

 MathWorks


MATLAB&SIMULINK

## On Computational Semantics as a Precise Foundation of an Industrial Toolchain for Analysis and Design of Multi-domain Systems


**Pieter J. Mosterman**

Senior Research Scientist  
Design Automation Department  
 MathWorks

Adjunct Professor  
School of Computer Science  
 McGill




© 2010 The MathWorks, Inc.


 MathWorks


MATLAB&SIMULINK

## On Computational Semantics as a Precise Foundation of an Industrial Toolchain for Analysis and Design of Multi-domain Systems


Pieter J. Mosterman

Senior Research Scientist  
Design Automation Department  
 MathWorks

Adjunct Professor  
School of Computer Science  
 McGill



© 2010 The MathWorks, Inc.





MathWorks

MATLAB&SIMULINK


## On Computational Semantics as a Precise Foundation of an Industrial Toolchain for Analysis and Design of **Multi-domain Systems**

Pieter J. Mosterman

Senior Research Scientist  
Design Automation Department  
 MathWorks

Adjunct Professor  
School of Computer Science  
 McGill

© 2010 The MathWorks, Inc.





MathWorks

MATLAB&SIMULINK

## On Computational Semantics as a Precise Foundation of an **Industrial Toolchain** for Analysis and Design of Multi-domain Systems

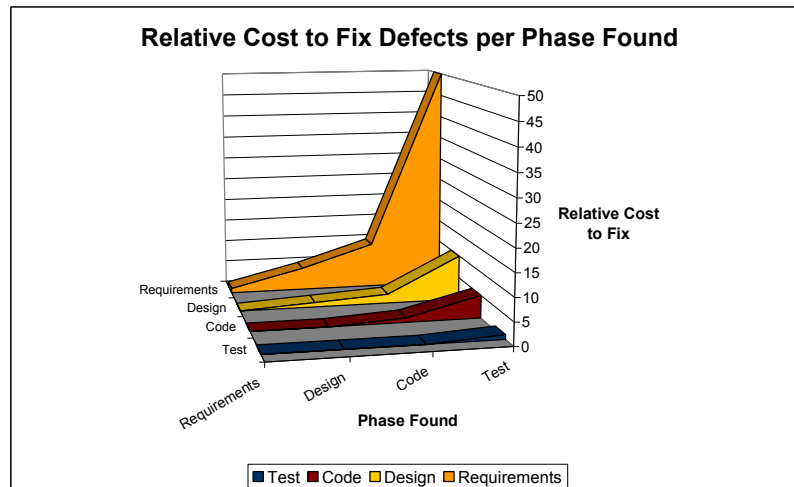
Pieter J. Mosterman

Senior Research Scientist  
Design Automation Department  
 MathWorks

Adjunct Professor  
School of Computer Science  
 McGill

© 2010 The MathWorks, Inc.

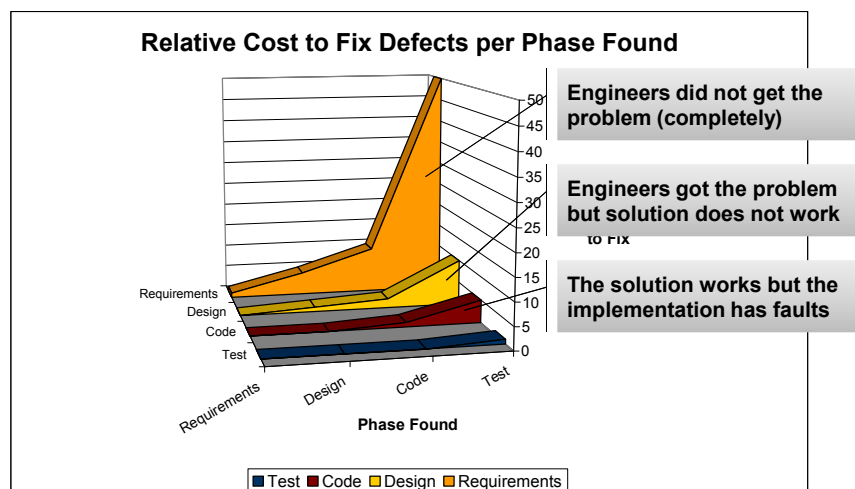
## Accumulating cost of latent errors



*NASA, Return on Investment for Independent Verification & Validation, 2004*

5

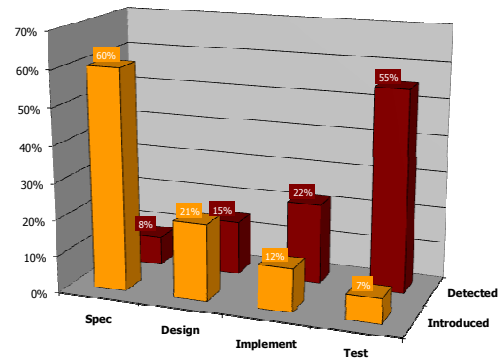
## Accumulating cost of latent errors



*NASA, Return on Investment for Independent Verification & Validation, 2004*

6

## Where errors are introduced and where errors are detected

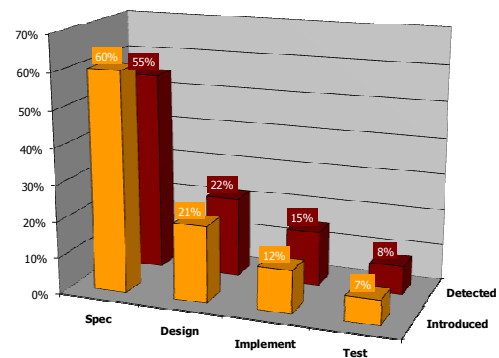


Paul Yanik, "Migration from Simulation to Verification with ModelSim®," EDA Tech Forum, Newton, MA, March 11, 2004

7

## Where errors can be detected

- Early verification with Model-Based Design



The MathWorks, Early Verification presentation

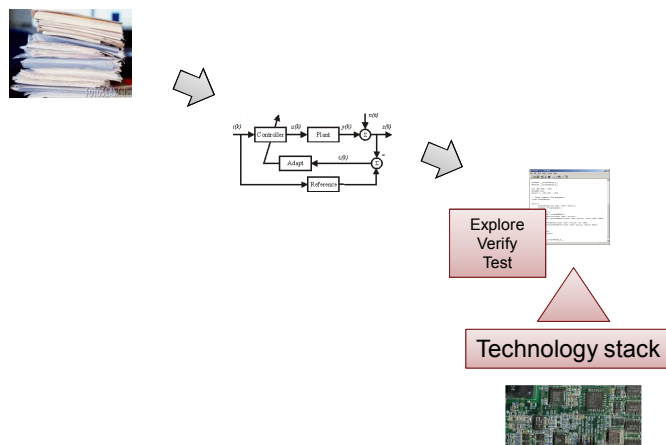
8

## Agenda

- ▪ Model-Based Design
- Computer Automated Multiparadigm Modeling
- Modeling time as discrete events
- A unifying semantic domain
- A heterogeneous system example
- Conclusions

