Why We Model: Using MBD Effectively in Critical Domains

Mike Whalen

Program Director, UMSEC

University of Minnesota

Acknowledgements

- Rockwell Collins (Darren Cofer, Andrew Gacek, Steven Miller, Lucas Wagner)
- UPenn: (Insup Lee, Oleg Sokolsky)
- UMN (Mats P. E. Heimdahl)
- NASA Langley (Ricky Butler)
- Lockheed Martin (Walter Storm, Greg Tallant, Peter Stanfill)

Note: all incorrect or controversial opinions are mine only ☺

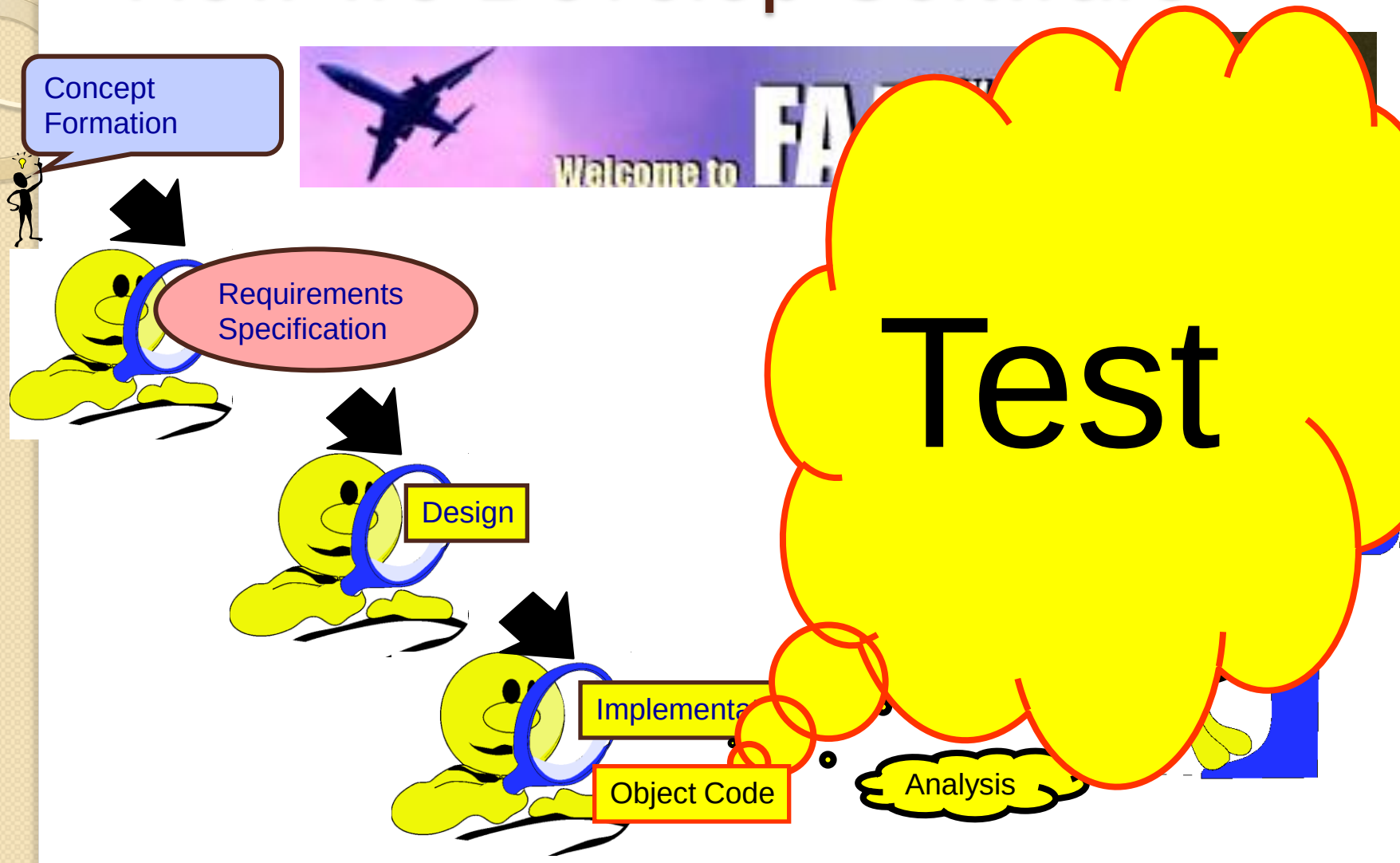
Outline of Presentation

Introduction

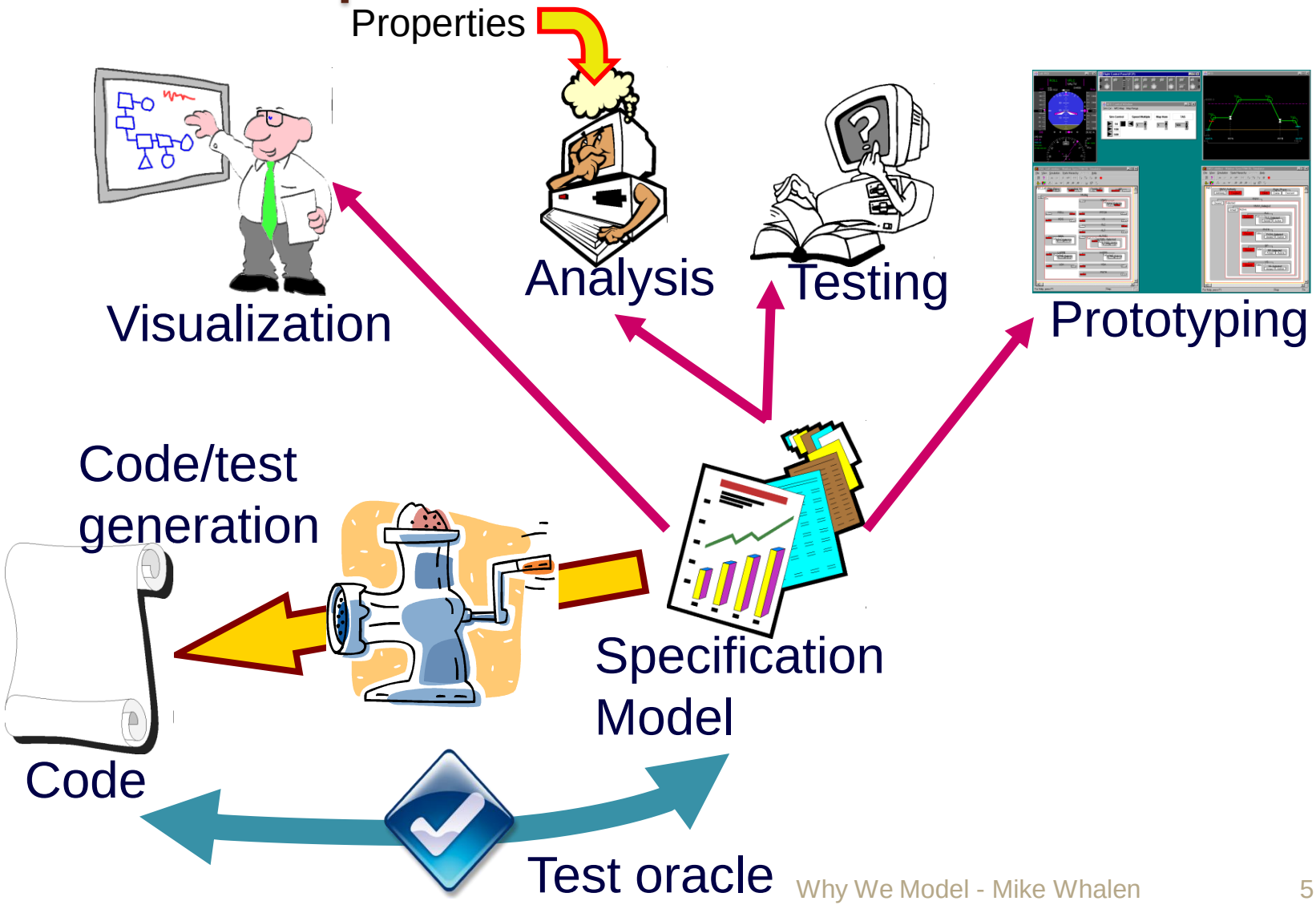
Why use Model-Based Development?
- Requirements
- Design
- Implementation: Code Generation
- Verification and Validation

