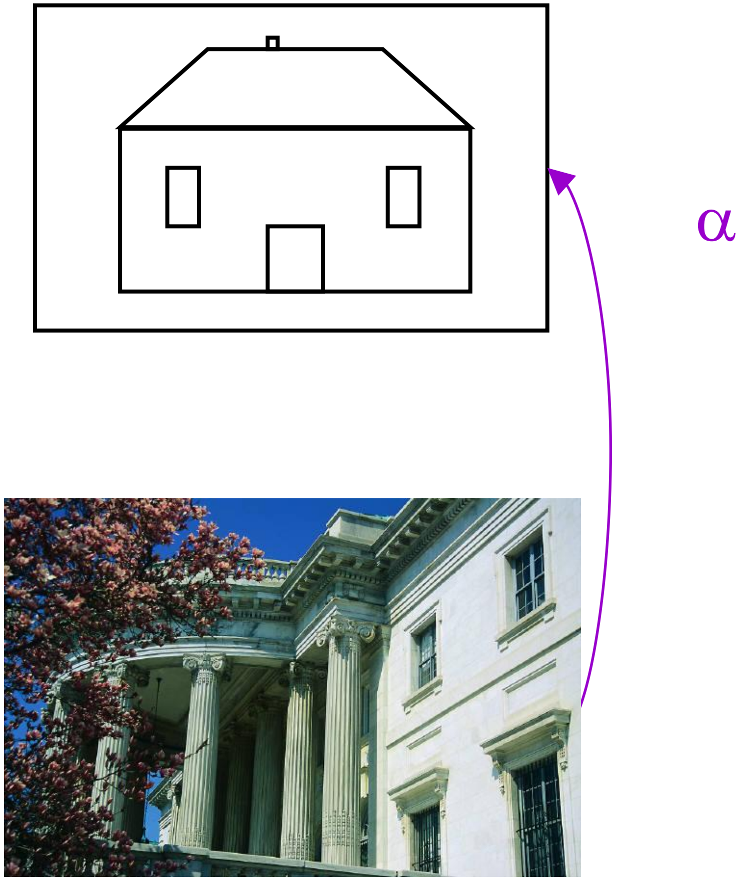
Independent code inspection

- Static verification tool that checks the generated code against model
- Automates DO-178B Table A-5 verification activities
- Technology allows seamless upgrades to new releases

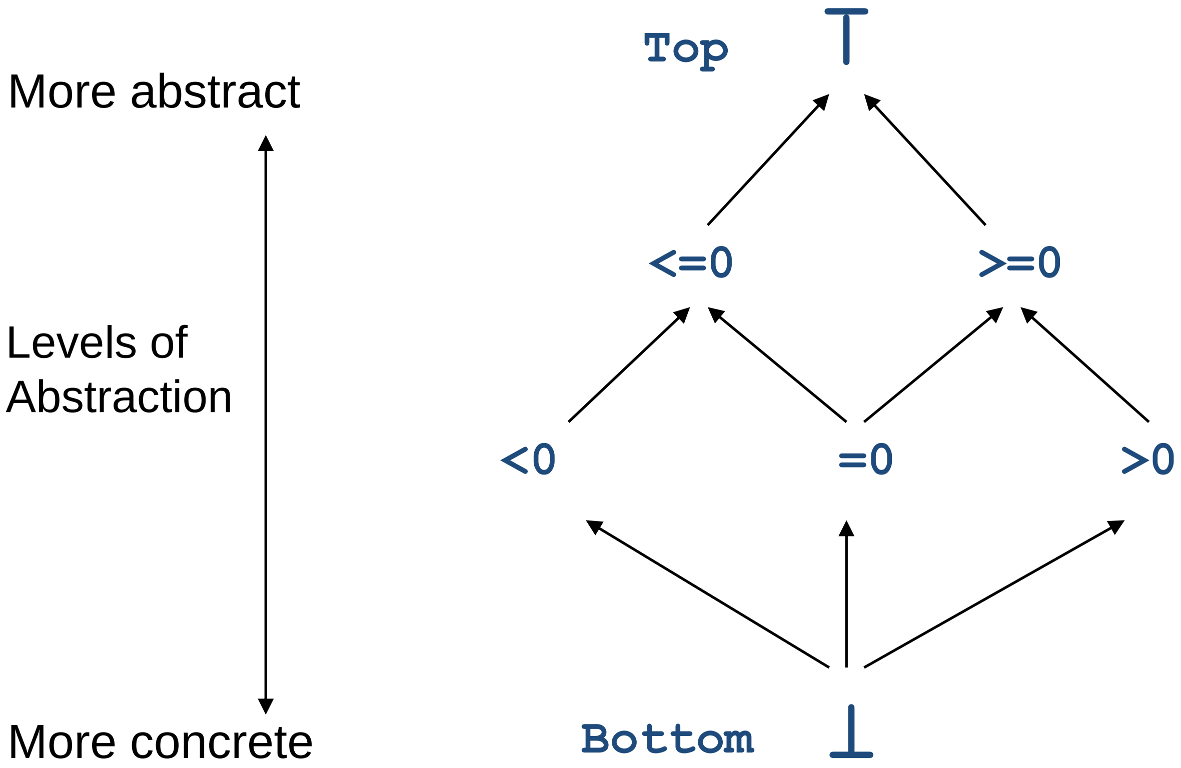


IR: Intermediate representation

How can you prove that no error occurred?
What is Abstract Interpretation?



Example of lattice for variables values: Signs



More abstract

Top ⊤

<=0 >=0

Levels of Abstraction

<0 =0 >0

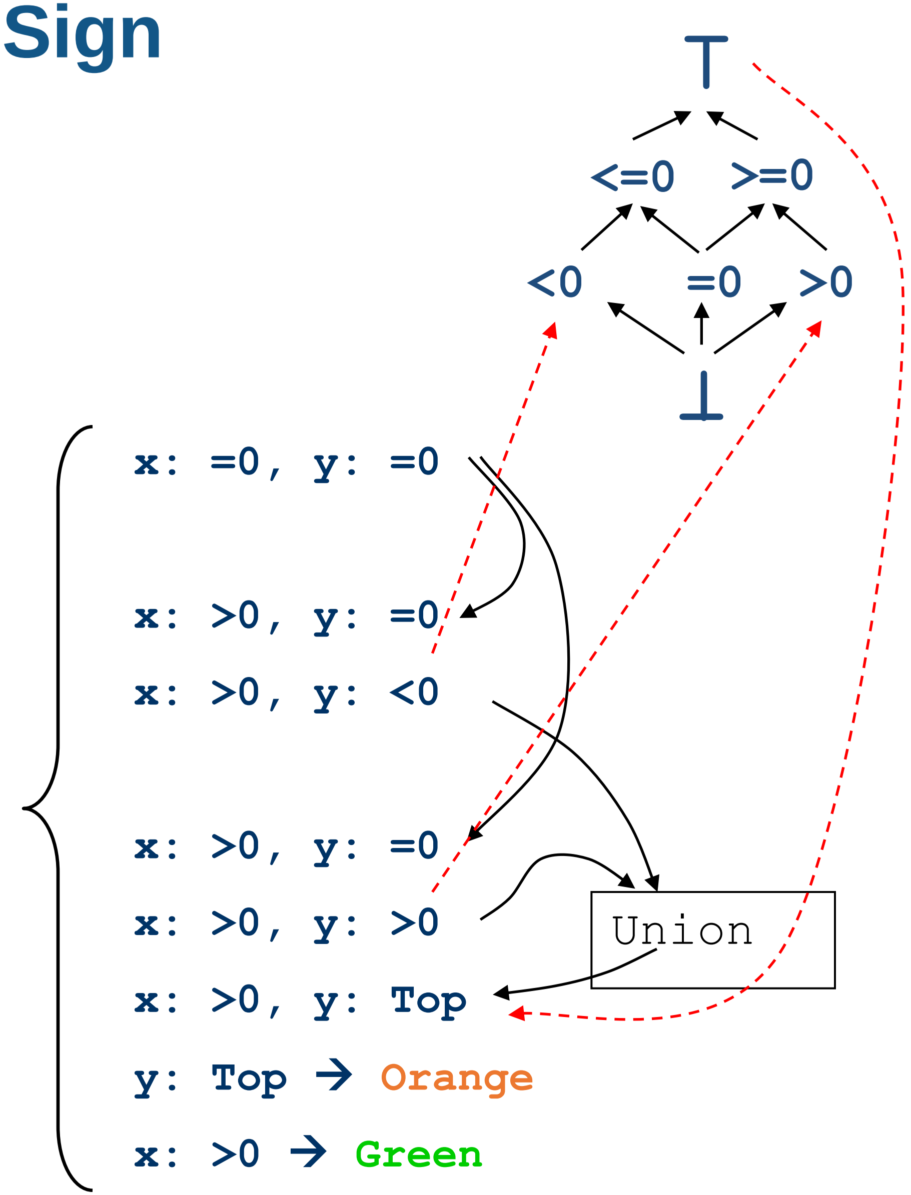
More concrete

Bottom ⊥

Example of abstraction: Sign

```
volatile int random;
int x=0, y=0;
if (random) {
  x++;
  y--;
} else {
  x += 2;
  y += 1;
}
assert(y > 0);
assert(x > 0);
```

α



x: =0, y: =0

x: >0, y: =0

x: >0, y: <0

x: >0, y: =0

x: >0, y: >0

x: >0, y: Top

y: Top → Orange

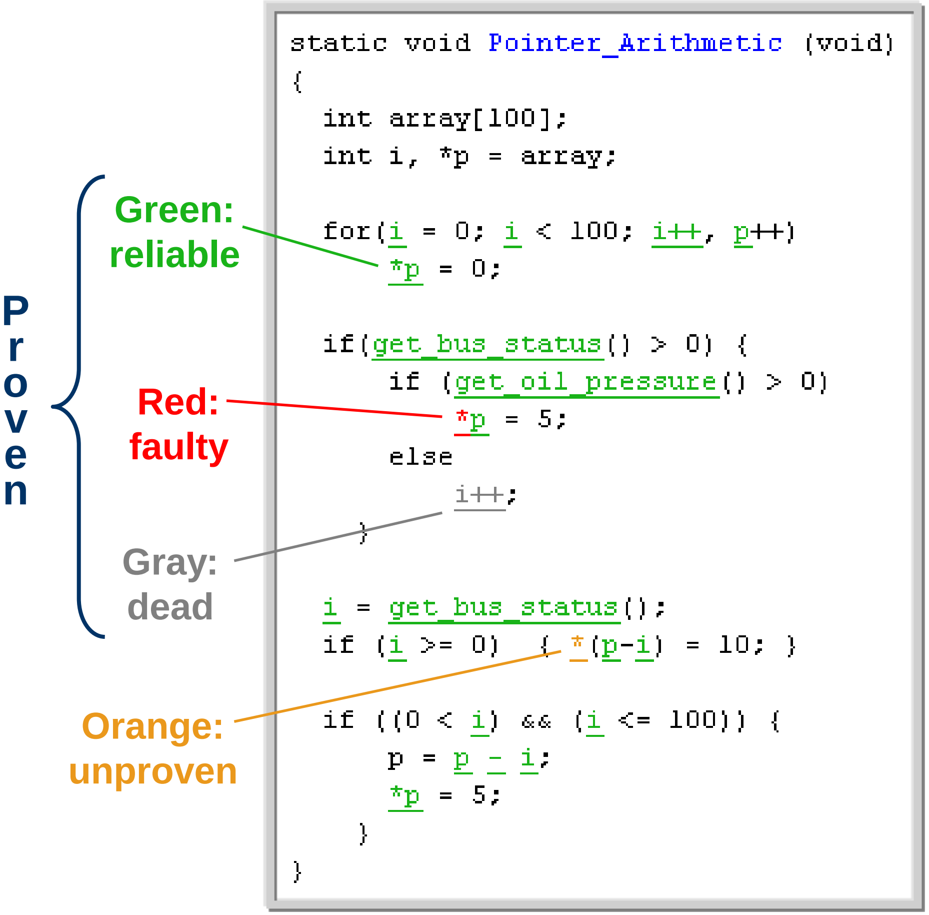
x: >0 → Green

PolySpace Products for Code Verification

- **Quality improvement**
  - Prove the absence of run-time errors in source code
  - Measure, improve, and control