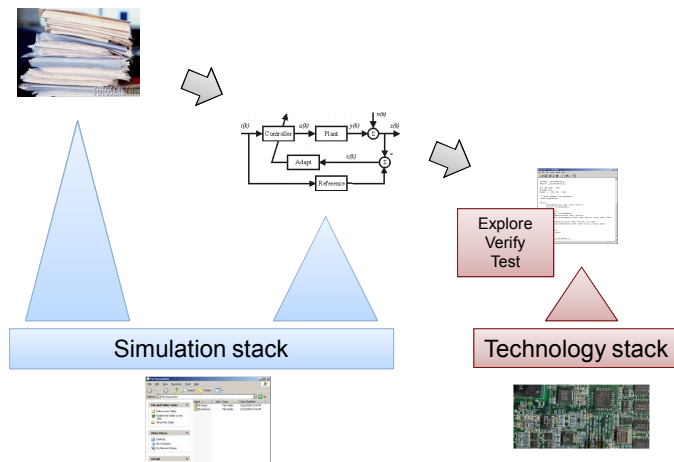
9

## Gaps in the traditional design workflow



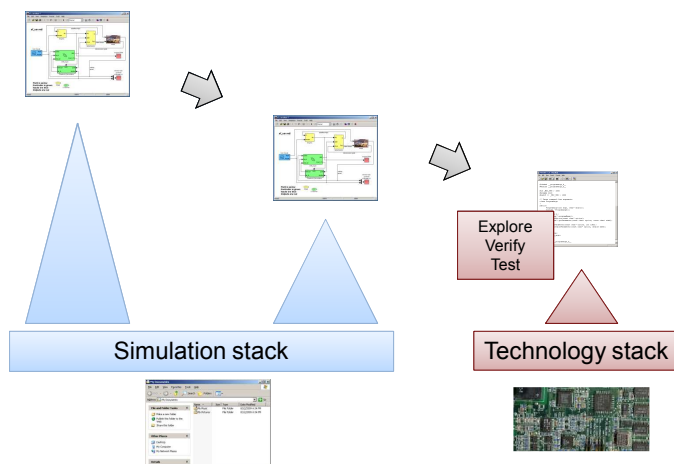
10

## Gaps in the traditional design workflow

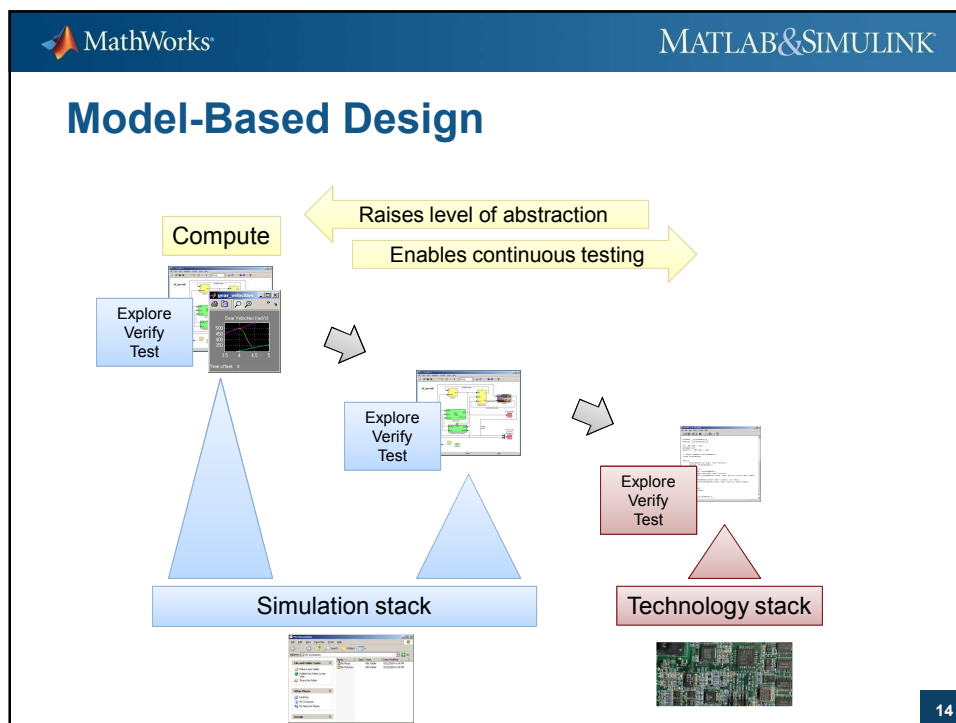
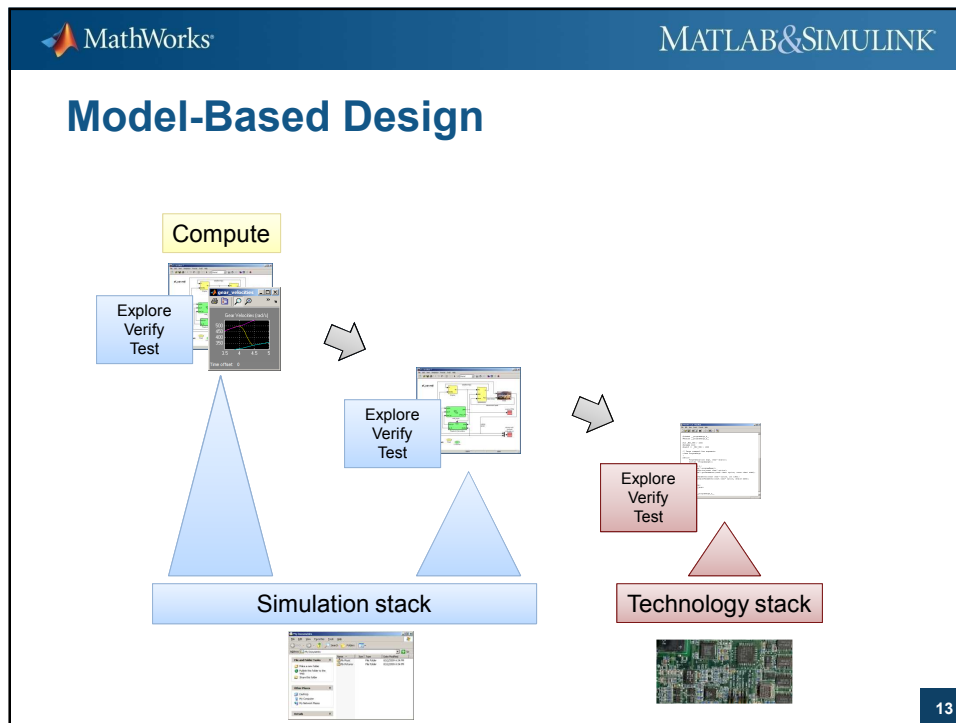


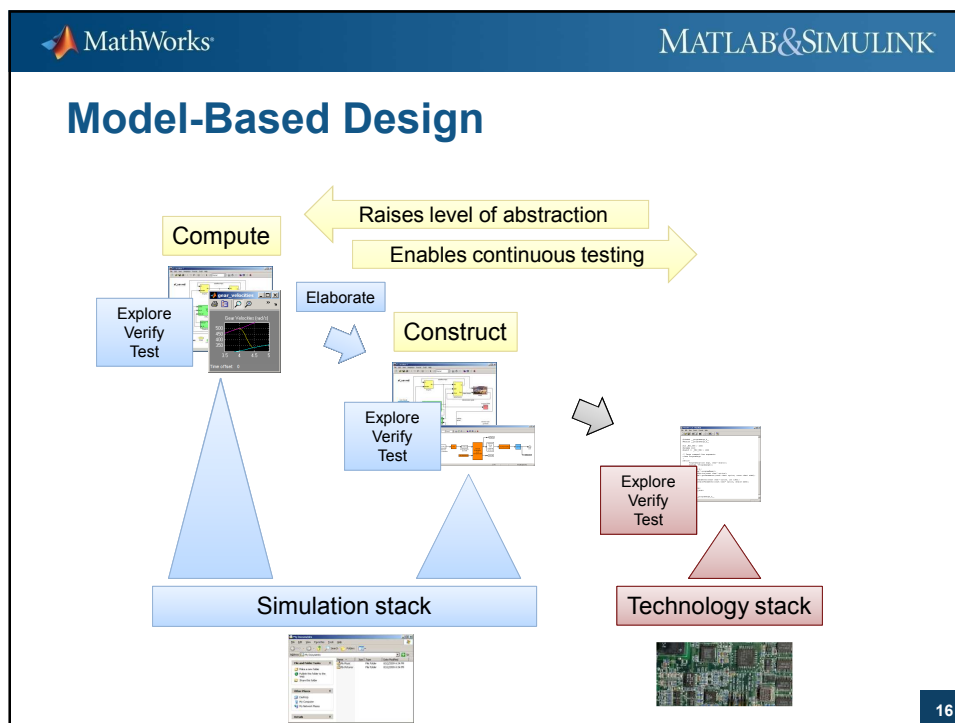
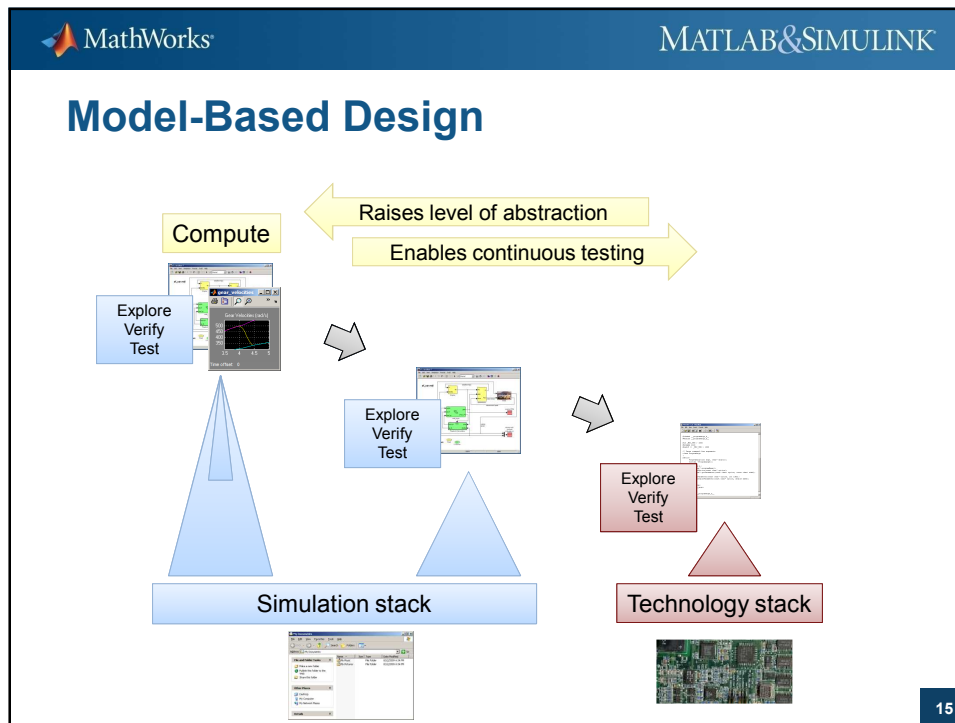
11

## Gaps in the traditional design workflow

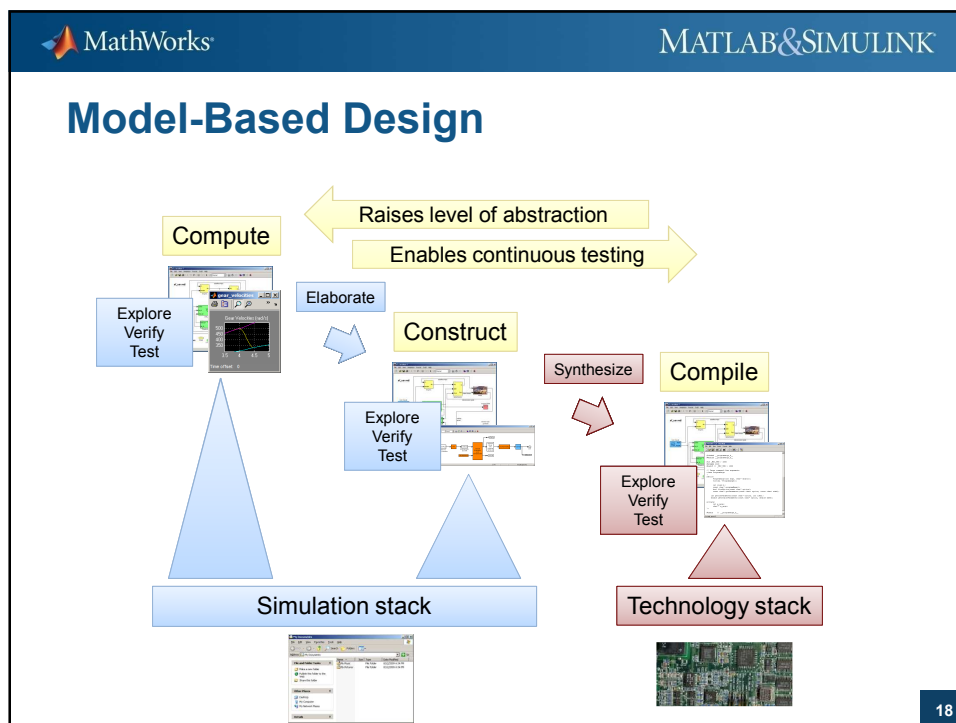
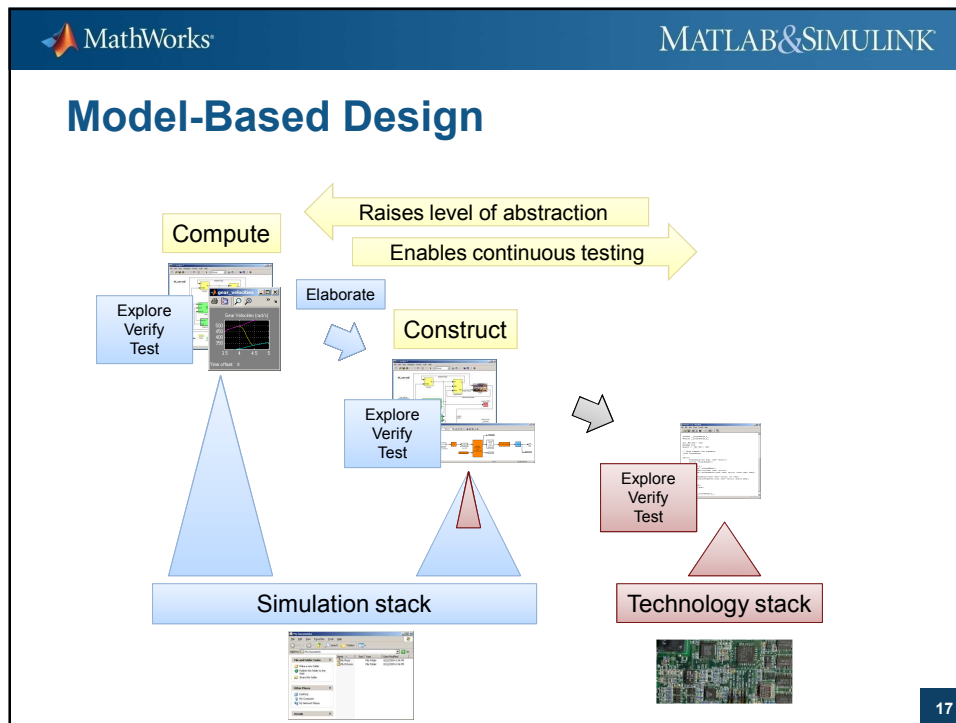