Pitfalls

How we Develop Software

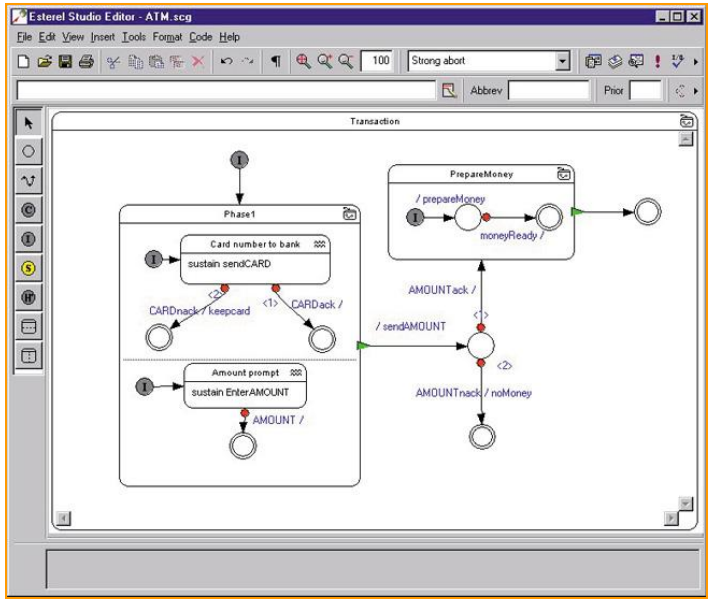


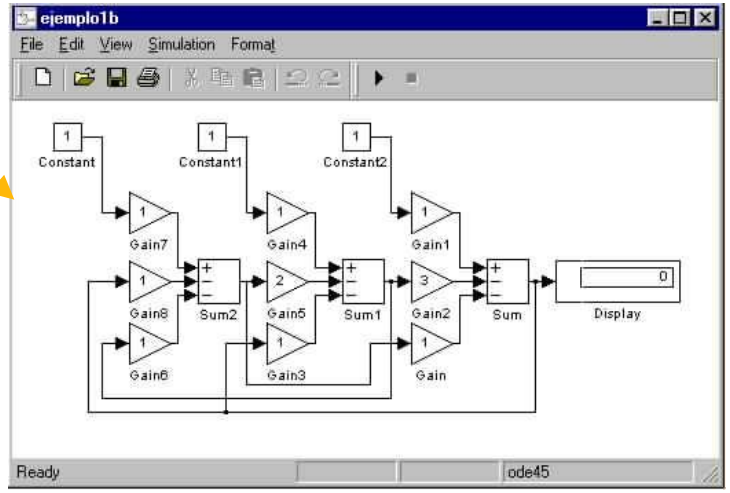
What is Model-Based Development?