- **Usage**
  - Simple colored source code
  - No compilation, no execution, no test cases
  - For C/C++ or Ada

- **Process**
  - Run early in development cycle
  - Use for automatically generated and handwritten code



```
static void Pointer_Arithmetic (void)
{
  int array[100];
  int i, *p = array;

  for(i = 0; i < 100; i++, p++)
      *p = 0;

  if(get_bus_status() > 0) {
      if (get_oil_pressure() > 0)
          *p = 5;
      else
          i++;
    }

  i = get_bus_status();
  if (i >= 0)  { *(p-i) = 10; }

  if ((0 < i) && (i <= 100)) {
      p = p - i;
      *p = 5;
    }
}
```

Proven

Green: reliable

Red: faulty

Gray: dead

Orange: unproven

How can you prove that no error occurred?

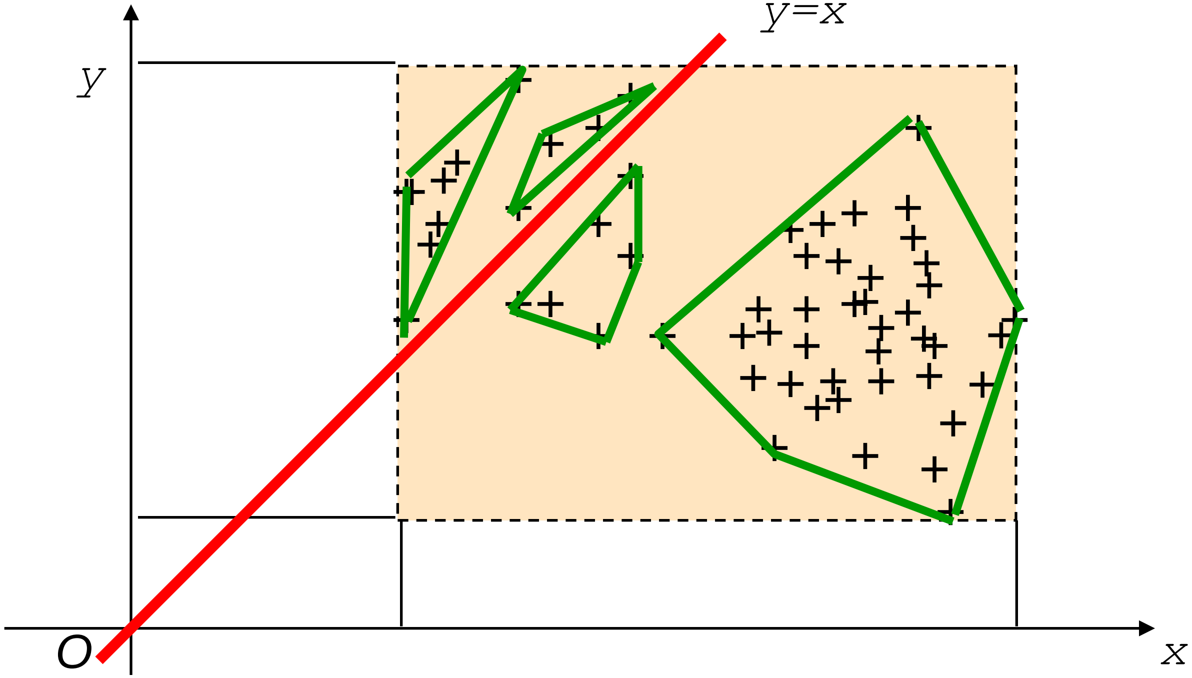
Verifying **`x = x / (x - y)`**

- Potential run-time errors:
  - Are `x` and `y` initialized?
  - Could a division by 0 occur?
  - Could there be an underflow or overflow on ‘−’, ‘/,’ or ‘=’ ?

The following slide focuses on the check for a division by 0.

No execution
No simulation
No test cases to write

Verifying `x = x // (x - y)`

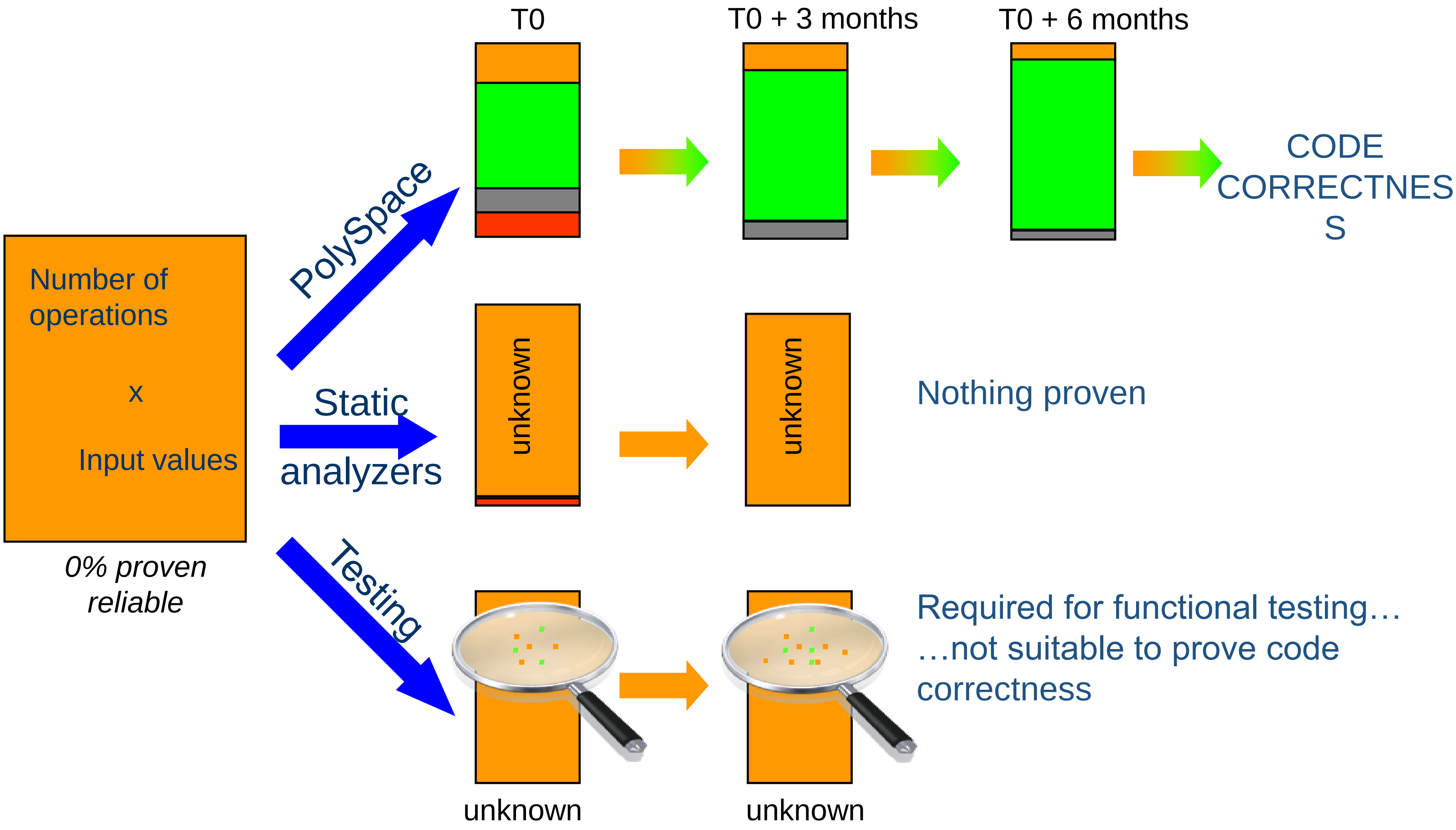


Type analysis

Abstract interpretation

What color is your code today?

How do you prove code correctness?



0% proven reliable

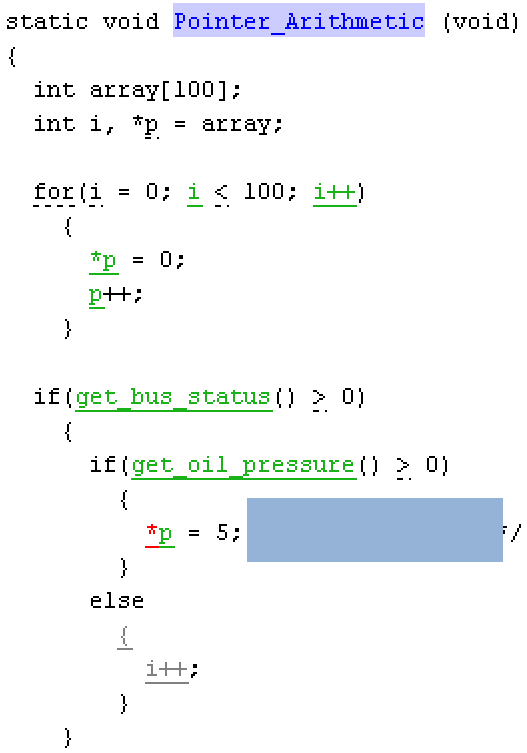
CODE CORRECTNES S

Nothing proven

Required for functional testing…
…not suitable to prove code correctness

Challenge…

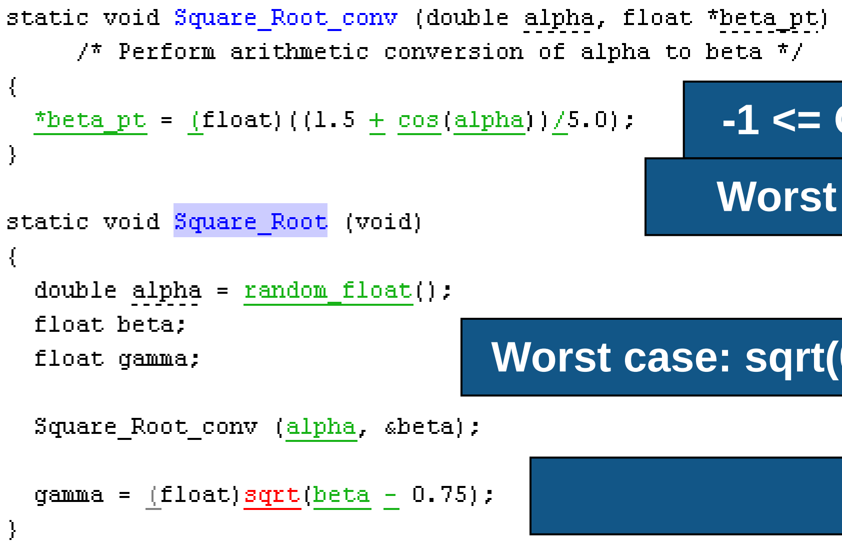
- Why is there red code here?



```
static void Pointer_Arithmetic (void)
{
  int array[100];
  int i, *p = array;

  for(i = 0; i < 100; i++)
    {
      *p = 0;
      p++;
    }

  if(get_bus_status() > 0)
    {
      if(get_oil_pressure() > 0)
        {
          *p = 5; [redacted]
        }
      else
        {
          i++;
        }
    }
```