12

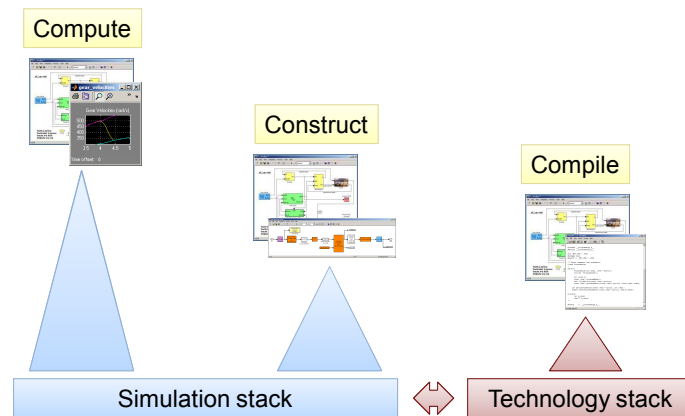






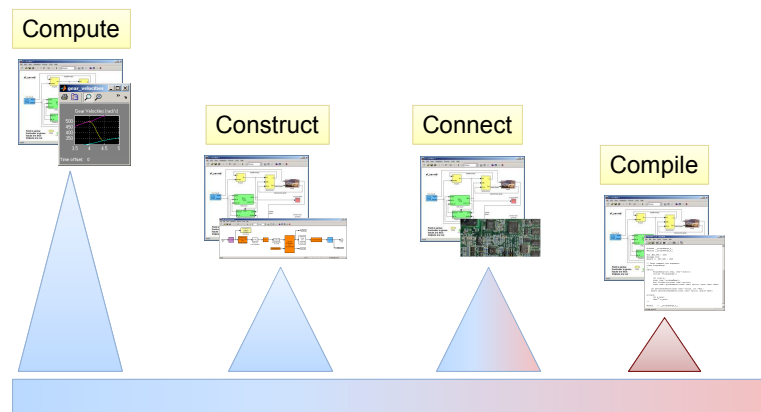


## Model-Based Design



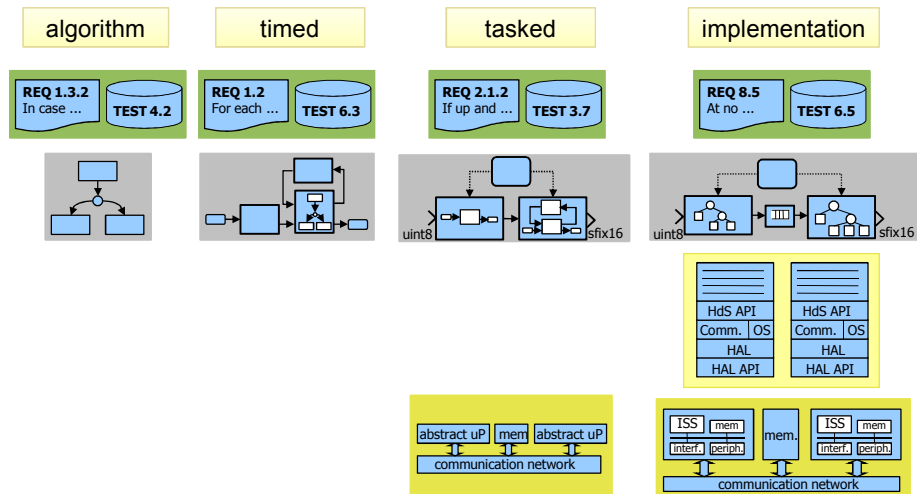
19

## Model-Based Design



20

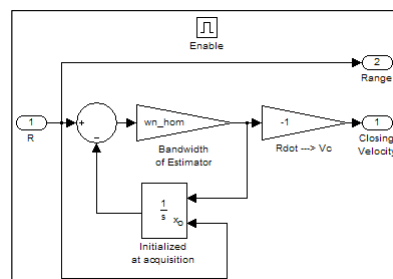
## Designing an embedded system



In collaboration with Katalin Popovici, TIMA

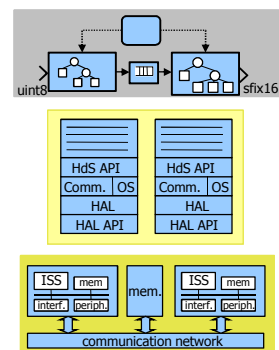
21

## Early and progressive system testing ... ?



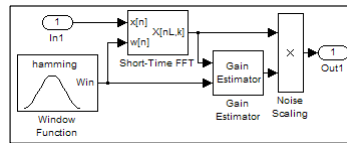
### implementation

REQ 8.5 At no ... TEST 6.5

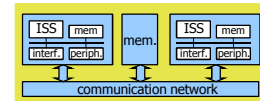
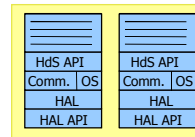
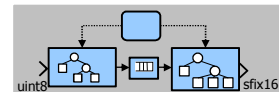
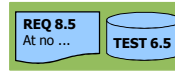


22

## Early and progressive system testing ... ?



### implementation

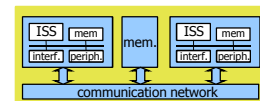
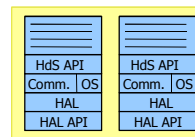
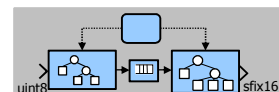
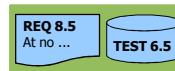


23

## Early and progressive system testing ... ?

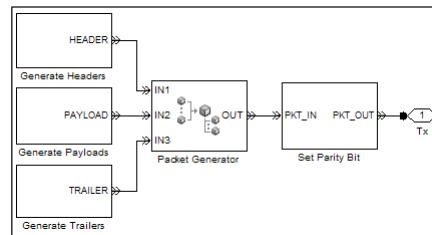
```
function [residual, xhatOut] = extkalman(meas, deltat)
persistent P, xhat;
Phi = [1 deltat 0 0; 0 1 0 0; 0 0 1 deltat; 0 0 0 1];
Q = diag([0 .005 0 .005]); R = diag([300^2 0.001^2]);
P = Phi*P*Phi' + Q; % Propagate covariance
xhat = Phi*xhat; % Track estimate
Rhat = sqrt(xhat(1)^2+xhat(3)^2); % Observation 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); % Kalmain gain
xhat = xhat + W*residual; % Update estimate
xhatOut = xhat;
P = (eye(4)-W*M)*P*(eye(4)-W*M)' + W*R*W';
```

### implementation

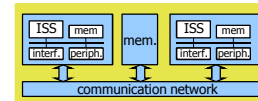
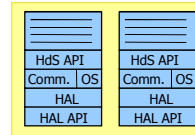
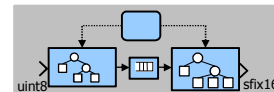
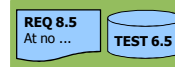


24

## Early and progressive system testing ... ?

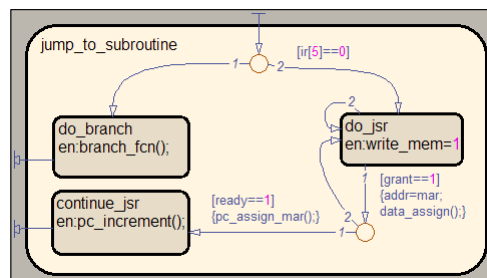


### implementation

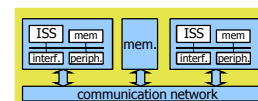
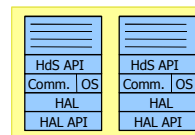
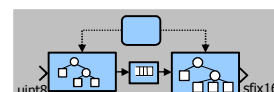
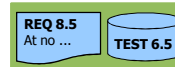


25

## Early and progressive system testing ... ?

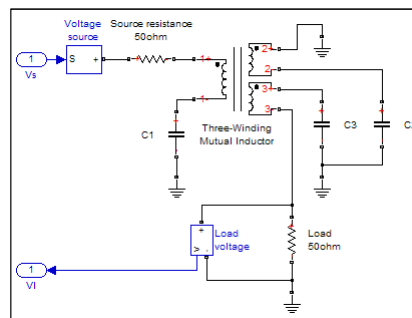


### implementation

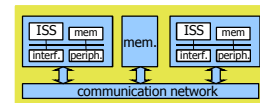
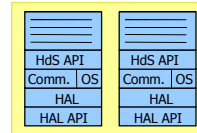
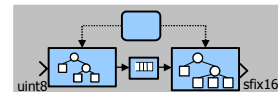


26

## Early and progressive system testing ... ?



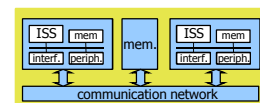
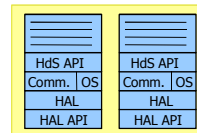
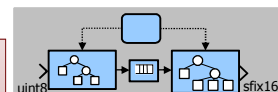
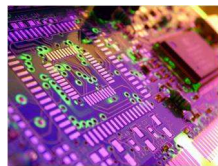
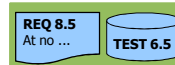
### implementation