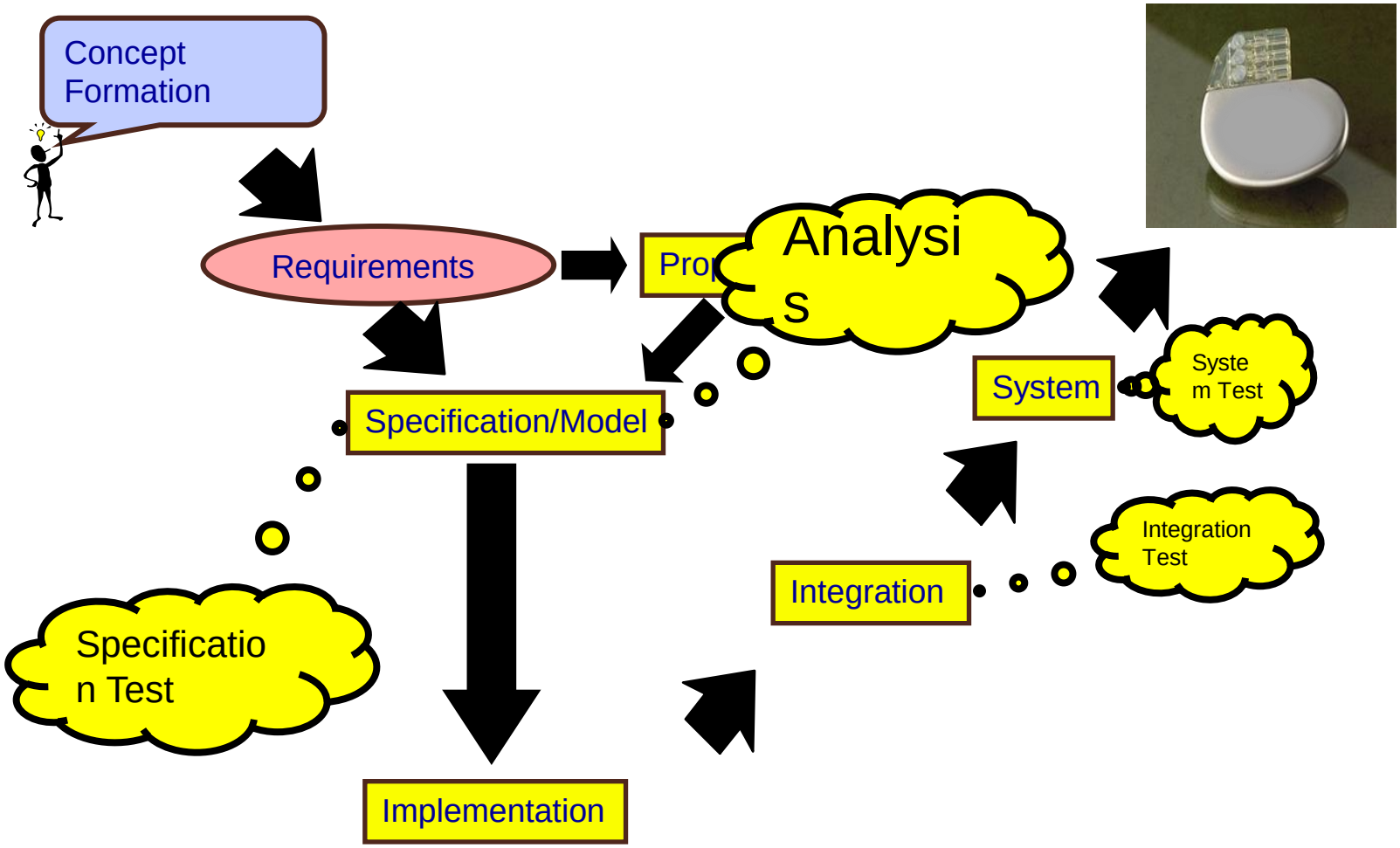
Model-Based Development Tools

- Esterel Studio and SCADE Studio from Esterel Technologies
- Rhapsody from I-Logix
- Simulink and Stateflow from Mathworks Inc.
- Rose Real-Time from Rational
- I will focus on Statecharts and Dataflow notations.





How we **Will** Develop Software (in theory)



Model-Based Development Examples

Company	*Product*	*Tools*	*Specified & Autocoded*	*Benefits Claimed*
Airbus	A340	SCADE With Code Generator	• 70% Fly-by-wire Controls • 70% Automatic Flight Controls • 50% Display Computer • 40% Warning & Maint Computer	• 20X Reduction in Errors • Reduced Time to Market
Eurocopter	EC-155/135 Autopilot	SCADE With Code Generator	• 90 % of Autopilot	• 50% Reduction in Cycle Time
GE & Lockheed Martin	FADEDC Engine Controls	ADI Beacon	• Not Stated	• Reduction in Errors • 50% Reduction in Cycle Time • Decreased Cost
Schneider Electric	Nuclear Power Plant Safety Control	SCADE With Code Generator	• 200,000 SLOC Auto Generated from 1,200 Design Views	• 8X Reduction in Errors while Complexity Increased 4x
US Spaceware	DCX Rocket	MATRIXx	• Not Stated	• 50-75% Reduction in Cost • Reduced Schedule & Risk
PSA	Electrical Management System	SCADE With Code Generator	• 50% SLOC Auto Generated	• 60% Reduction in Cycle Time • 5X Reduction in Errors
CSEE Transport	Subway Signaling System	SCADE With Code Generator	• 80,000 C SLOC Auto Generated	• Improved Productivity from 20 to 300 SLOC/day
Honeywell Commercial Aviation Systems	Primus Epic Flight Control System	MATLAB Simulink	• 60% Automatic Flight Controls	• 5X Increase in Productivity • No Coding Errors • Received FAA Certification

Does Model-Based Development Scale?

Airbus A380

Length	239 ft 6 in
Wingspan	261 ft 10 in
Maximum Takeoff Weight	1,235,000 lbs
Passengers	Up to 840
Range	9,383 miles

Systems Developed Using MBD

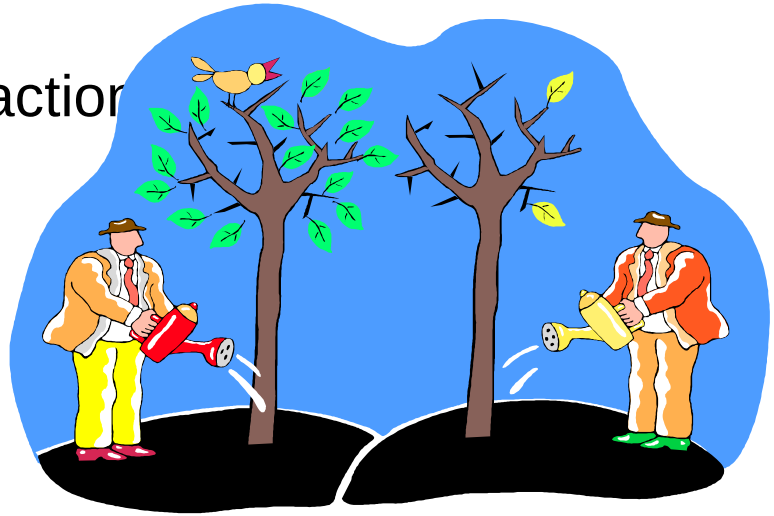
- Flight Control
- Auto Pilot
- Fight Warning
- Cockpit Display
- Fuel Management
- Landing Gear
- Braking
- Steering
- Anti-Icing
- Electrical Load Management

…But it is not all roses

- Many MBD projects fail to meet their original goals of cost, productivity
  - These tend not to get as much publicity!
- Clear eyed understanding of *why* you model and *what you expect* is necessary

A Personal Anecdote

- Part of two large projects using Model-Based Development
  - Same company, similar quality developers
  - One great success
    - Significant cost reductions
    - Improvement in quality
    - Excellent customer satisfaction
  - One great failure
    - Large cost overruns
    - Models considered less useful than code
    - Group abandoned MBD



Outline of Presentation

Introduction

Why use Model-Based Development?
- Requirements
- Design
- Implementation: Code Generation
- Verification and Validation