Challenge …

- Why is there red code here?



```
static void Square_Root_conv (double alpha, float *beta_pt)
     /* Perform arithmetic conversion of alpha to beta */
{
  *beta_pt = (float)((1.5 + cos(alpha))/5.0);
}

static void Square_Root (void)
{
  double alpha = random_float();
  float beta;
  float gamma;

  Square_Root_conv (alpha, &beta);

  gamma = (float)sqrt(beta - 0.75);
}
```

-1 <= Cos(alpha) <= 1

Worst case: 0.5/5 = 0.1

Worst case: sqrt(0.1 - 0.75) is a run time Error

Takeaways

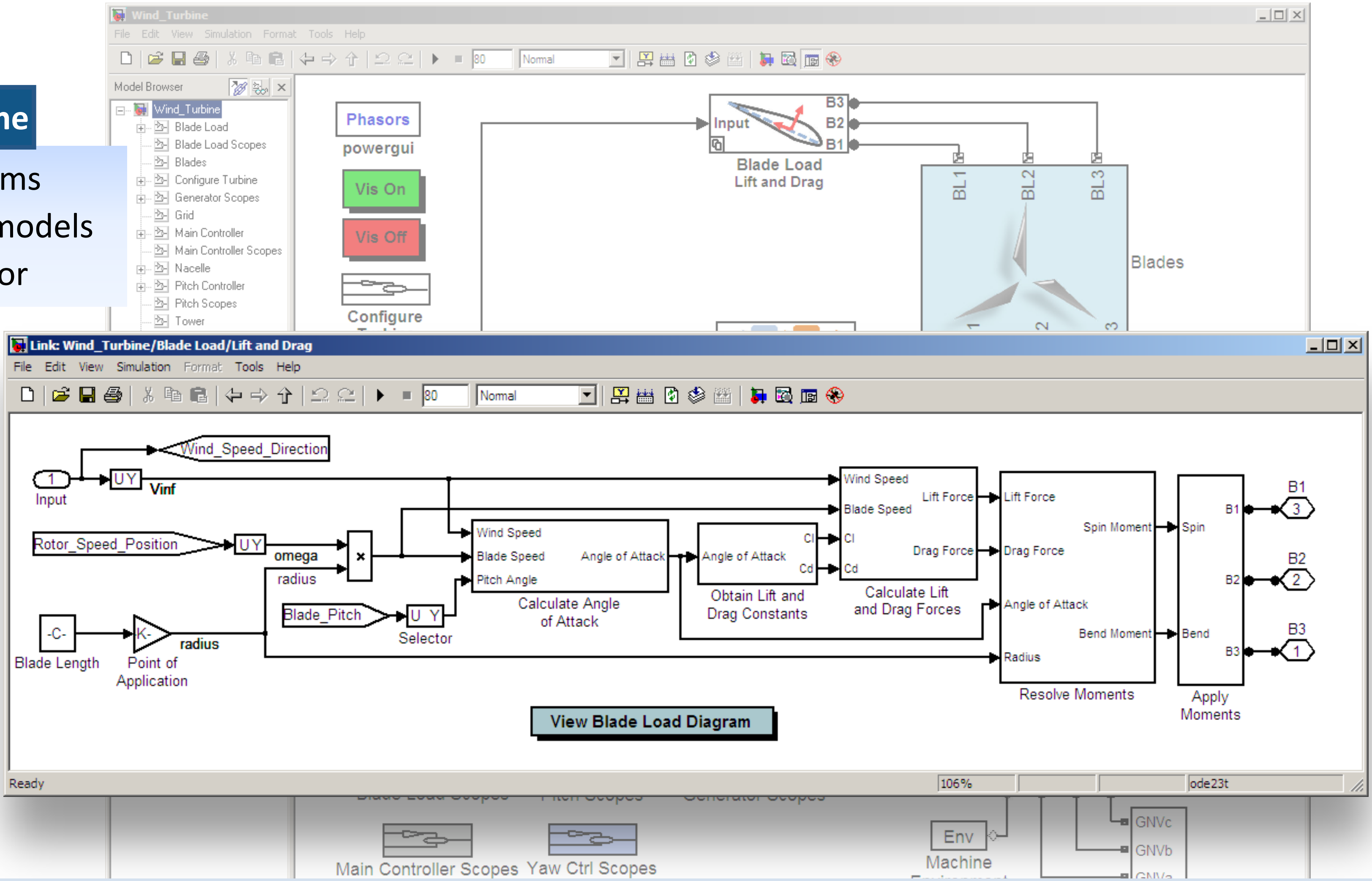
- Challenge
  - Prove absence of run time errors
  - Code reviews takes a long time
  - Coding standards
  - Testing is an ambiguious word, depends how it is implemented

- Suggestion
  - PolySpace can help
  - Formally prove absence of run time errors
  - Create reports on how well your code is tested