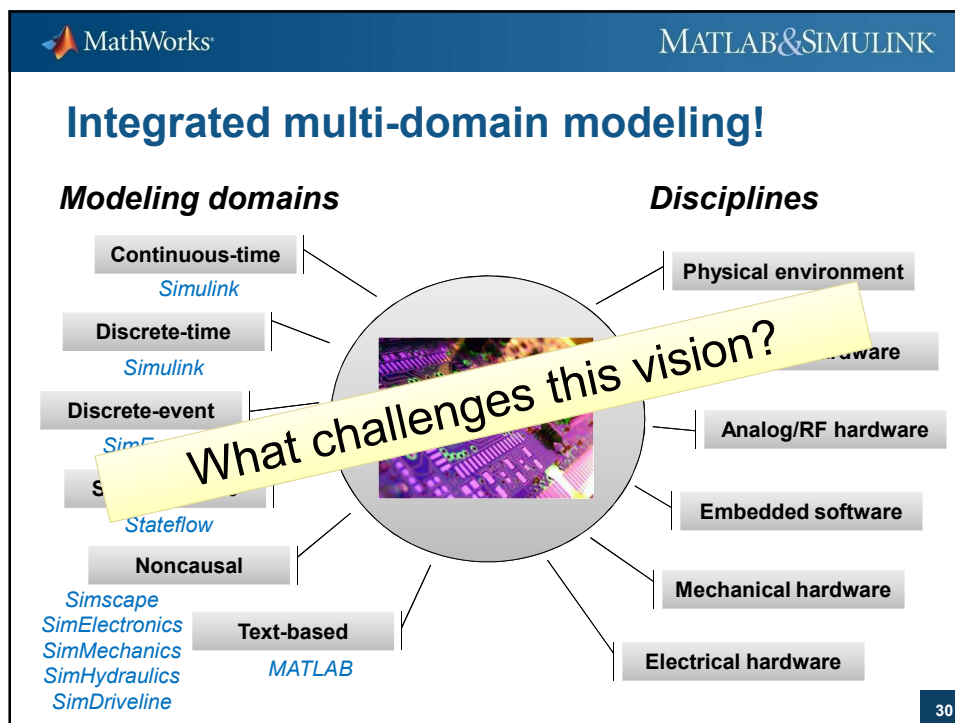
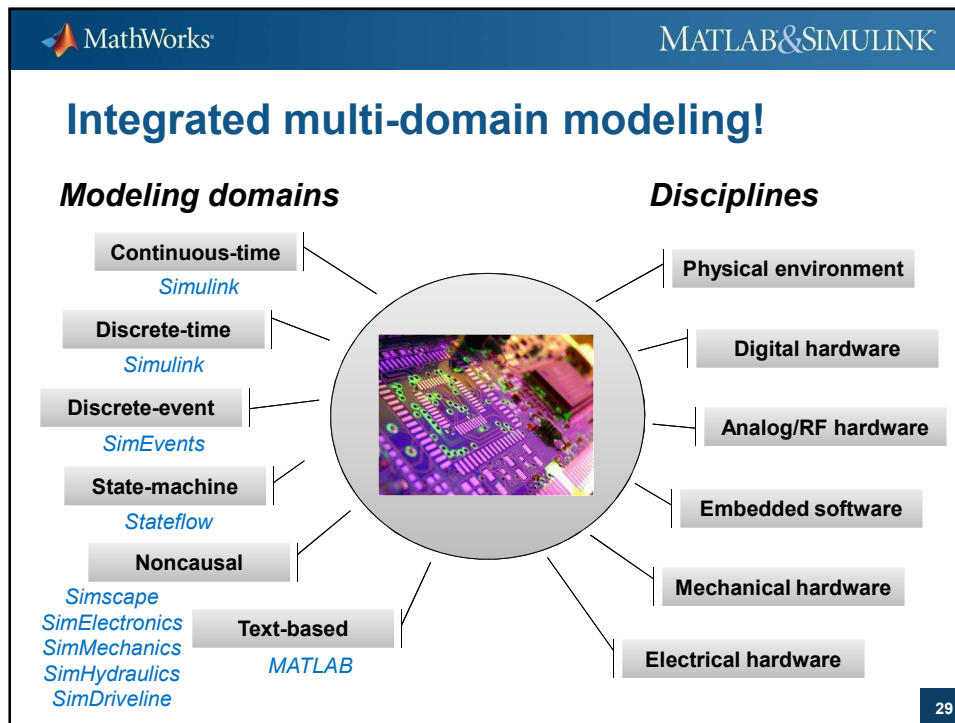
27

## Early and progressive system testing ... ?

### implementation



28



## Integration of computational semantics across many domains

- Once a model has been designed, **consistency** is more important than 'semantic correctness'
- This requires a precise definition of the computational semantics
- Multiple formalisms in **heterogeneous systems** amplify the importance of computational semantics
  - Block diagrams with discrete functionality (switches, reinitialization, inequalities, etc.)
  - State transition diagrams
- Can we define a common semantic domain?

highly  
sensitive

31

## Agenda

- Model-Based Design
- ▪ Computer Automated Multiparadigm Modeling
  - Modeling time as discrete events
  - A unifying semantic domain
  - A heterogeneous system example
- Conclusions

32



## What is a model anyway?



Jean Bézivin

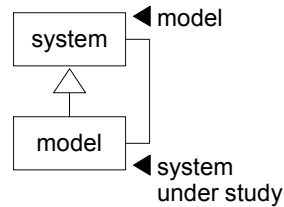
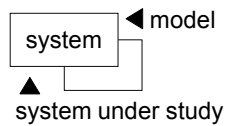
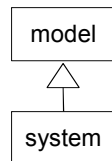


Jean-Marie Favre



Pieter J. Mosterman

“Everything is a model”   “Nothing is a model”   “Nothing is not a model”



33

## What is a model anyway?



Jean Bézivin



Jean-Marie Favre

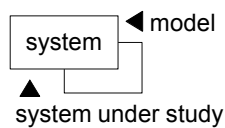
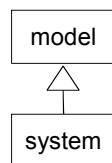


Pieter J. Mosterman



Hans Vangheluwe

“Everything is a model”   “Nothing is a model”   “Nothing is not a model”   “Model everything”



Computer Automated  
Multiparadigm Modeling  
(CAMPaM)

*In collaboration with Hans Vangheluwe, McGill University*

34

## Computer Automated Multiparadigm Modeling (CAMPaM)

- Initiated in 2000
  - Two special sessions at IEEE CACSD Symposium
  - Annual McGill Bellairs workshop since 2004
- Three elements of CAMPaM
  - Multi-(domain-specific)-formalism models
  - Metamodeling
  - Multiple levels of abstraction
- **Model everything!**
  - Reasoning vs. efficiency (denotation vs. operation)
  - Model model transformation (e.g., graph grammar)

35

## A domain-specific formalism: syntax versus semantics

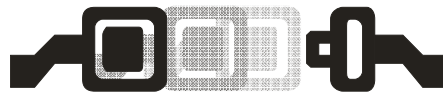
- Fasten seat belt syntax



- Fasten semantics ...



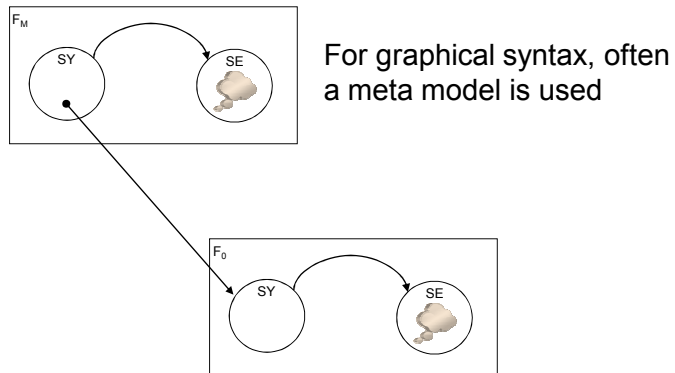
- ... or open semantics?



36

## The elements of a formalism

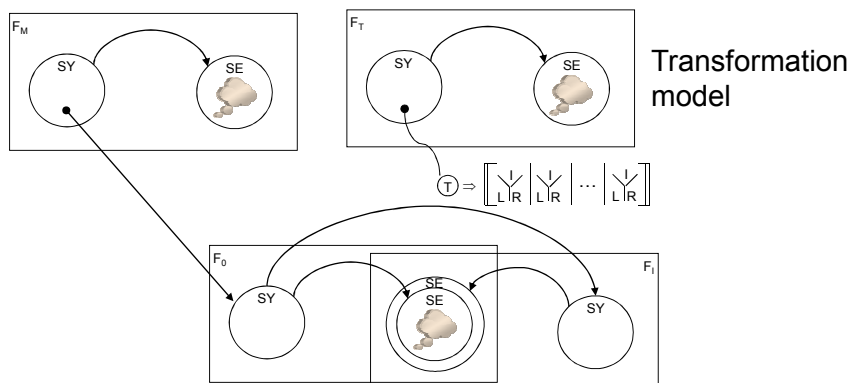
- A **syntax**, a **semantic domain**, and a **mapping**



37

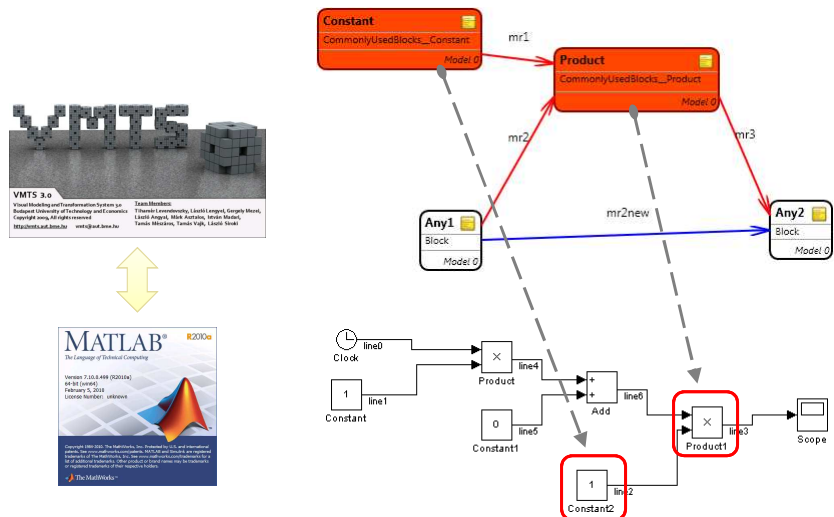
## Define semantics as a syntactic transformation—semantic anchoring