Pitfalls

What are your models *for?*

- Possible to use MBD for many different purposes:
- Requirements
- Design
- Simulation
- Visualization
- Testing
  - Test Generation
  - Test Oracle
- Formal Verification
- Code Generation
  - Complete implementation
  - Code skeleton
- Prototyping
- Communication with Customer

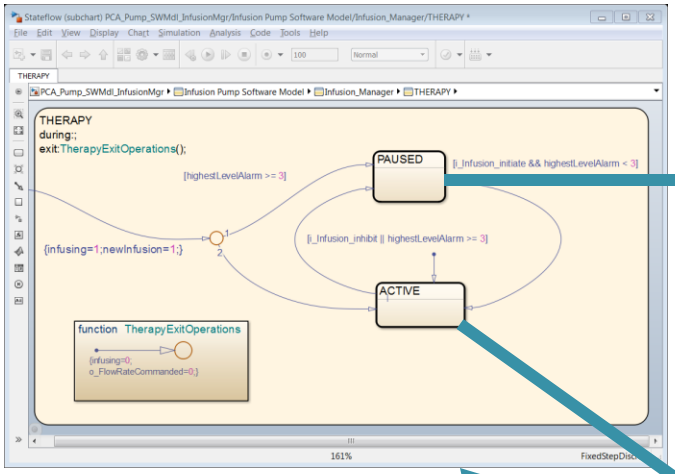
You must understand, **up front**, what you expect to do with models in order to successfully adopt MBD

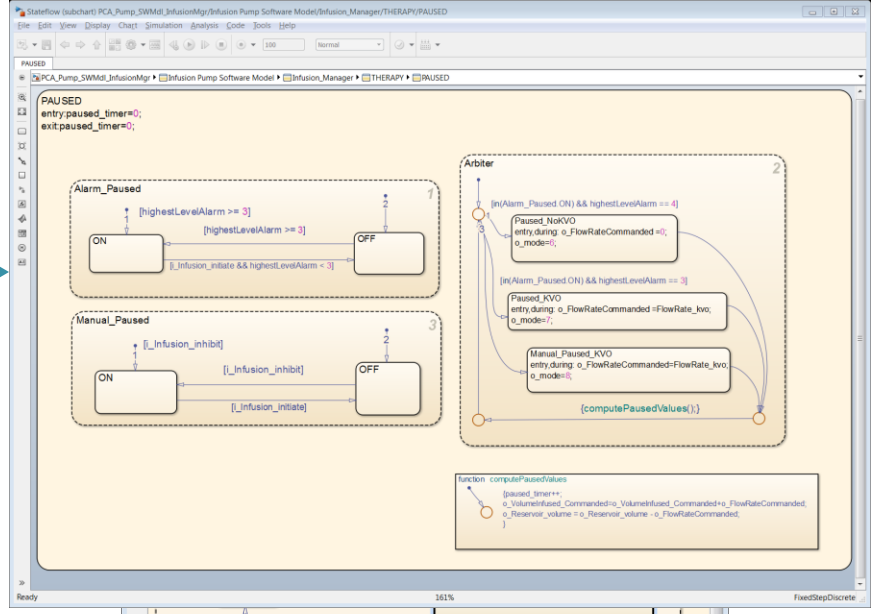
Major opportunity for improvement in V&V