Multidomain System Modeling

Continuous-time

Dynamic systems
Environment models
Analog behavior



- **Ordinary differential equations (ODEs)**

  $dx/dt = f(x,u,t)$

  $y = g(x,u,t)$

- **State space (linear first-order ODEs)**

  $dx/dt = Ax + Bu$

  $y = Cx + Du$

- **Transfer functions**

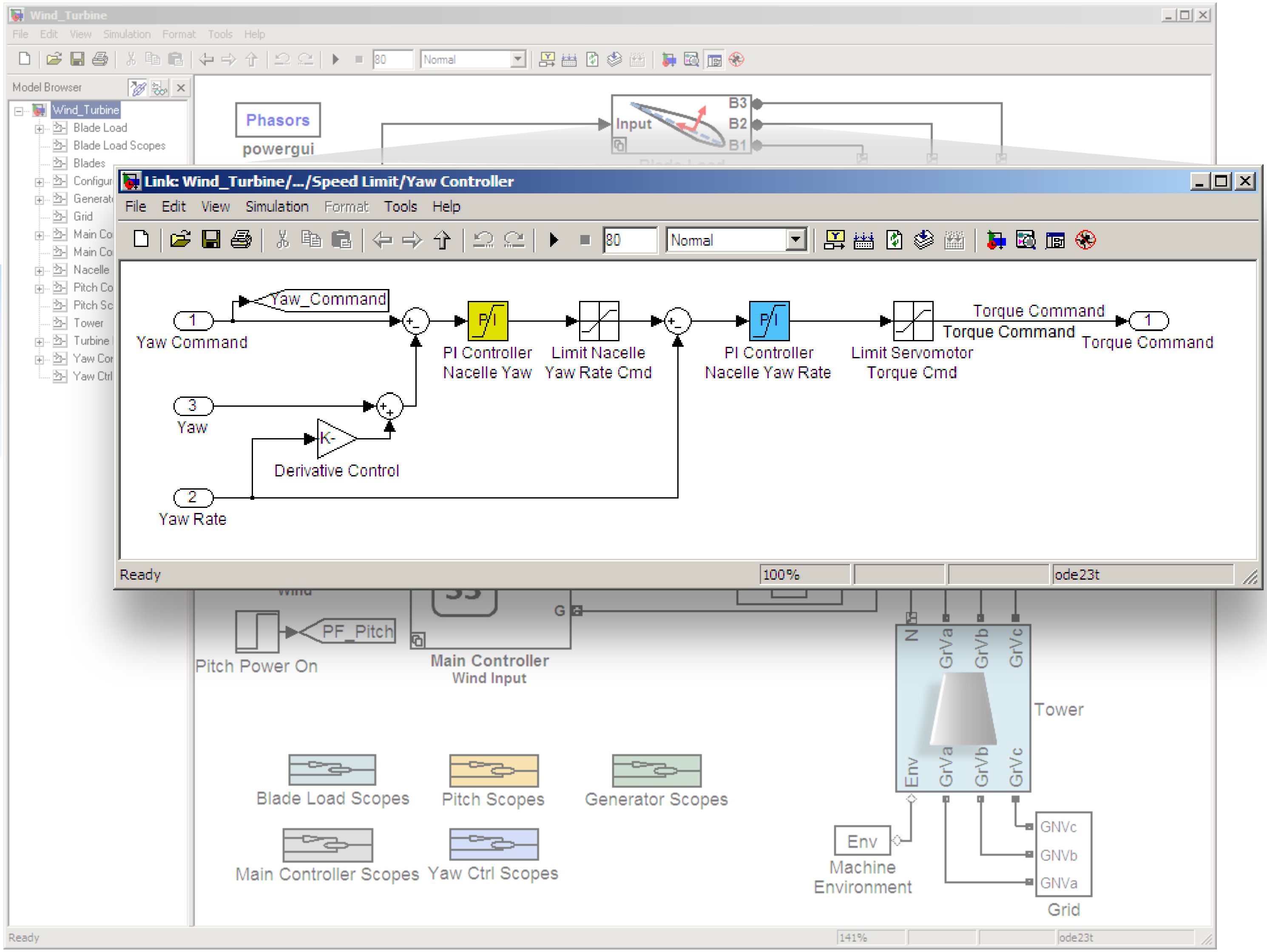
  $H(s) = b(s)/a(s)$

Multidomain System Modeling

Continuous-time

Discrete-time

- Difference Equations
- DSP
- Image/video
- Digital control



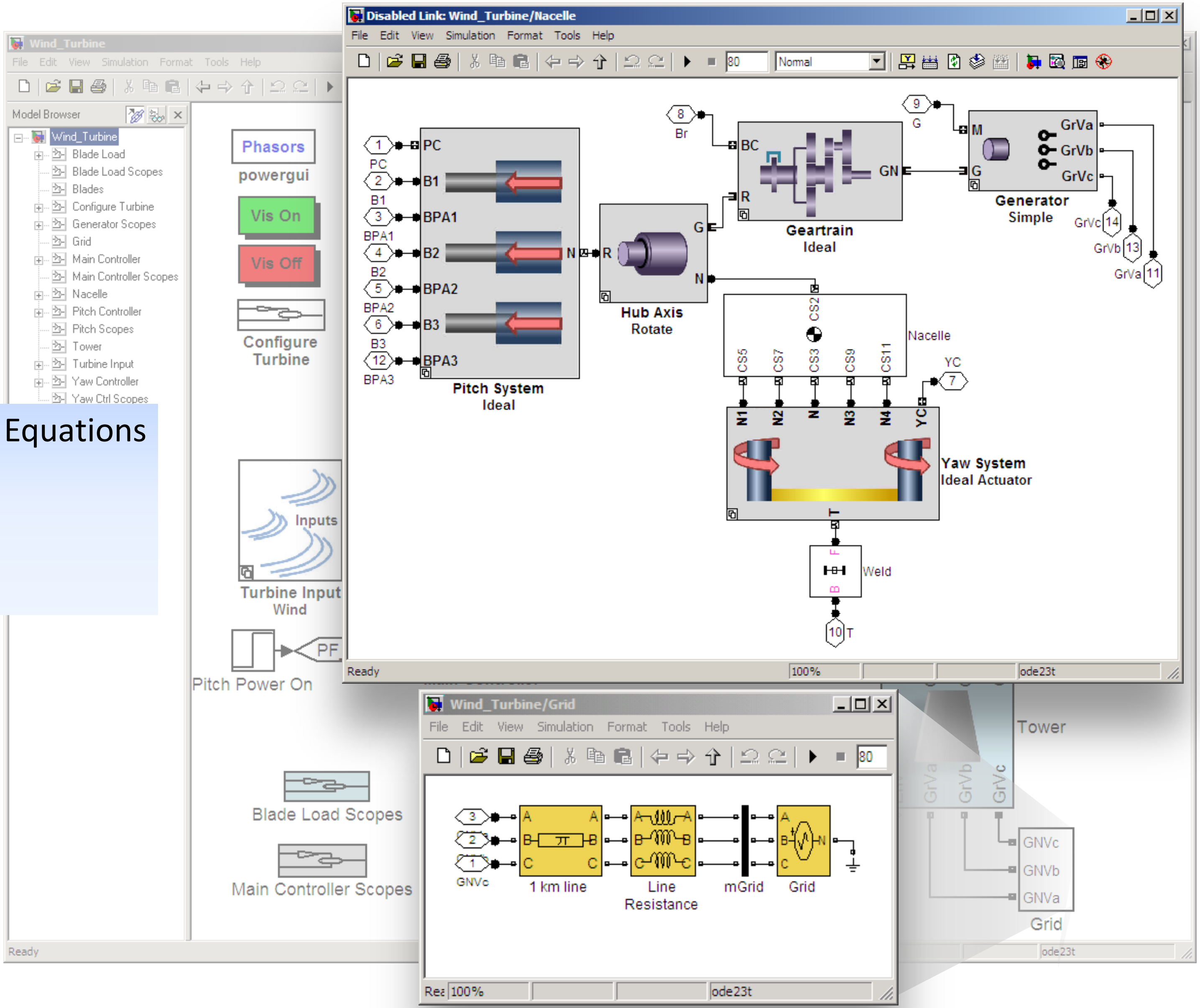
Multidomain System Modeling

Continuous-time

Discrete-time

Physical models

Differential Algebraic Equations
Electronics
Mechanics
Hydraulics
Other domains



Multidomain System Modeling

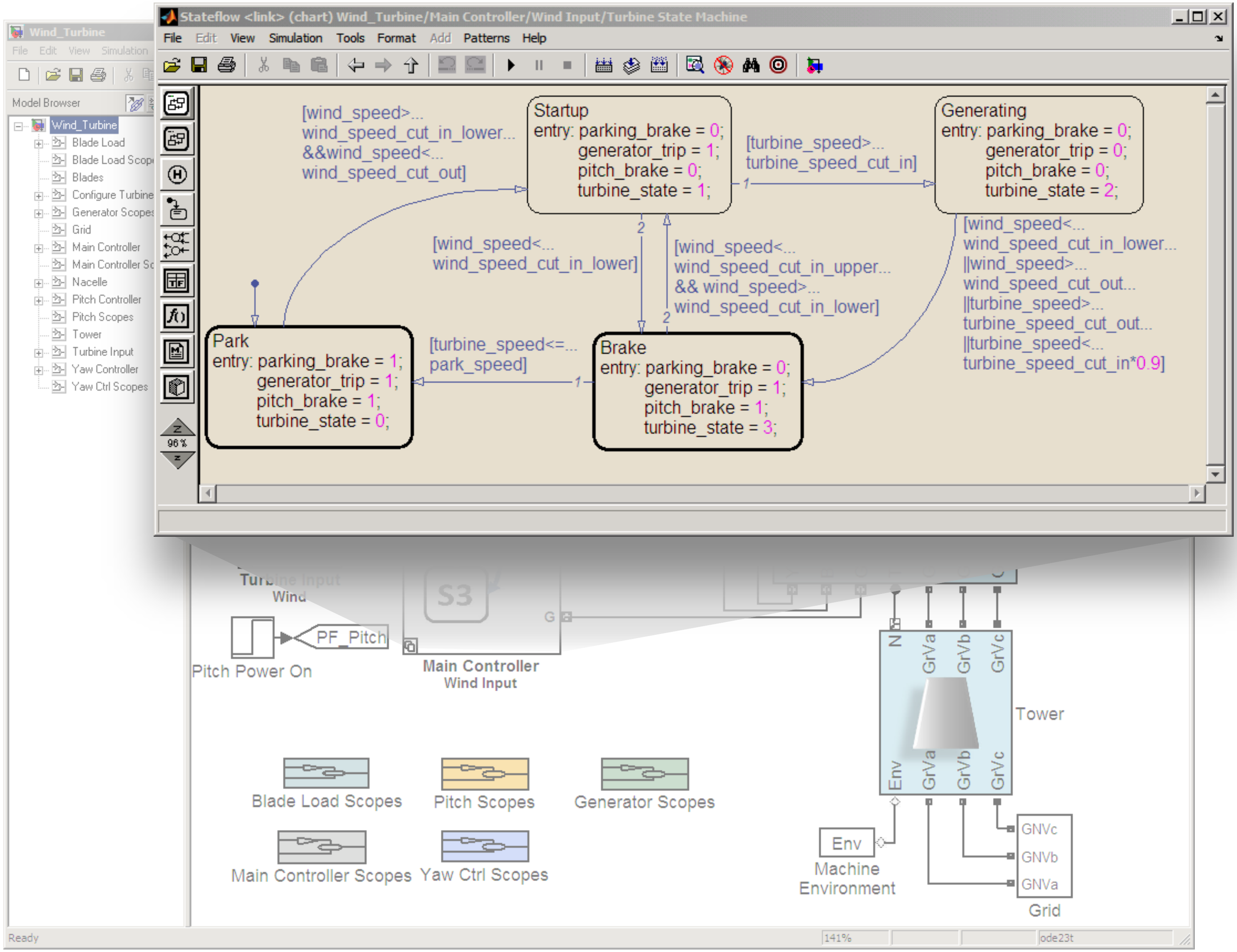
Continuous-time

Discrete-time

Physical models

State machines

Control logic
Mode logic



Stateflow <link> (chart) Wind_Turbine/Main Controller/Wind Input/Turbine State Machine

```
Startup
entry: parking_brake = 0;
       generator_trip = 1;
       pitch_brake = 0;
       turbine_state = 1;

Generating
entry: parking_brake = 0;
       generator_trip = 0;
       pitch_brake = 0;
       turbine_state = 2;

Park
entry: parking_brake = 1;
       generator_trip = 1;
       pitch_brake = 1;
       turbine_state = 0;

Brake
entry: parking_brake = 0;
       generator_trip = 1;
       pitch_brake = 1;
       turbine_state = 3;
```

```
[wind_speed>...
wind_speed_cut_in_lower...
&&wind_speed<...
wind_speed_cut_out]

[turbine_speed>...
turbine_speed_cut_in]

[wind_speed<...
wind_speed_cut_in_lower]

[wind_speed<...
wind_speed_cut_in_upper...
&& wind_speed>...
wind_speed_cut_in_lower]

[wind_speed<...
wind_speed_cut_in_lower...
||wind_speed>...
wind_speed_cut_out...
||turbine_speed>...
turbine_speed_cut_out...
||turbine_speed<...
turbine_speed_cut_in*0.9]

[turbine_speed<=...
park_speed]
```