- Target semantic domain must be subsumed



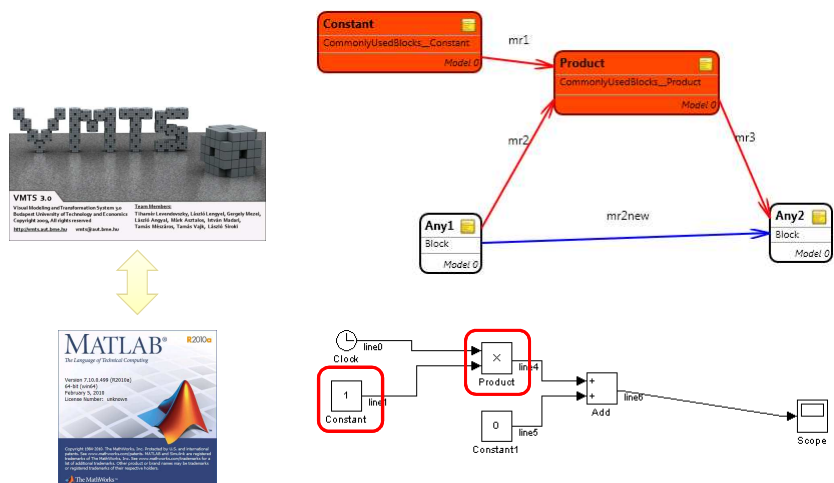
38

## Modeling a model transformation



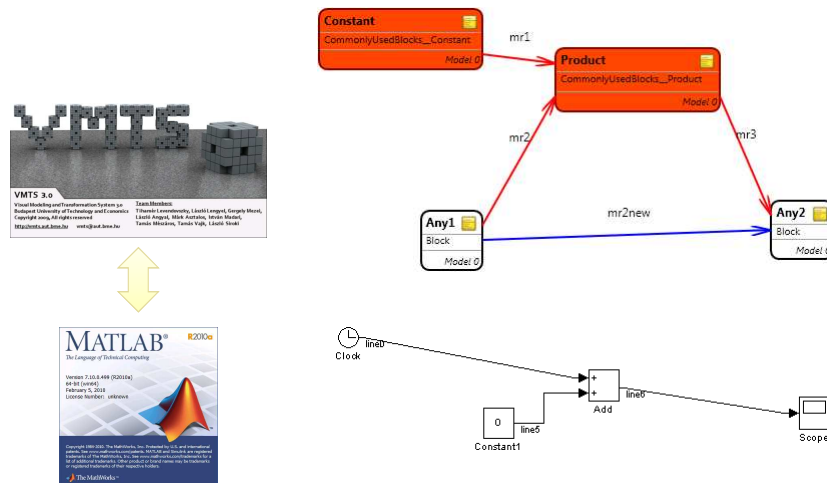
39

## Modeling a model transformation



40

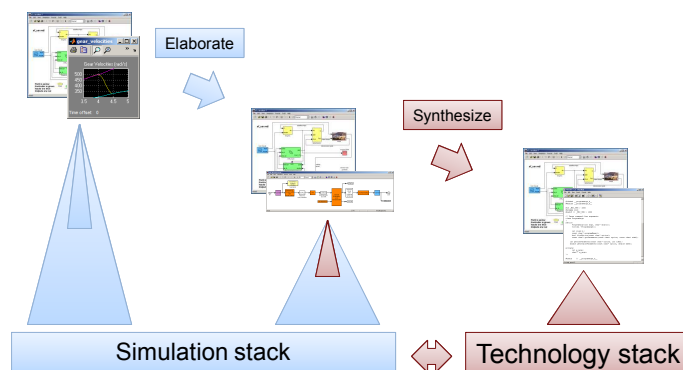
## Modeling a model transformation



41

## Now back to Model-Based Design

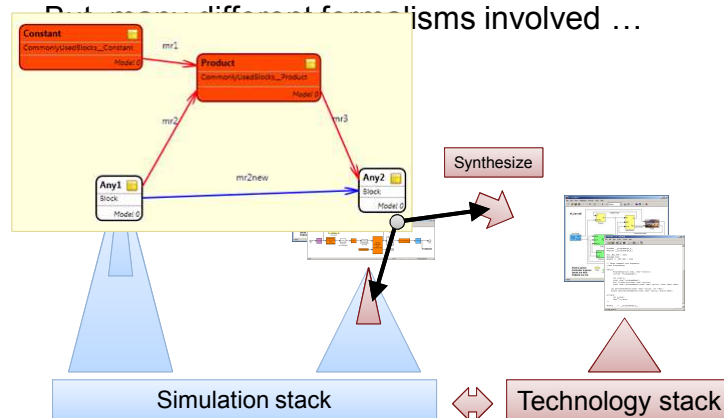
- Declarative and reusable transformations
  - But, many different formalisms involved ...



42

## Now back to Model-Based Design

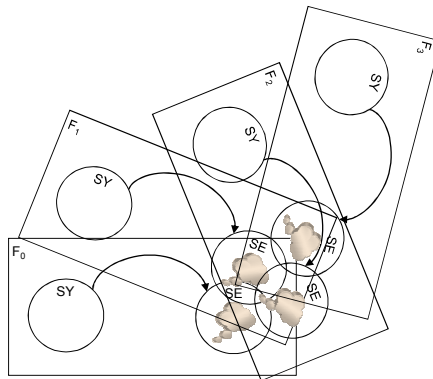
- Declarative and reusable transformations



43

## Multi-domain models comprise many formalisms ...

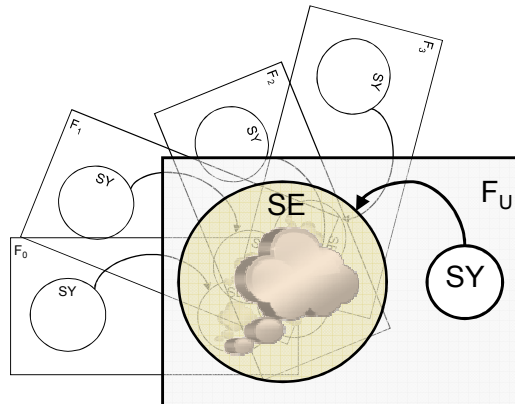
- Can we develop a unifying semantic domain?



44

## Multi-domain models comprise many formalisms ...

- Can we develop a unifying semantic domain?



45

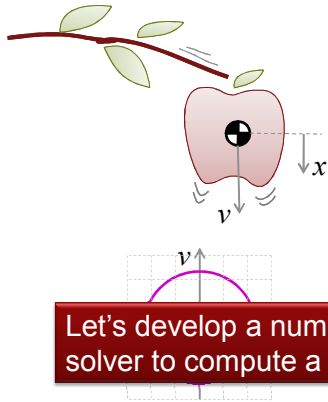
## Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Modeling time as discrete events
- A unifying semantic domain
- A heterogeneous system example
- Conclusions

46

## Modeling a physical system

From first principles ...



Let's develop a numerical solver to compute a solution ...

Hooke's Law:  $F = -\frac{x - x_0}{C}$

Newton's Second:  $F = ma$

A bit of calculus:  $a(t) = \frac{dv(t)}{dt}$

$$v(t) = \frac{dx(t)}{dt}$$

An ideal oscillator:  $v(t) = \frac{dx(t)}{dt}$

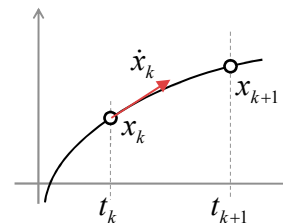
$$m \frac{dv(t)}{dt} = -\frac{x(t) - x_0}{C}$$

47

## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$



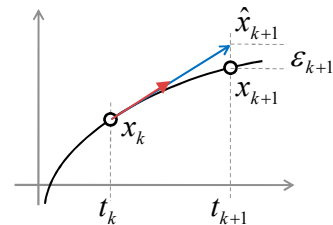
48



## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$



49

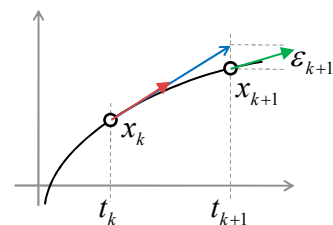
## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

*Trapezoidal:* average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$



50

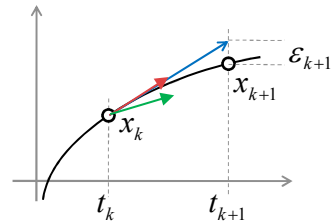
## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

*Trapezoidal:* average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$



51

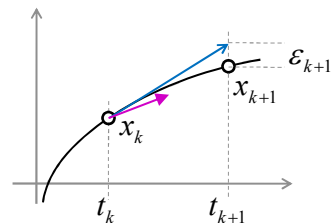
## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k)h_k$$

*Trapezoidal:* average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$



52

## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

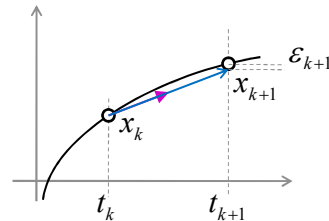
$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k) h_k$$

*Trapezoidal:* average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$

Taylor series expansion for error analysis

$$x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!} h_k + \frac{\ddot{x}(t_k)}{2!} h_k^2 + O(h_k^3)$$



53

## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k) h_k$$

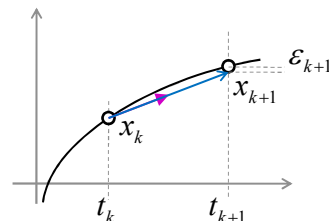
*Trapezoidal:* average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$

Taylor series expansion for error analysis

$$x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!} h_k + \frac{\ddot{x}(t_k)}{2!} h_k^2 + O(h_k^3)$$

$\epsilon_e(t_{k+1})$



When  $x(t)$  changes little,  $h_k$  can be large!