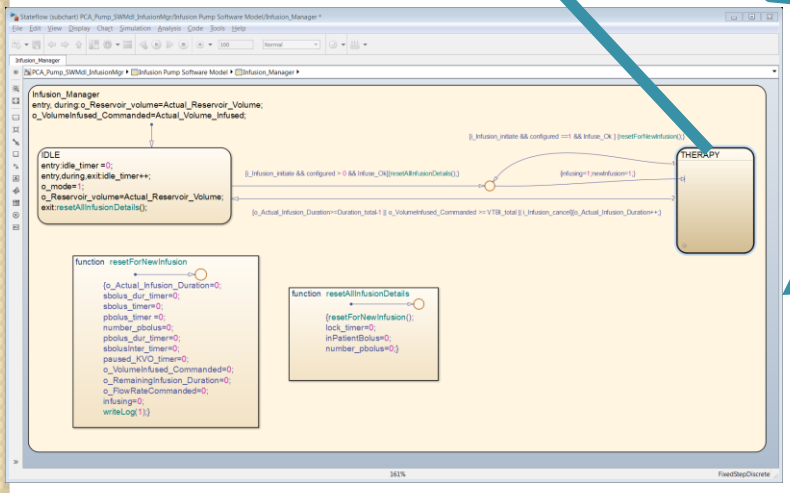
MBD Models as Requirements

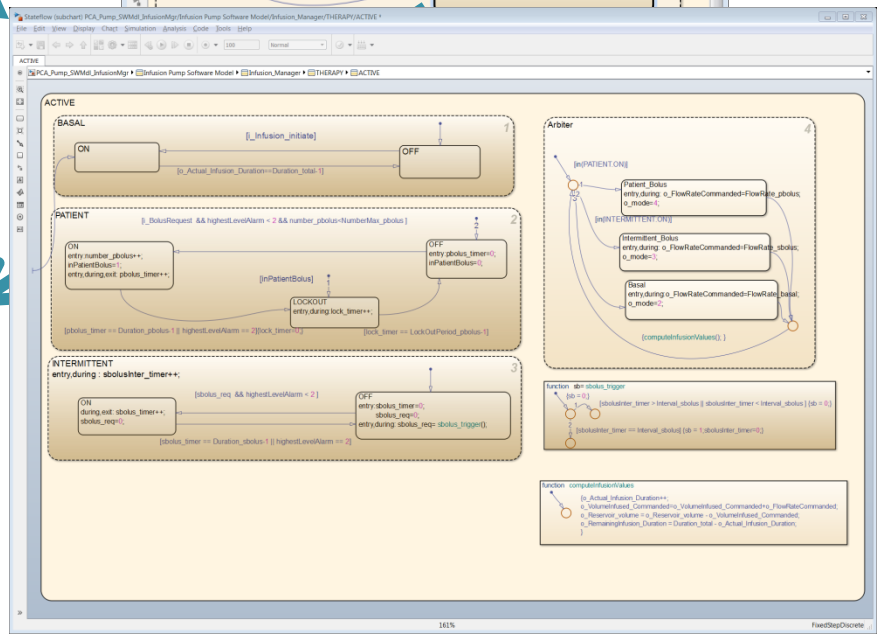
- Are MBD models requirements?

- Notations in this talk are executable; good at describing *how* system works









- Lots of design detail
- Difficult to see “full system” behavior.
- Straightforward to generate code

Outline of Presentation

Introduction

Why use Model-Based Development?

- Requirements
- Design
- Implementation: Code Generation
- Verification and Validation

Pitfalls

The Most Important Issue for Successful Adoption of MBD

Do the Domain-Specific Notations provide a natural representation for your problem?

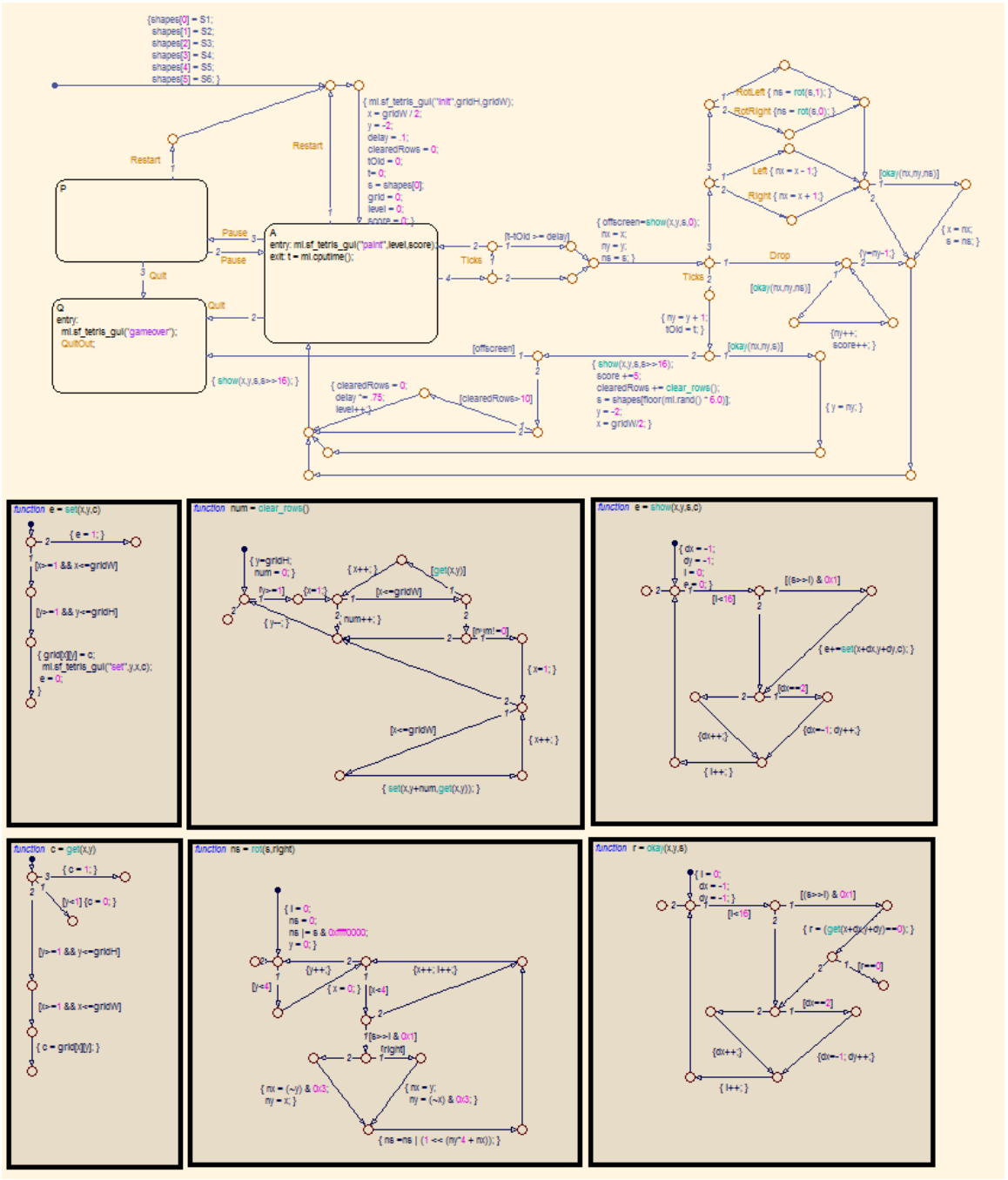
- Block diagrams are *very natural* for control problems
- Statecharts are *very natural* for description of system modes & mode transitions
- Both block diagrams and statecharts are *very unnatural* for representing complex data structures
- Neither notation naturally supports iteration or recursion
  - It can be “faked”, but not well