Multidomain System Modeling

Continuous-time

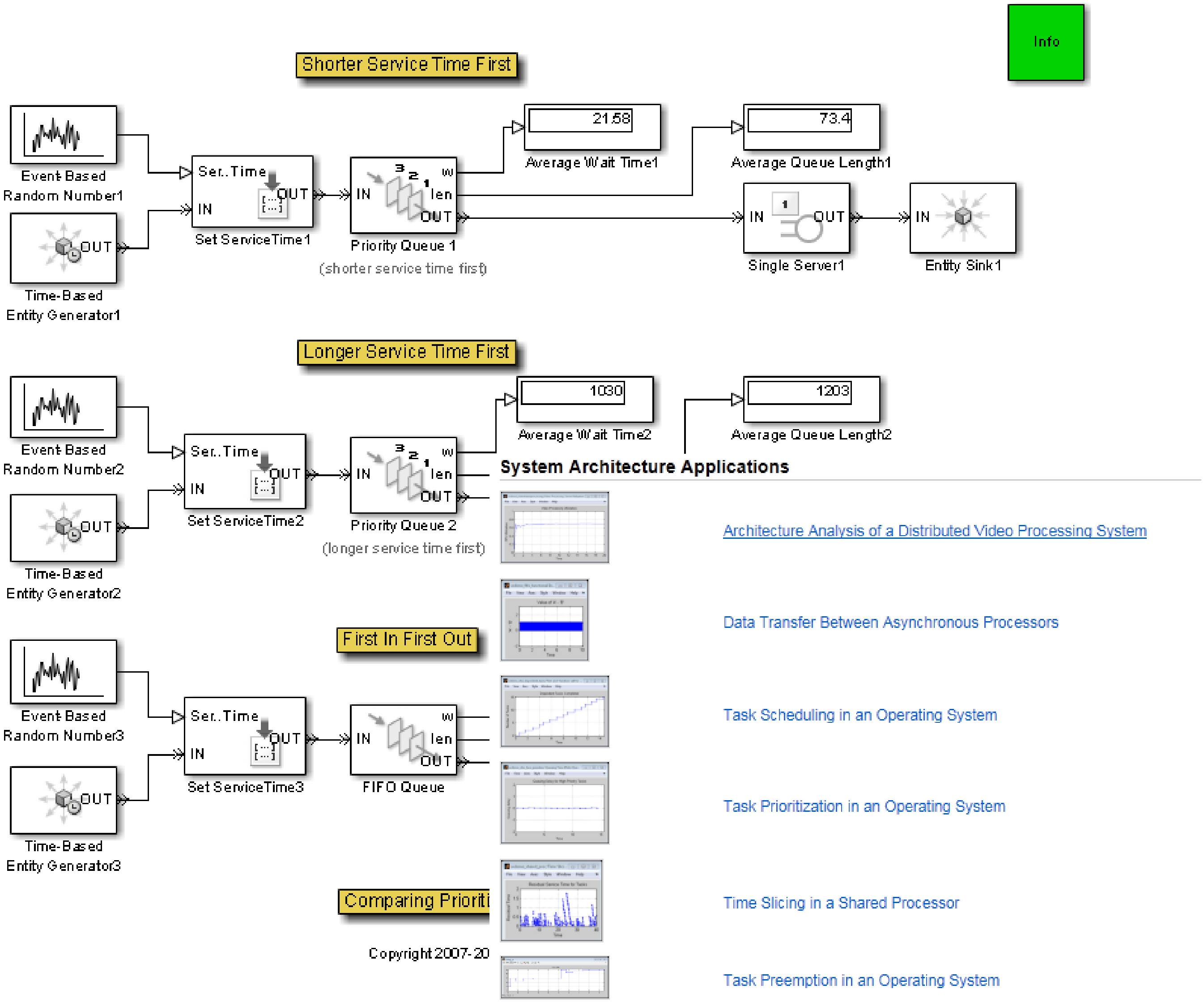
Discrete-time

Physical models

State machines

Discrete-event

SimEvents



System Architecture Applications

Architecture Analysis of a Distributed Video Processing System

Data Transfer Between Asynchronous Processors

Task Scheduling in an Operating System

Task Prioritization in an Operating System

Time Slicing in a Shared Processor

Task Preemption in an Operating System

Multidomain System Modeling

Extended Kalman Filter

Continuous-time

Discrete-time

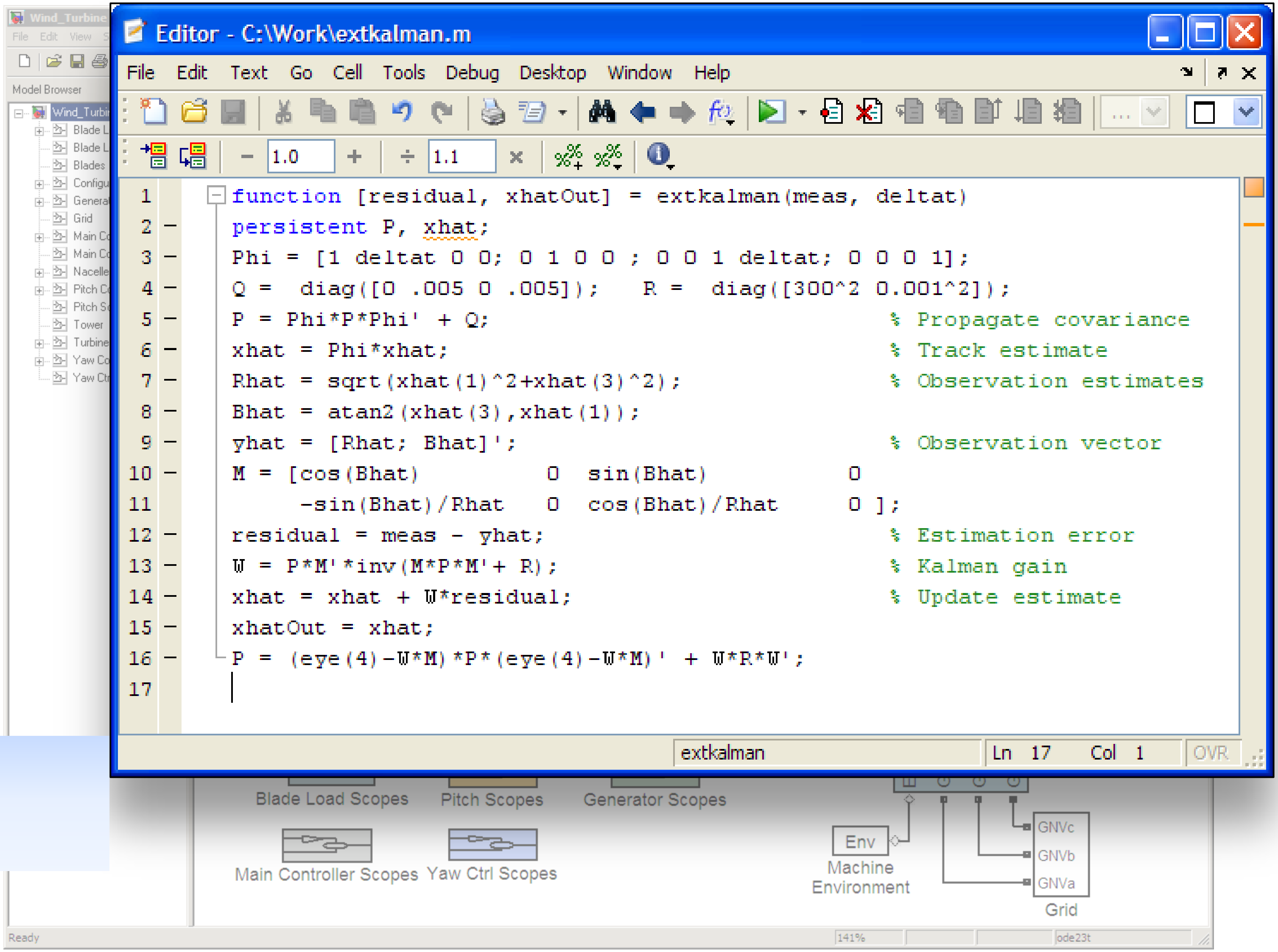
Physical models

State machines

Discrete-event

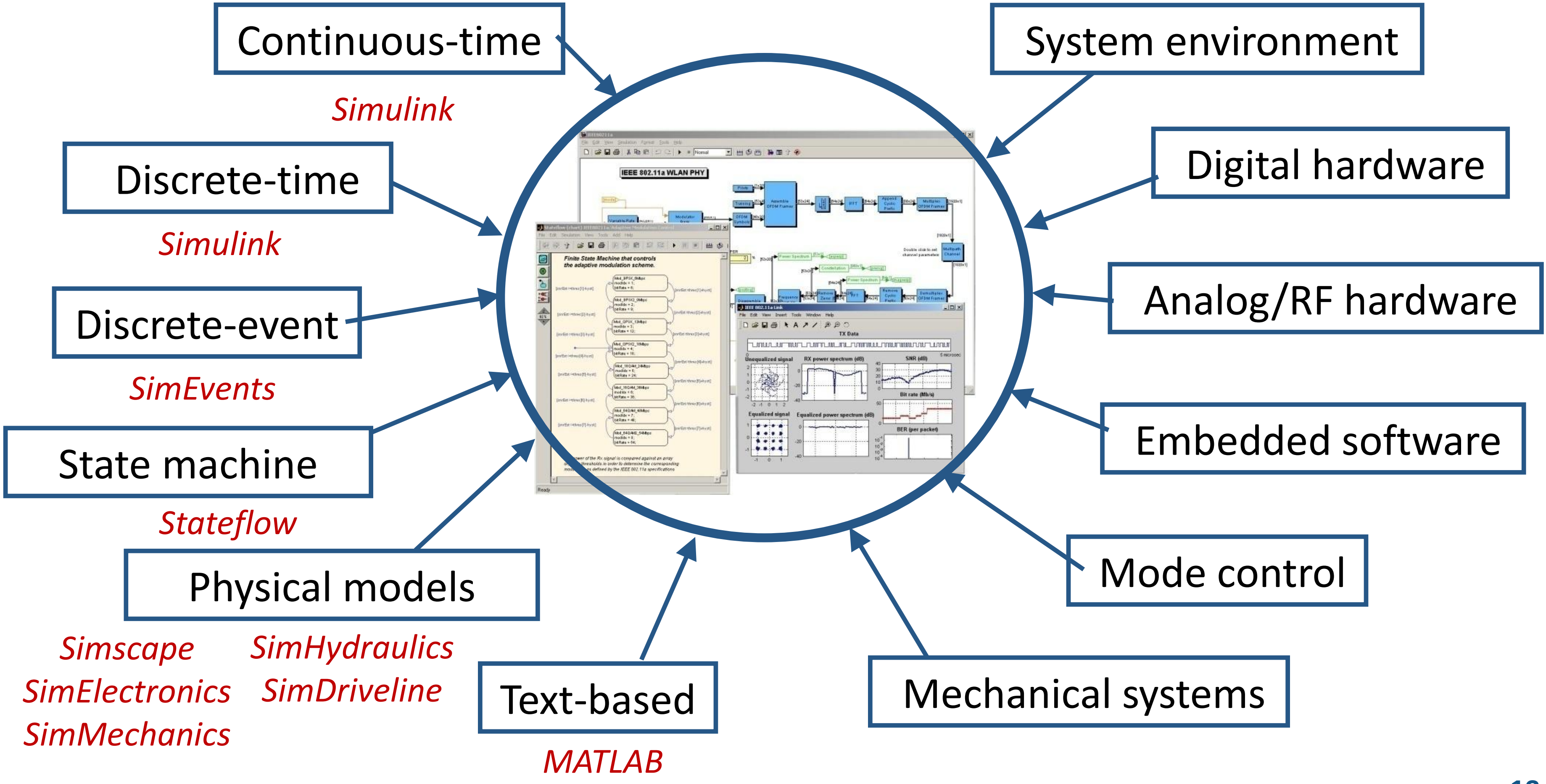
Text-based

MATLAB
Simscape language
System objects



```matlab
function [residual, xhatOut] = extkalman(meas, deltat)
persistent P, xhat;
Phi = [1 deltat 0 0; 0 1 0 0 ; 0 0 1 deltat; 0 0 0 1];
Q =  diag([0 .005 0 .005]);    R =  diag([300^2 0.001^2]);
P = Phi*P*Phi' + Q;                           % Propagate covariance
xhat = Phi*xhat;                              % Track estimate
Rhat = sqrt(xhat(1)^2+xhat(3)^2);             % Observation estimates
Bhat = atan2(xhat(3),xhat(1));
yhat = [Rhat; Bhat]';                         % Observation vector
M = [cos(Bhat)          0  sin(Bhat)         0
     -sin(Bhat)/Rhat    0  cos(Bhat)/Rhat    0 ];
residual = meas - yhat;                       % Estimation error
W = P*M'*inv(M*P*M'+ R);                      % Kalman gain
xhat = xhat + W*residual;                     % Update estimate
xhatOut = xhat;
P = (eye(4)-W*M)*P*(eye(4)-W*M)' + W*R*W';
```

Modeling Multidomain Systems



Modeling domains

Continuous-time
Simulink

Discrete-time
Simulink

Discrete-event
SimEvents

State machine
Stateflow

Physical models
Simscape *SimHydraulics*
SimElectronics *SimDriveline*
SimMechanics

Text-based
MATLAB

System elements

System environment

Digital hardware

Analog/RF hardware

Embedded software

Mode control

Mechanical systems

Tools in Industry – Transitioning to Industry

- Open Video User Story

Video User Story

- Open Lear Video User Story

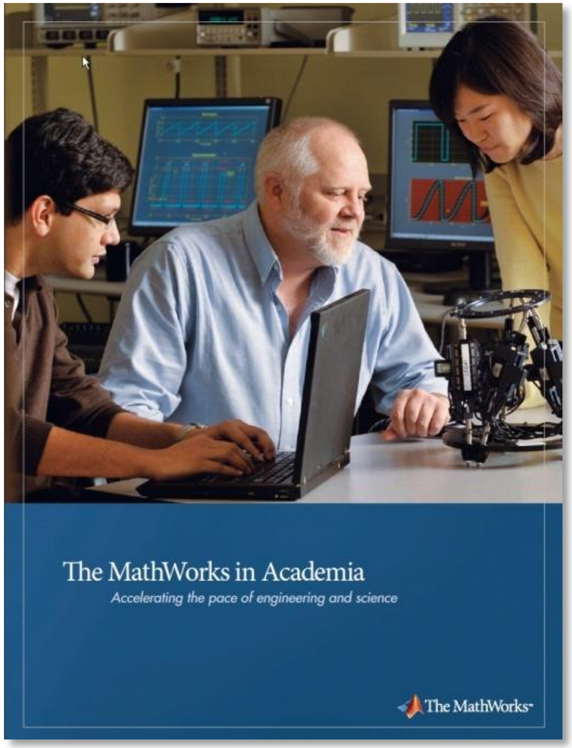
Focal Points

- With MATLAB/Simulink my professors think I am clever
- Knowing MATLAB/Simulink will help you get a job!
- Multi-Modeling Techniques are often needed