54

## Numerical integration

*Euler:* step  $h$  in time along  $\dot{x} = f(x, t)$

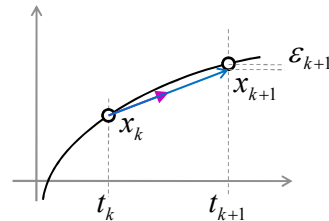
$$\hat{x}_e(t_{k+1}) = x(t_k) + \dot{x}(t_k) h_k$$

*Trapezoidal:* average the end points

$$\hat{x}_t(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_{k+1}) + \dot{x}(t_k)}{2} h_k$$

Taylor series expansion for error analysis

$$x(t_{k+1}) = x(t_k) + \frac{\dot{x}(t_k)}{1!} h_k + \underbrace{\frac{\ddot{x}(t_k)}{2!} h_k^2}_{\varepsilon_e(t_{k+1})} + \underbrace{O(h_k^3)}_{\varepsilon_t(t_{k+1})}$$

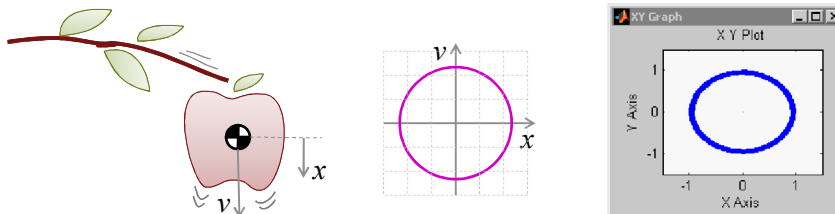


Change step size based on estimate:  $\hat{x}_e(t_{k+1}) - \hat{x}_t(t_{k+1}) \approx \frac{\ddot{x}(t_k)}{2!} h_k^2$

55

## Sophisticated solver ... ?

- Let's compute a solution to the ideal oscillator

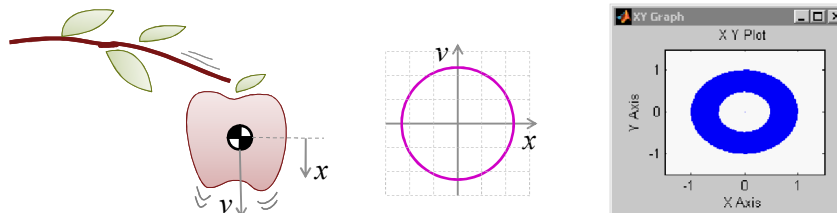


- We can make the error small ... but only locally!

56

## Sophisticated solver ... ?

- Let's compute a solution to the ideal oscillator

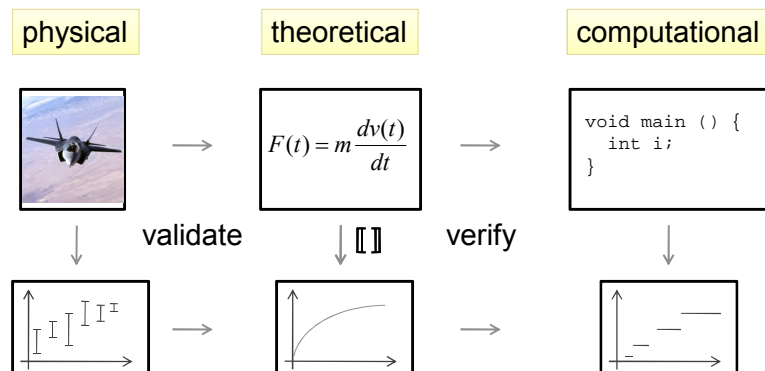


- We can make the error small ... but only locally!
- It accumulates for 'long time' behavior
- So, ... *how come the JSF flies?!*

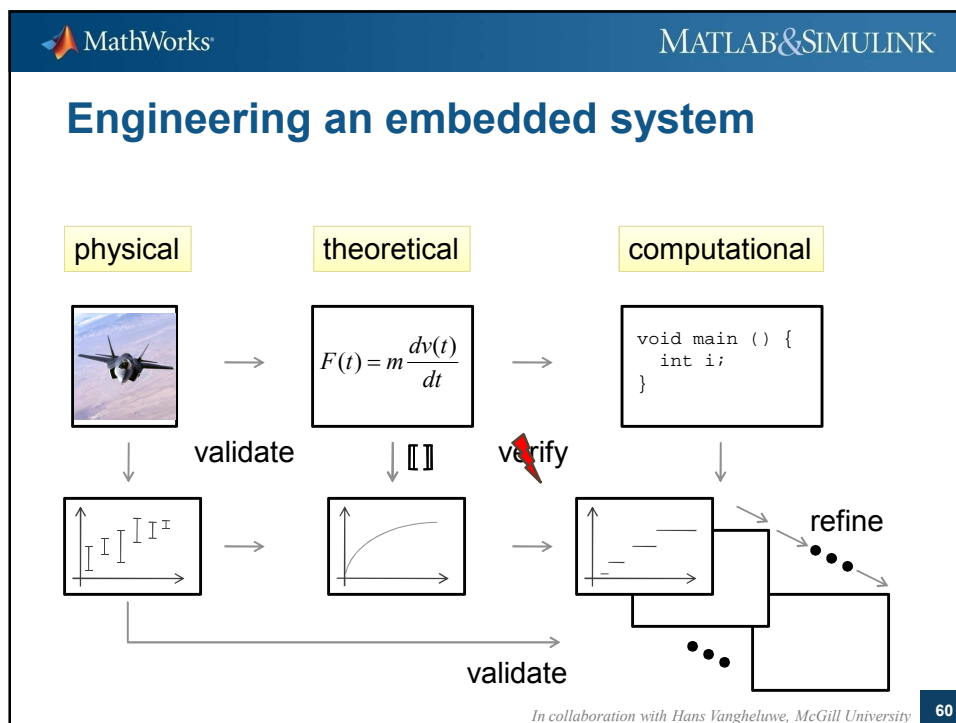
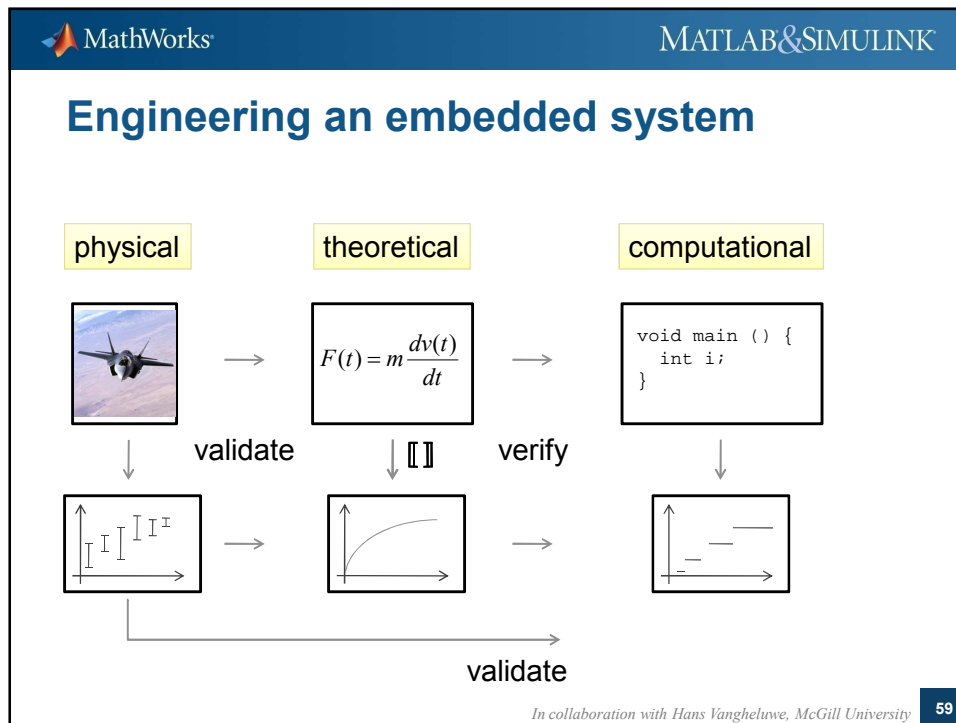


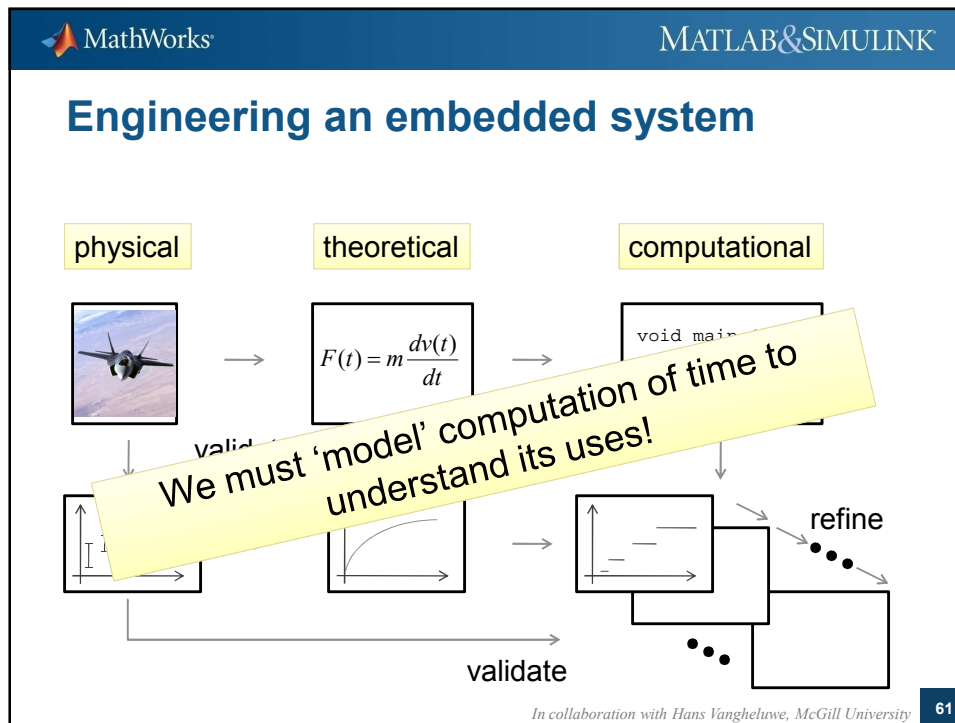
57

## Engineering an embedded system



58





MathWorks MATLAB & SIMULINK

## Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Modeling time as discrete events
- ▪ A unifying semantic domain
- A heterogeneous system example
- Conclusions

62

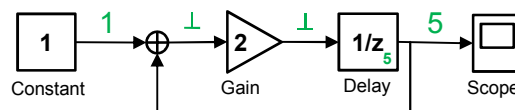
## A declarative formalism with fix-point semantics

A LATTICE-THEORETICAL FIXPOINT THEOREM  
AND ITS APPLICATIONS

ALFRED TARSKI

*Pacific J. Math.* 5 (1955), 285 - 309

- Repeated application of a monotonically increasing partial function converges to a fixed point



63

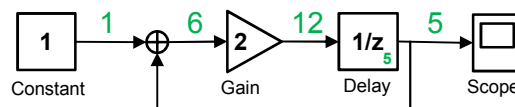
## A declarative formalism with fix-point semantics