Available Resources

Visit MathWorks Web Site

- Learn about MathWorks products
- Discover resources for learning, teaching, and research
- Learn how MathWorks products are used in academia and industry

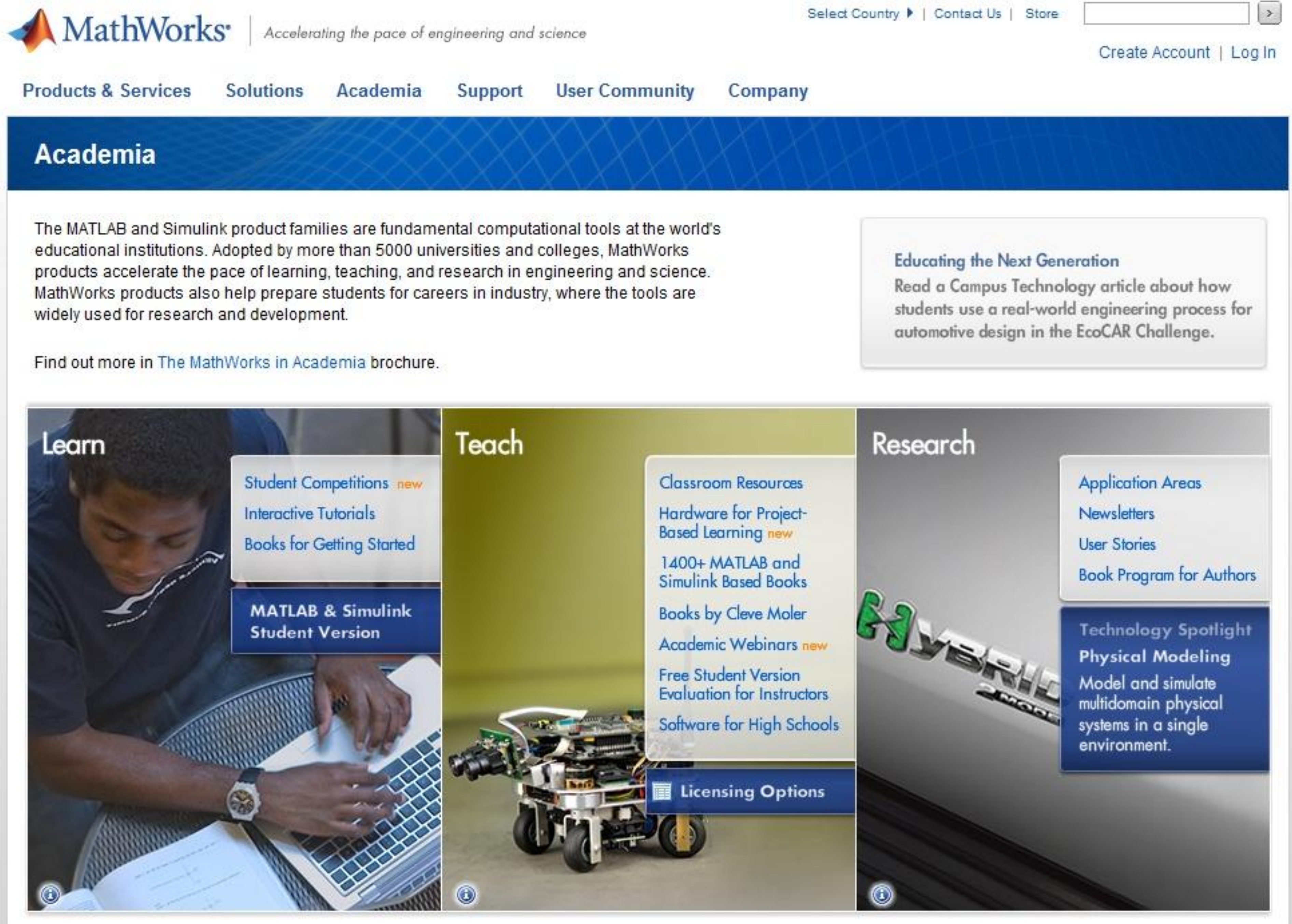


The MathWorks in Academia

Accelerating the pace of engineering and science

The MathWorks

Visit www.mathworks.com/academia



Academia

The MATLAB and Simulink product families are fundamental computational tools at the world's educational institutions. Adopted by more than 5000 universities and colleges, MathWorks products accelerate the pace of learning, teaching, and research in engineering and science. MathWorks products also help prepare students for careers in industry, where the tools are widely used for research and development.

Find out more in The MathWorks in Academia brochure.

Educating the Next Generation
Read a Campus Technology article about how students use a real-world engineering process for automotive design in the EcoCAR Challenge.

Learn

- Student Competitions new
- Interactive Tutorials
- Books for Getting Started

MATLAB & Simulink Student Version

Teach

- Classroom Resources
- Hardware for Project-Based Learning new
- 1400+ MATLAB and Simulink Based Books
- Books by Cleve Moler
- Academic Webinars new
- Free Student Version Evaluation for Instructors
- Software for High Schools

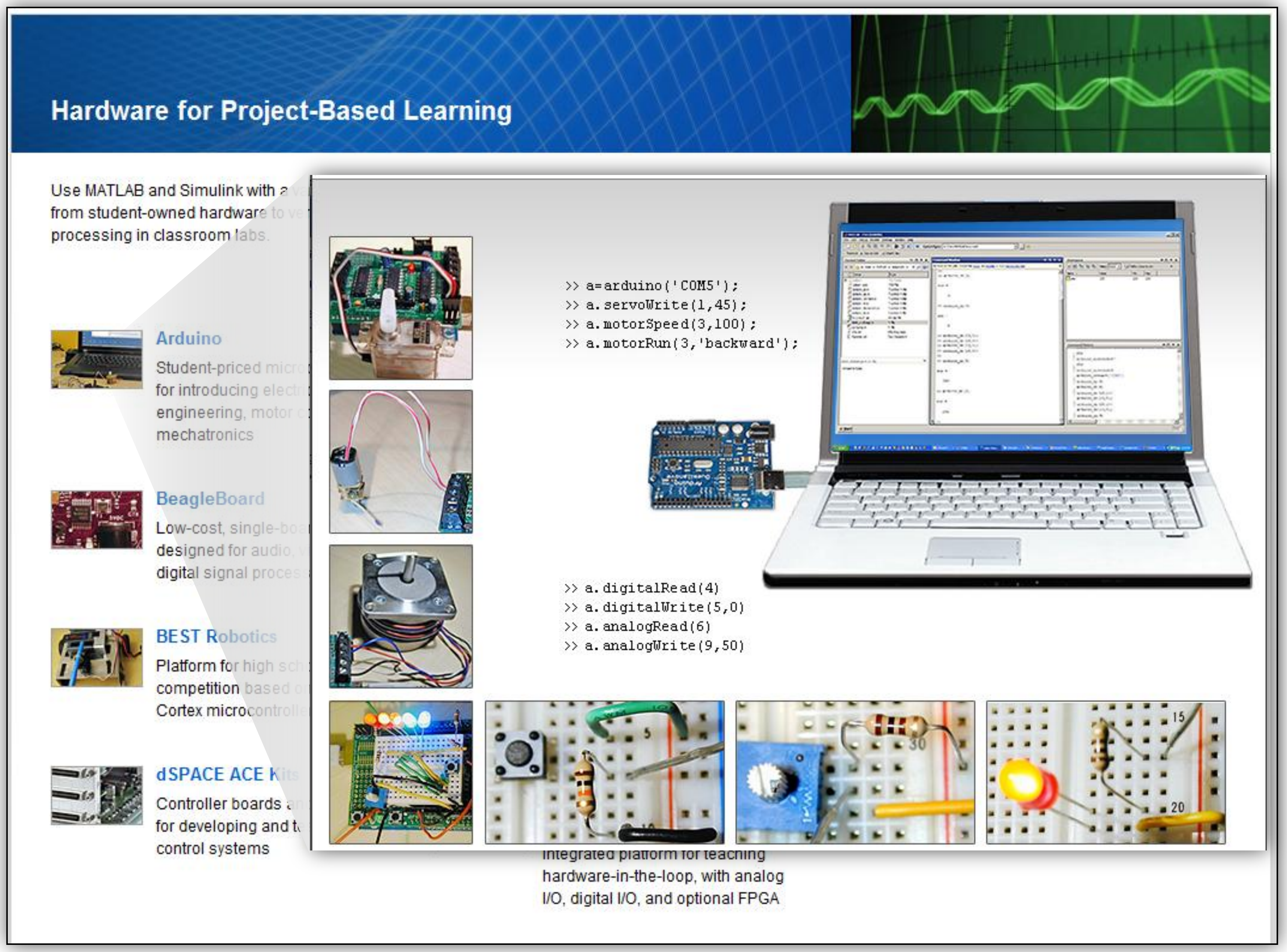
Licensing Options

Research

- Application Areas
- Newsletters
- User Stories
- Book Program for Authors

Technology Spotlight
Physical Modeling
Model and simulate multidomain physical systems in a single environment.

Visit www.mathworks.com/academia



Hardware for Project-Based Learning

Use MATLAB and Simulink with a va
from student-owned hardware to ve
processing in classroom labs.

Arduino

Student-priced micro
for introducing electr
engineering, motor c
mechatronics

BeagleBoard

Low-cost, single-boa
designed for audio, v
digital signal proces

BEST Robotics

Platform for high sch
competition based o
Cortex microcontroll

dSPACE ACE Kits

Controller boards a
for developing and t
control systems

```
>> a=arduino('COM5');
>> a.servoWrite(1,45);
>> a.motorSpeed(3,100);
>> a.motorRun(3,'backward');
```

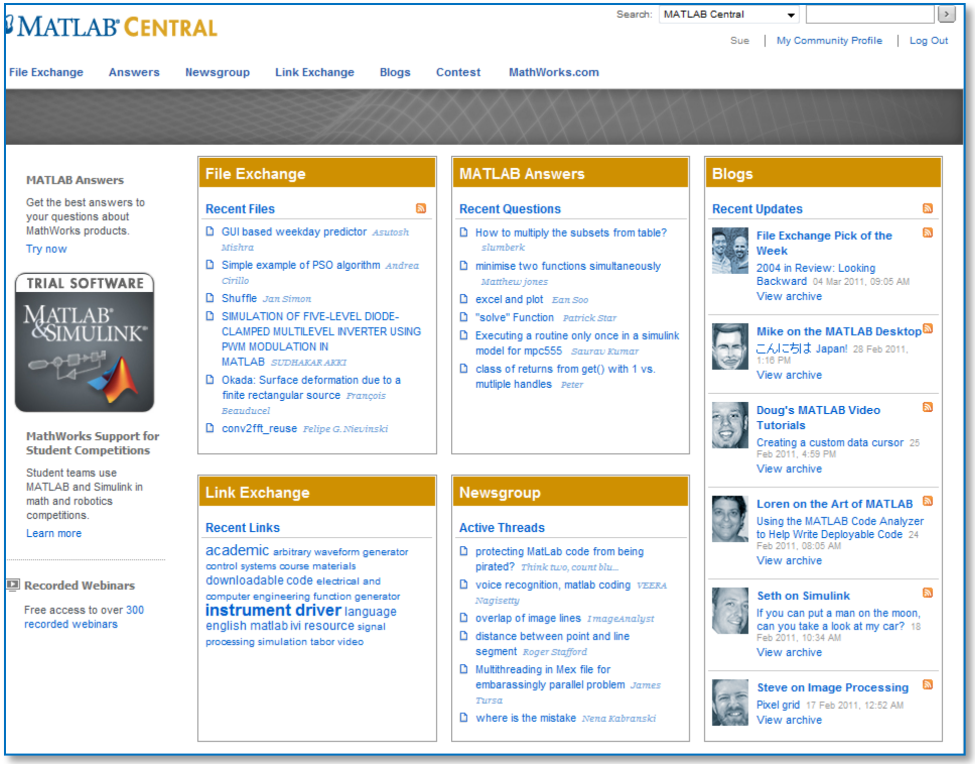
```
>> a.digitalRead(4)
>> a.digitalWrite(5,0)
>> a.analogRead(6)
>> a.analogWrite(9,50)
```

integrated platform for teaching
hardware-in-the-loop, with analog
I/O, digital I/O, and optional FPGA

Visit www.mathworks.com/academia/hardware-resources/

MATLAB Central

- Open exchange for the MATLAB and Simulink user community
  - 1.2 million visits per month
- File Exchange
  - Upload/download free files including MATLAB code, Simulink models, and documents
  - Rate files, comment, and ask questions
- Newsgroup
  - Web forum and newsgroup for technical discussions about MATLAB and Simulink
- Blogs
  - Read posts from key MathWorks developers who design and build the products



MATLAB Answers

Get the best answers to your questions about MathWorks products.

Try now

MathWorks Support for Student Competitions

Student teams use MATLAB and Simulink in math and robotics competitions.

Learn more

Recorded Webinars

Free access to over 300 recorded webinars

File Exchange

Recent Files

- GUI based weekday predictor *Asutosh Mishra*
- Simple example of PSO algorithm *Andrea Cirillo*
- Shuffle *Jan Simon*
- SIMULATION OF FIVE-LEVEL DIODE-CLAMPED MULTILEVEL INVERTER USING PWM MODULATION IN MATLAB *SUDHAKAR AKKI*
- Okada: Surface deformation due to a finite rectangular source *François Beauducel*
- conv2fft_reuse *Felipe G.Nievinski*

MATLAB Answers

Recent Questions

- How to multiply the subsets from table? *slumberk*
- minimise two functions simultaneously *Matthew jones*
- excel and plot *Ean Soo*
- "solve" Function *Patrick Star*
- Executing a routine only once in a simulink model for mpc555 *Saurav Kumar*
- class of returns from get() with 1 vs. mutliple handles *Peter*

Blogs

Recent Updates

File Exchange Pick of the Week
2004 in Review: Looking Backward 04 Mar 2011, 09:05 AM
View archive

Mike on the MATLAB Desktop
こんにちは Japan! 28 Feb 2011, 1:16 PM
View archive

Doug's MATLAB Video Tutorials
Creating a custom data cursor 25 Feb 2011, 4:59 PM
View archive

Loren on the Art of MATLAB
Using the MATLAB Code Analyzer to Help Write Deployable Code 24 Feb 2011, 08:05 AM
View archive

Seth on Simulink
If you can put a man on the moon, can you take a look at my car? 18 Feb 2011, 10:34 AM
View archive

Steve on Image Processing
Pixel grid 17 Feb 2011, 12:52 AM
View archive

Link Exchange

Recent Links

academic arbitrary waveform generator control systems course materials downloadable code electrical and computer engineering function generator instrument driver language english matlab ivi resource signal processing simulation tabor video

Newsgroup

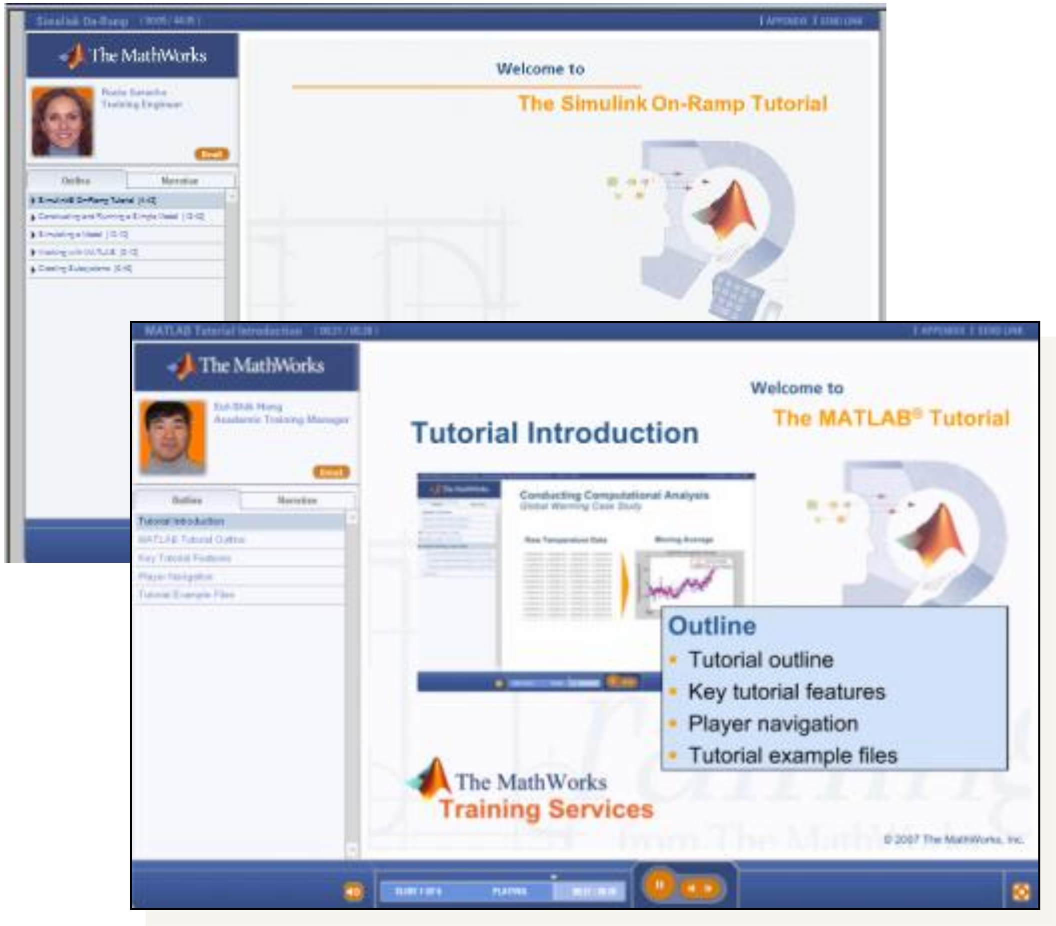
Active Threads

- protecting MatLab code from being pirated? *Think two, count blu...*
- voice recognition, matlab coding *VEERA Nagisetty*
- overlap of image lines *ImageAnalyst*
- distance between point and line segment *Roger Stafford*
- Multithreading in Mex file for embarassingly parallel problem *James Tursa*
- where is the mistake *Nena Kabranski*

Visit www.mathworks.com/matlabcentral

Learning Resources

- **Interactive Video Tutorials** – Students learn the basics outside of the classroom with self-guided tutorials provided by MathWorks
  - MATLAB
  - Simulink
  - Signal Processing
  - Control Systems
  - Computational Mathematics



Welcome to

The Simulink On-Ramp Tutorial

Tutorial Introduction

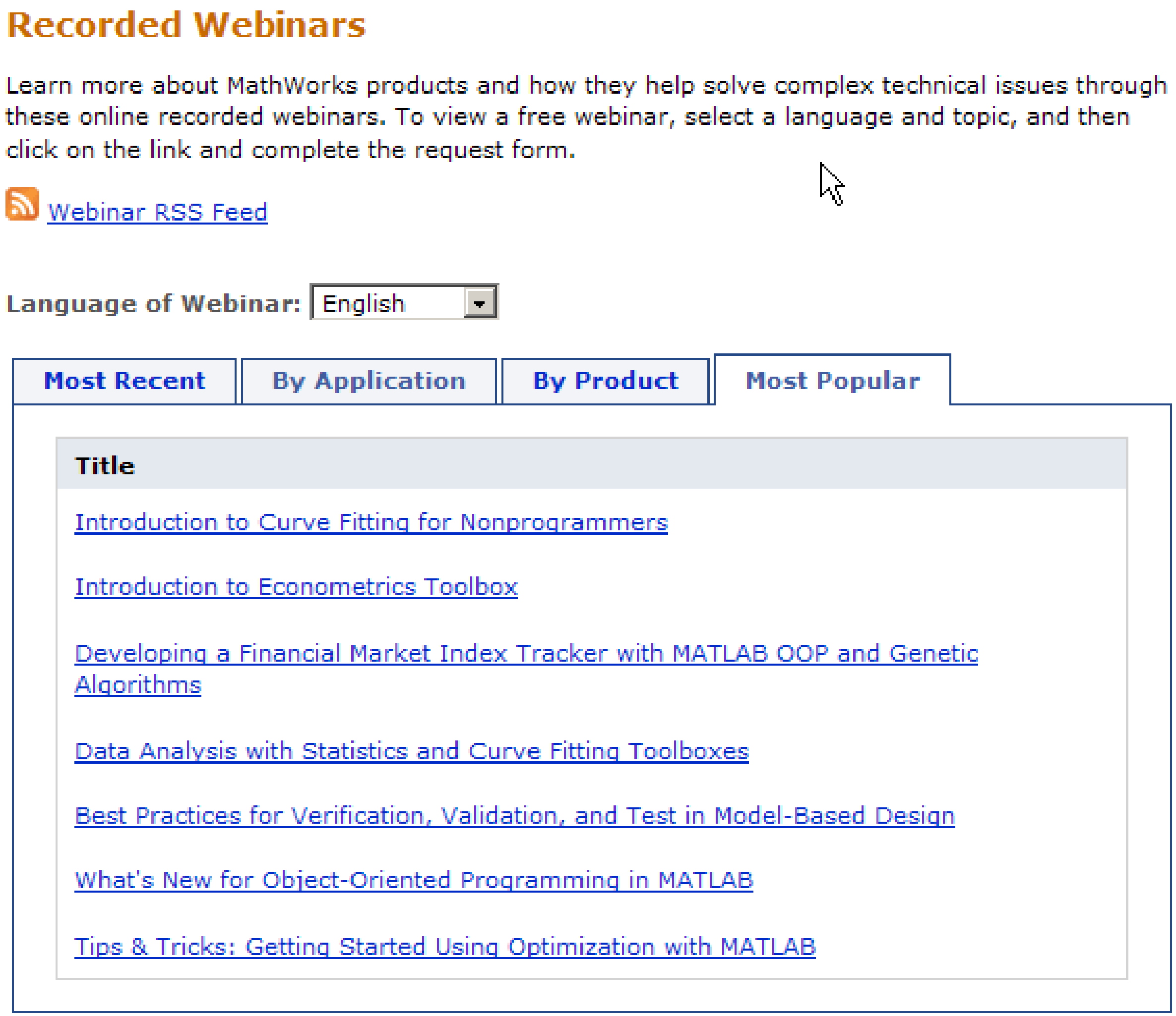
Welcome to

The MATLAB® Tutorial

Outline

- Tutorial outline
- Key tutorial features
- Player navigation
- Tutorial example files