A LATTICE-THEORETICAL FIXPOINT THEOREM  
AND ITS APPLICATIONS

ALFRED TARSKI

*Pacific J. Math.* 5 (1955), 285 - 309

- Repeated application of a monotonically increasing partial function converges to a fixed point



64

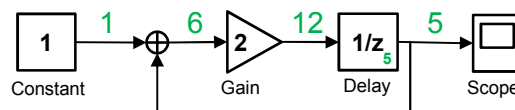


## A declarative formalism with fix-point semantics

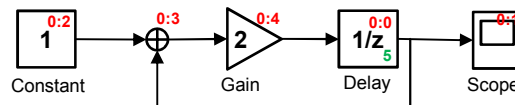
A LATTICE-THEORETICAL FIXPOINT THEOREM  
AND ITS APPLICATIONS  
ALFRED TARSKI

*Pacific J. Math.* 5 (1955), 285 - 309

- Repeated application of a monotonically increasing partial function converges to a fixed point



- One implementation is a data dependency schedule



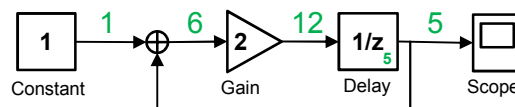
65

## A declarative formalism with fix-point semantics

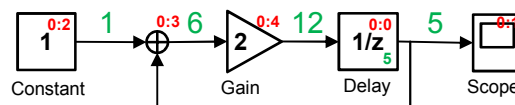
A LATTICE-THEORETICAL FIXPOINT THEOREM  
AND ITS APPLICATIONS  
ALFRED TARSKI

*Pacific J. Math.* 5 (1955), 285 - 309

- Repeated application of a monotonically increasing partial function converges to a fixed point



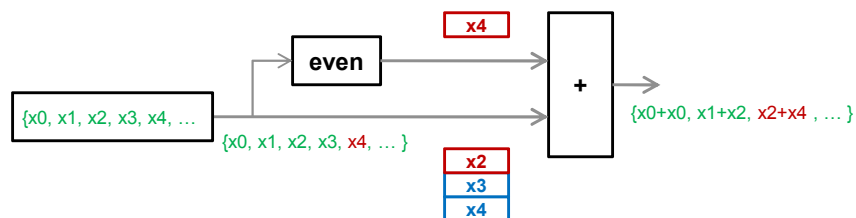
- One implementation is a data dependency schedule



66

## Multiple rates; a potential problem ...

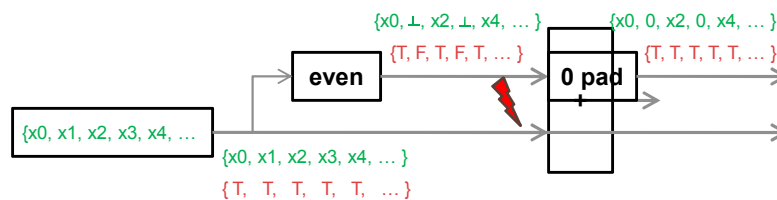
- Streams are only practical if we can limit the stream entries being accessed
- Not this:



67

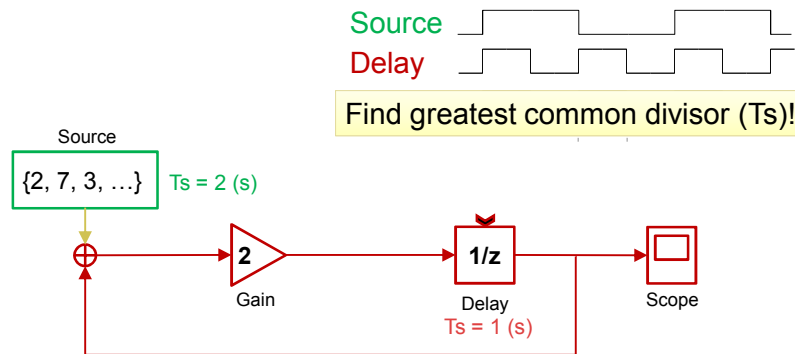
## Clock calculus to detect

- Require compatible **clocks**: the synchronous assumption
- Match against base clock



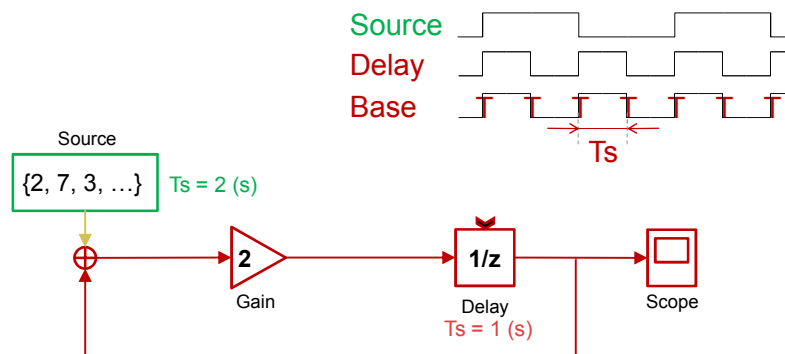
68

## A multi-rate system example



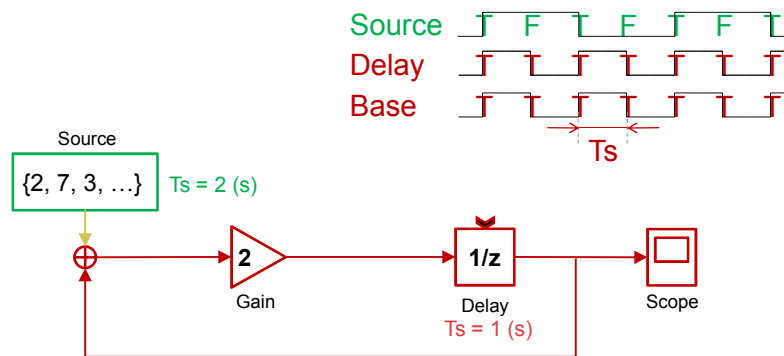
69

## A multi-rate system example



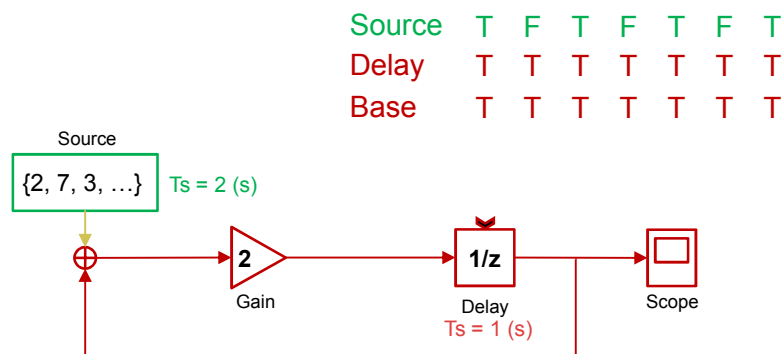
70

## A multi-rate system example



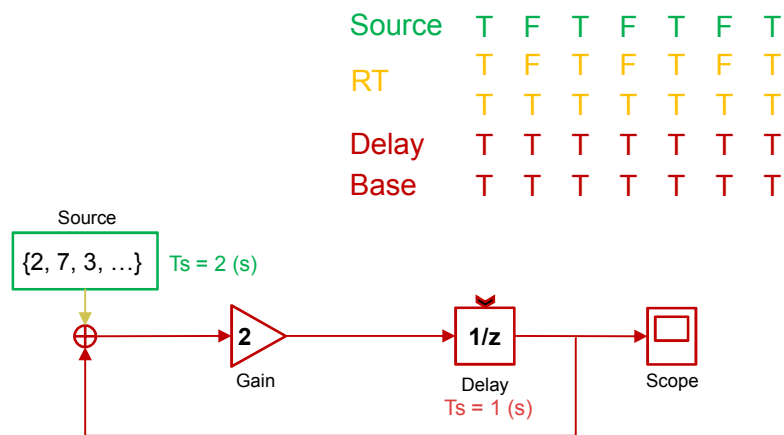
71

## A multi-rate system example



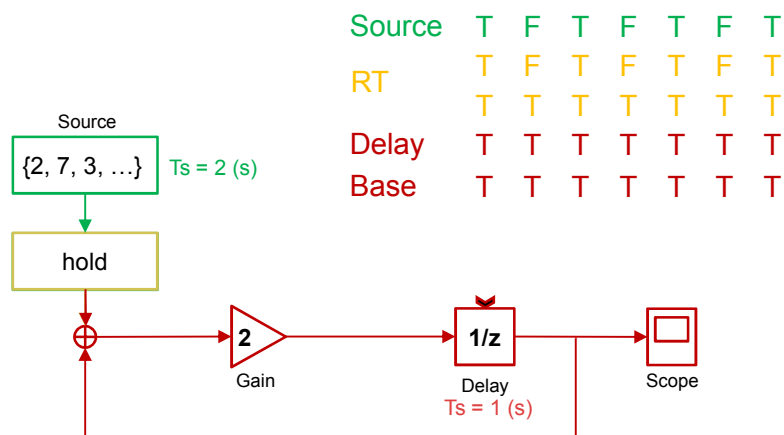
72

## A multi-rate system example



73

## A multi-rate system example



74

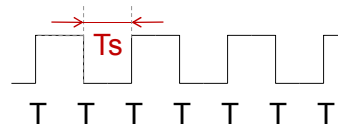
## Can we use this framework to define a solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations

75

## Can we use this framework to define a solver?

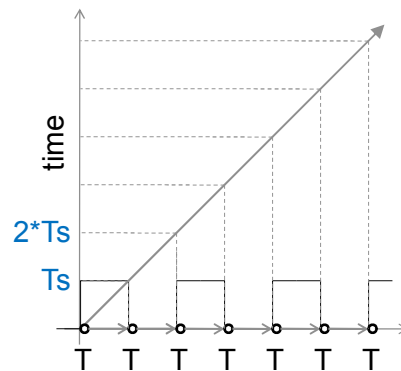
- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



76

## Can we use this framework to define a variable-step solver?

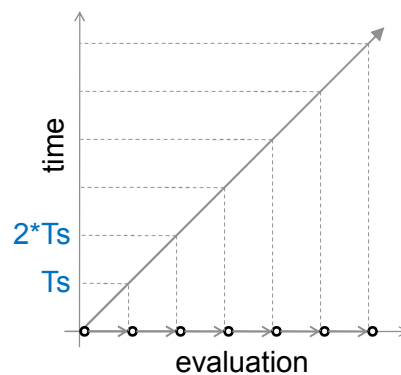
- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