Visit www.mathworks.com/academia/student_center/tutorials

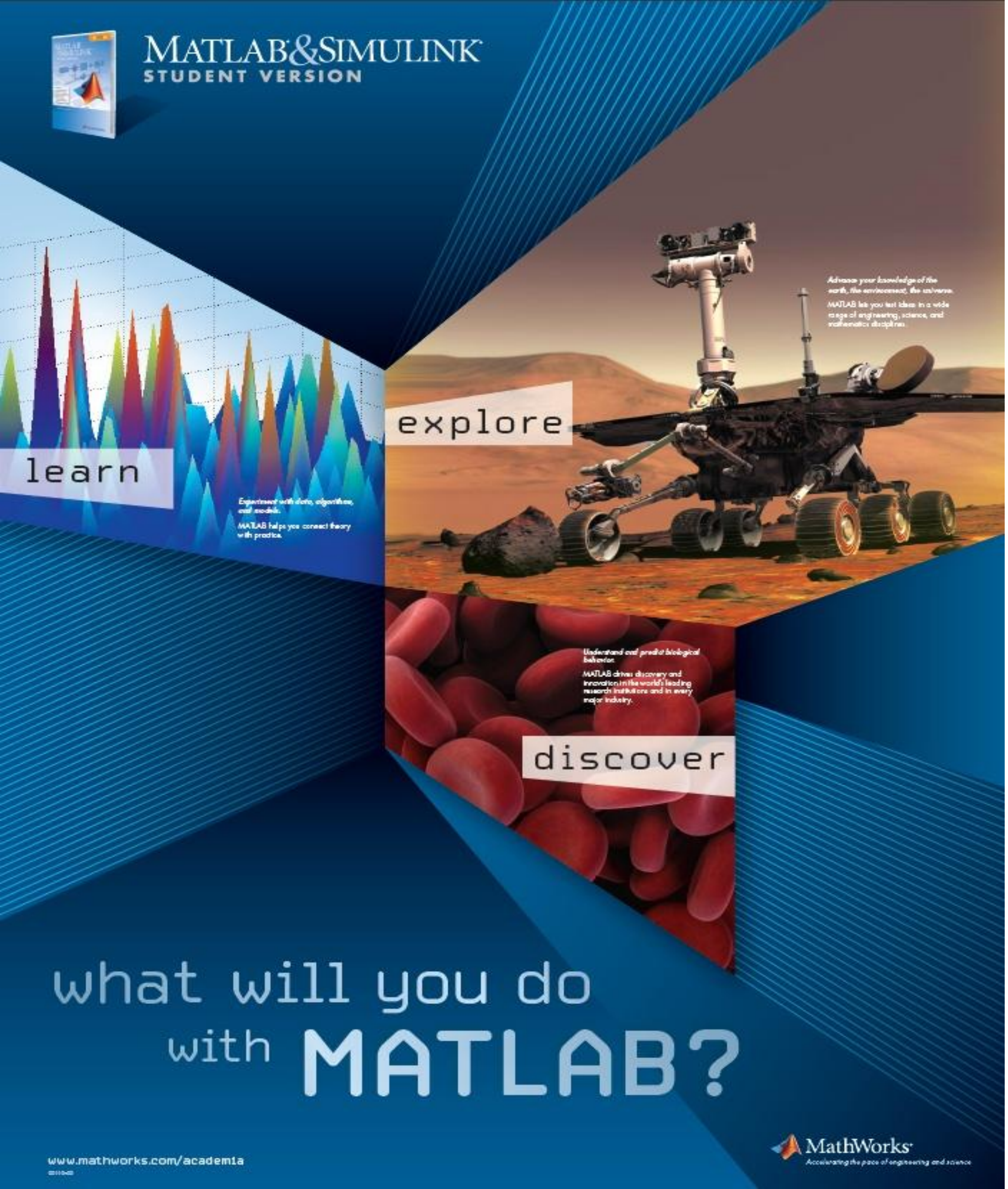


Recorded Webinars

Learn more about MathWorks products and how they help solve complex technical issues through these online recorded webinars. To view a free webinar, select a language and topic, and then click on the link and complete the request form.

Language of Webinar: English

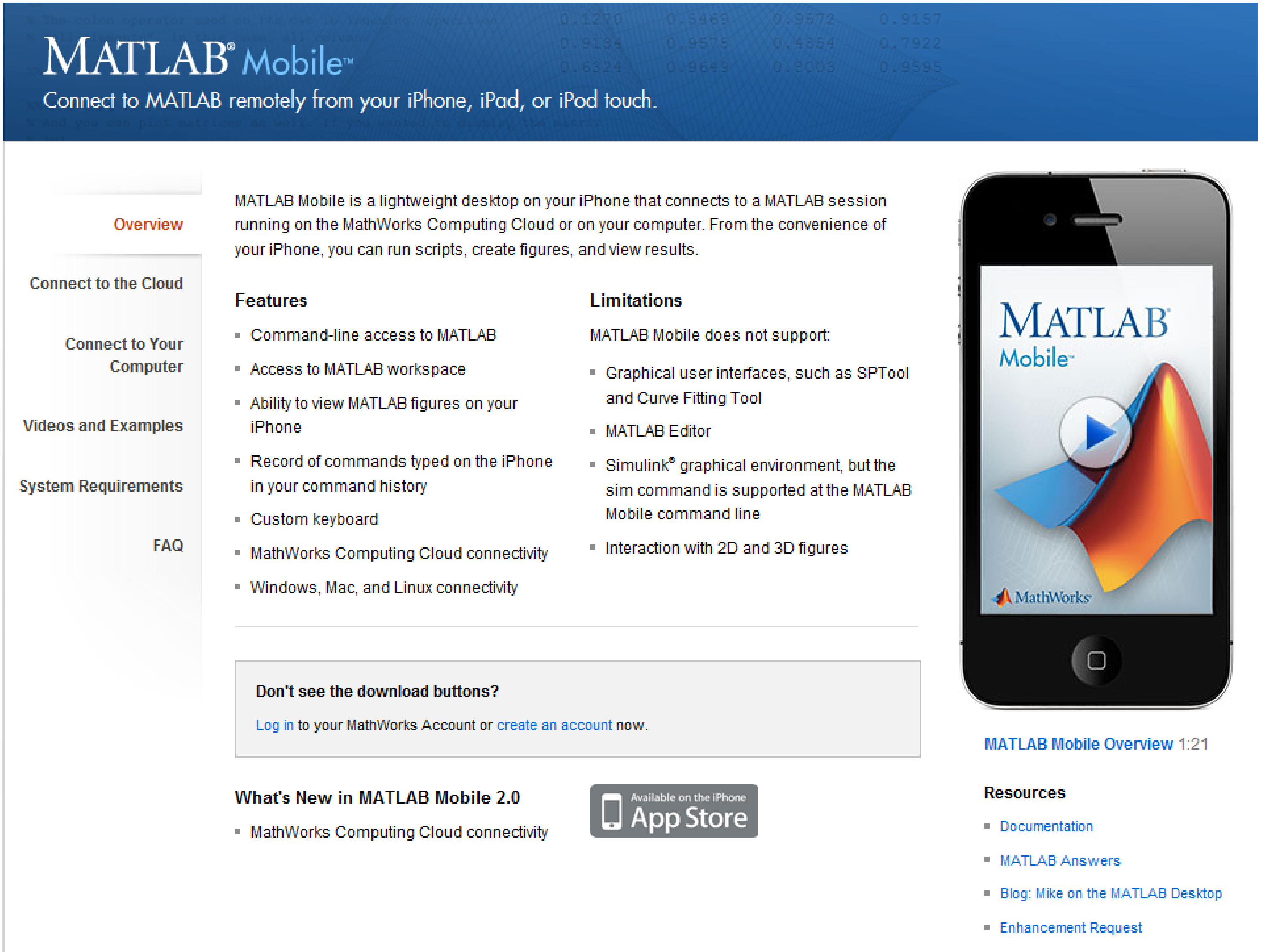
Title
Introduction to Curve Fitting for Nonprogrammers
Introduction to Econometrics Toolbox
Developing a Financial Market Index Tracker with MATLAB OOP and Genetic Algorithms
Data Analysis with Statistics and Curve Fitting Toolboxes
Best Practices for Verification, Validation, and Test in Model-Based Design
What's New for Object-Oriented Programming in MATLAB
Tips & Tricks: Getting Started Using Optimization with MATLAB


MATLAB&SIMULINK
STUDENT VERSION
learn
Experiment with data, algorithms, and models.
MATLAB helps you connect theory with practice.
explore
Advance your knowledge of the earth, the environment, the universe.
MATLAB lets you test ideas in a wide range of engineering, science, and mathematics disciplines.
Understand and predict biological behavior.
MATLAB drives discovery and innovation in the world's leading research institutions and in every major industry.
discover
what will you do
with MATLAB?
www.mathworks.com/academia
MathWorks
Accelerating the pace of engineering and science

Student Version R2012b

- MATLAB
- Simulink
- 7 popular add-on products
  - Control System Toolbox
  - Signal Processing Toolbox
  - DSP System Toolbox
  - Statistics Toolbox
  - Optimization Toolbox
  - Image Processing Toolbox
  - Symbolic Math Toolbox

MATLAB Mobile



MATLAB® Mobile™

Connect to MATLAB remotely from your iPhone, iPad, or iPod touch.

MATLAB Mobile is a lightweight desktop on your iPhone that connects to a MATLAB session running on the MathWorks Computing Cloud or on your computer. From the convenience of your iPhone, you can run scripts, create figures, and view results.

Features

- Command-line access to MATLAB
- Access to MATLAB workspace
- Ability to view MATLAB figures on your iPhone
- Record of commands typed on the iPhone in your command history
- Custom keyboard
- MathWorks Computing Cloud connectivity
- Windows, Mac, and Linux connectivity

Limitations

MATLAB Mobile does not support:

- Graphical user interfaces, such as SPTool and Curve Fitting Tool
- MATLAB Editor
- Simulink® graphical environment, but the sim command is supported at the MATLAB Mobile command line
- Interaction with 2D and 3D figures

Don't see the download buttons?

Log in to your MathWorks Account or create an account now.

What's New in MATLAB Mobile 2.0

- MathWorks Computing Cloud connectivity

MATLAB Mobile Overview 1:21

Resources

- Documentation
- MATLAB Answers
- Blog: Mike on the MATLAB Desktop
- Enhancement Request

Thank you for attention