77

## Can we use this framework to define a variable-step solver?

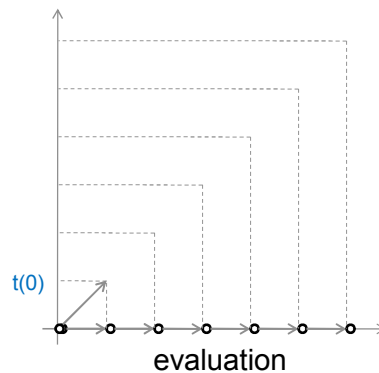
- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



78

## Can we use this framework to define a variable-step solver?

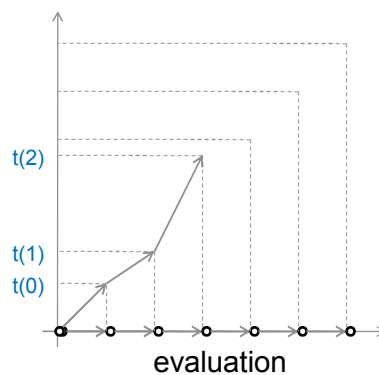
- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations



79

## Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable

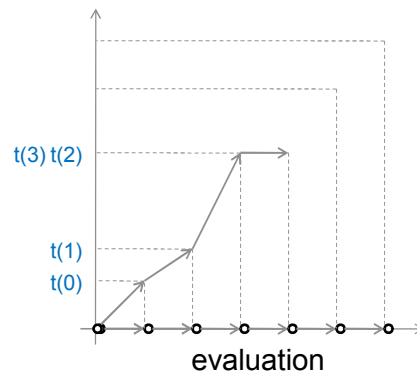


80



## Can we use this framework to define a variable-step solver?

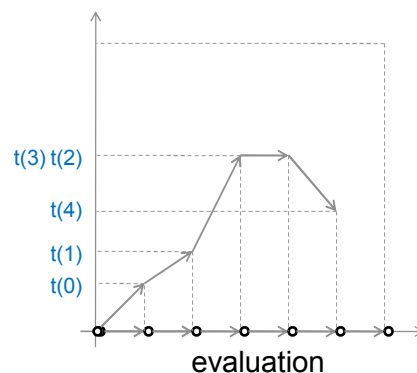
- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable
  - Step may be 0



81

## Can we use this framework to define a variable-step solver?

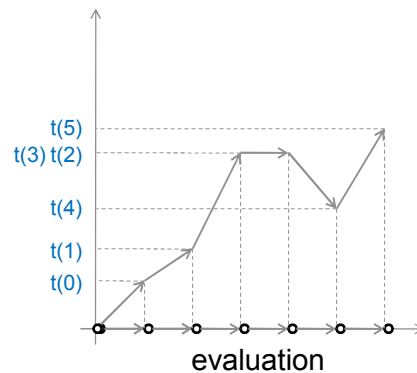
- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable
  - Step may be 0
  - Step may be negative
    - Time may recede



82

## Can we use this framework to define a variable-step solver?

- Separate
  - Time (explicit)
  - Evaluations (ordered)
- Time as a function of evaluations
  - Step is variable
  - Step may be 0
  - Step may be negative
    - Time may recede



83

## A stream based functional solver

### Euler integration

$$y_e(e) = \begin{cases} \sum_{i=1}^e \overset{\text{increment}}{u(i)h(i)} - \overset{\text{previous increment}}{u(i-2)h(i-2)p(i)} & \text{if } \text{odd}(e) \\ y_e(e-1) & \text{otherwise} \end{cases}$$

### Trapezoidal integration

$$y_t(e) = \sum_{i=1}^e \overset{\text{increment}}{\frac{(u(i-1) + u(i))h(i-1)}{2}} - \overset{\text{previous increment}}{\frac{(u(i-3) + u(i-2))h(i-3)}{2}} p(i-1)$$

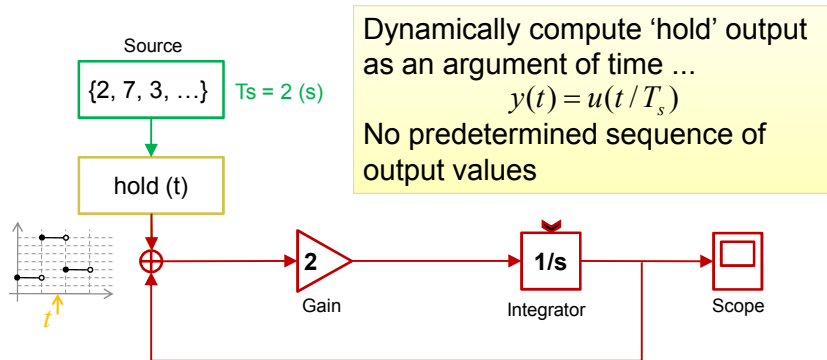
### Error computation

$$d(e) = \frac{(u(e-3) + u(e-2))h(e-3)}{2} - u(e-2)h(e-2) < \text{tol}$$

84

## Rate transition a function of time

Now we can create a variable step solver inside 1/s that maps onto the synchronous paradigm



85

## Agenda

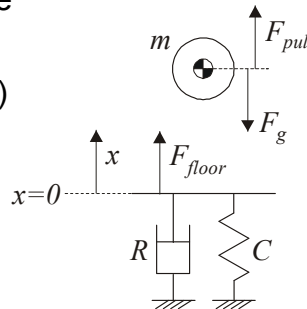
- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Modeling time as discrete events
- A unifying semantic domain
- ▪ A heterogeneous system example
- Conclusions

86

## Unifying formalisms with different semantics

- Newton's and Hooke's Laws
  - Differential equations as before
- Control behavior
  - Sampled data (periodic  $T_s=0.5$ )
- Contact behavior
  - Discontinuous changes ...

$$F_{pull}(k) = \begin{cases} 20 & \text{if } k = 0 \\ 10 & \text{if } k = 1 \\ 0 & \text{else} \end{cases}$$



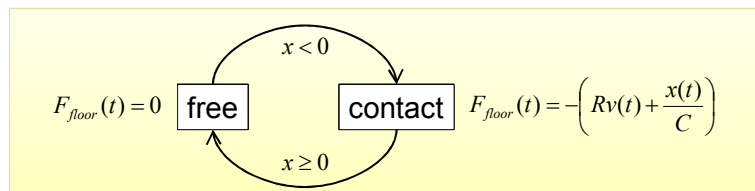
87

## Modeling the contact behavior

- Simultaneous inequalities

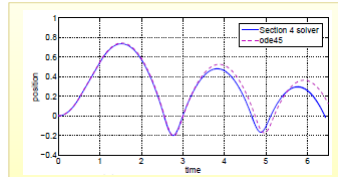
$$F_{floor}(t) = \begin{cases} -\left(Rv(t) + \frac{x(t)}{C}\right) & \text{if } x(t) < 0 \\ 0 & \text{otherwise} \end{cases}$$

- Finite state machine

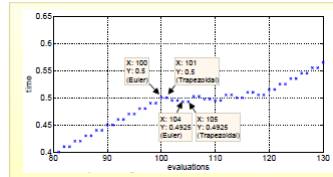


88

## Computational simulation



Position vs. time



Time vs. evaluations (detail)

### Simultaneous inequalities

Eval	Time	Position	Velocity	$F_{floor}$	Error
532	2.5450	0.0037	-1.4381	0	
533	2.5450	-0.0035	-1.4381	8.5888	0
534	2.5550	-0.0107	-1.4398	11.4717	
535	2.5550	-0.0179	-1.4377	14.3430	0.0021
536	2.5500	-0.0035	-1.4348	8.5700	
537	2.5500	-0.0107	-1.4369	11.4555	0.0021
538	2.5475	0.0001	-1.4375	0	
539	2.5475	-0.0071	-1.4395	10.0333	0.0020
540	2.5462	0.0019	-1.4380	0	
541	2.5462	-0.0053	-1.4389	9.3125	0.0009
542	2.5456	0.0028	-1.4381	0	
543	2.5456	-0.0044	-1.4385	8.9508	0.0004

### Finite state machine

Eval	Time	Position	Velocity	$F_{floor}$	Error	$\xi_{con}$
532	2.5450	0.0037	-1.4381	0	0	0
533	2.5450	-0.0035	-1.4381	0	0	0
534	2.5550	-0.0107	-1.4521	11.5331		1
535	2.5550	-0.0179	-1.4438	14.3980	0.0082	1
536	2.5500	-0.0035	-1.4348	8.5820		1
537	2.5500	-0.0107	-1.4369	11.4614	0.0021	1
538	2.5475	0.0001	-1.4375	7.1446		1
539	2.5475	-0.0071	-1.4382	0	0.0008	0
540	2.5462	0.0019	-1.4308	6.4386		1
541	2.5462	-0.0053	-1.4392	0	0.0006	0

89

## Characteristics of the semantic domain

- Declarative
  - Purely functional (no side effects)
- Ordered evaluations
- Untimed
  - Time as explicit function,  $t(e)$
  - Time is not strictly increasing
- Broadly applicable to dynamic systems
  - Differential equations, difference equations, discrete events

90

## Agenda

- Model-Based Design
- Computer Automated Multiparadigm Modeling
- Modeling time as discrete events
- A unifying semantic domain
- A heterogeneous system example
- ▪ Conclusions

91

## Conclusions

- Transformation technology in Model-Based Design
  - Vertical; the simulation stack
  - Horizontal; the elaboration and synthesis
- Model the transformations
  - Denotational vs. operational
- Nonmonotonic tagged synchronous formulation
  - Unifying semantic domain for multi-domain models

92

## Precise computational semantics as a foundation for ...

- Integrated multi-domain modeling
- End-to-end system analysis
- Support for design automation
  - Multi-view/abstraction models (with approximation)
  - Design of new languages (e.g., for concurrency, heterogeneity)
  - Compositionality and composability
- Reference

Pieter J. Mosterman, Justyna Zander, Gregoire Hamon, and Ben Denckla, "Towards Computational Hybrid System Semantics for Time-Based Block Diagrams," in *3rd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'09)*, A. Giua, C. Mahulea, M. Silva, and J. Zaytoon (eds.), pp. 376-385, Zaragoza, Spain, September 16-18, 2009, plenary paper.

93

## Acknowledgments

Justyna Zander  
Harvard University  
Fraunhofer Institute FOKUS, Berlin

Gregoire Hamon  
MathWorks

Ben Denckla  
Independent Thinker

Hans Vangheluwe  
University of Antwerp  
McGill University

Many thanks for their continuing collaboration!

94