Just…No

Stateflow model of Tetris game (included in the Stateflow Demo models from the Mathworks!).

Diagram is essentially a control-flow graph of a program that implements tetris.

Much harder to read and modify than an equivalent program.



Tools Matter

- Often notations are much more cumbersome to use than text
  - No diff / merge capabilities
  - Adding information requires many clicks
- Expressible != Easy
- Anecdote: Simulink vs. SCADE at Rockwell Collins in 2006
  - SCADE had formal pedigree, strong analysis
    - But tools kept crashing on our Windows boxes
  - Simulink had better tools and better salespeople

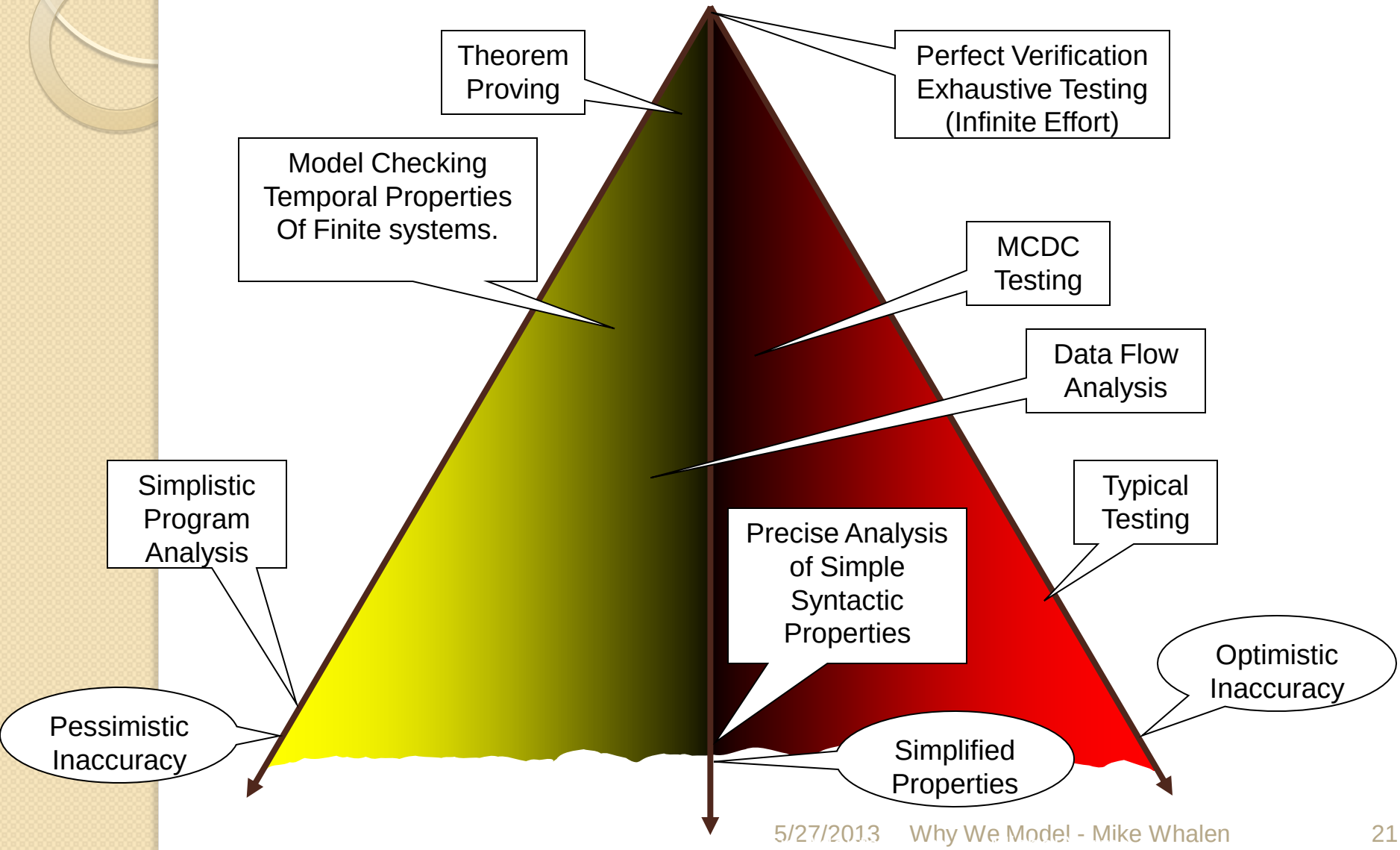
Outline of Presentation

Introduction

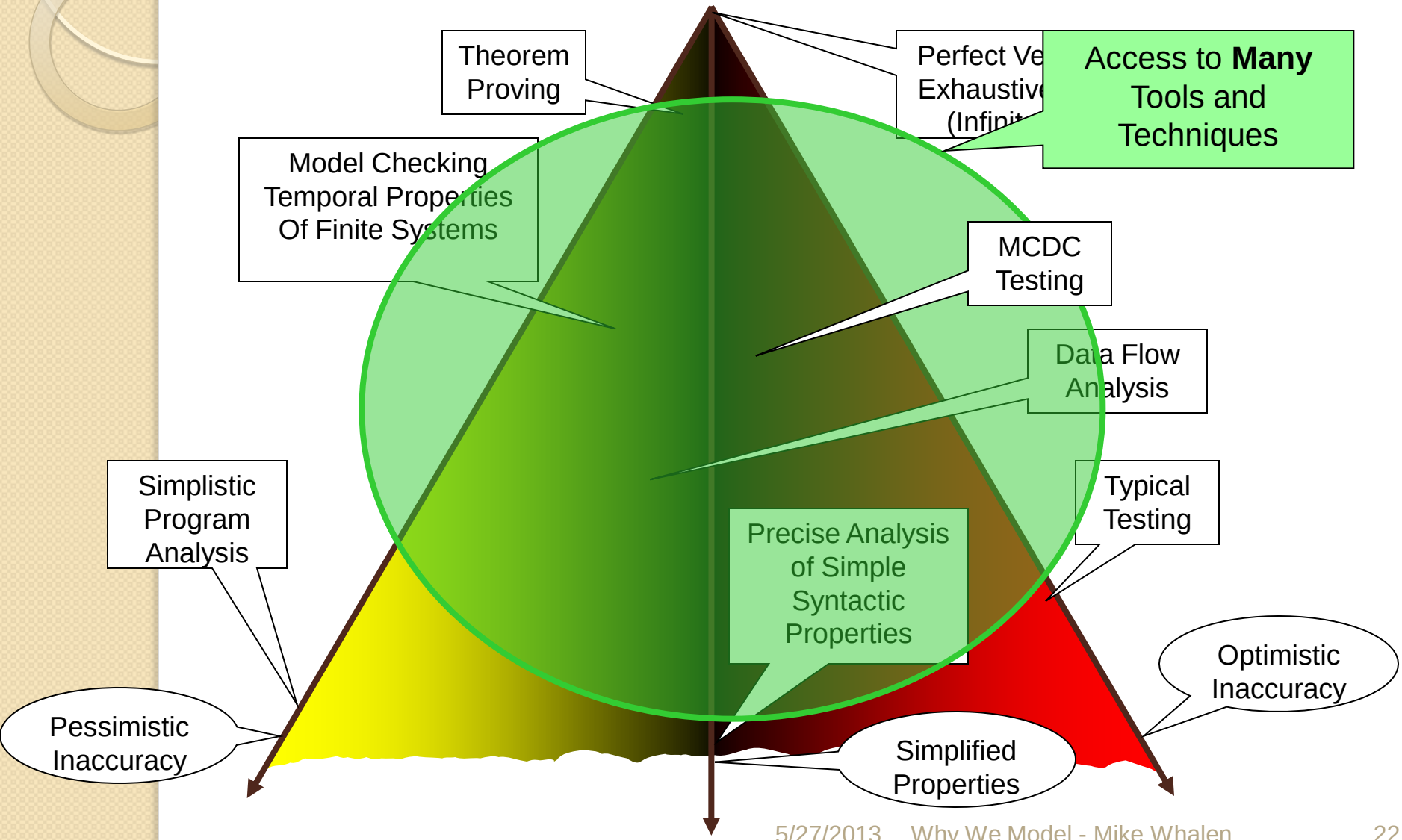
Why use Model-Based Development?
- Requirements
- Design
- Implementation: Code Generation
- Verification and Validation

Pitfalls

Analysis Pyramid



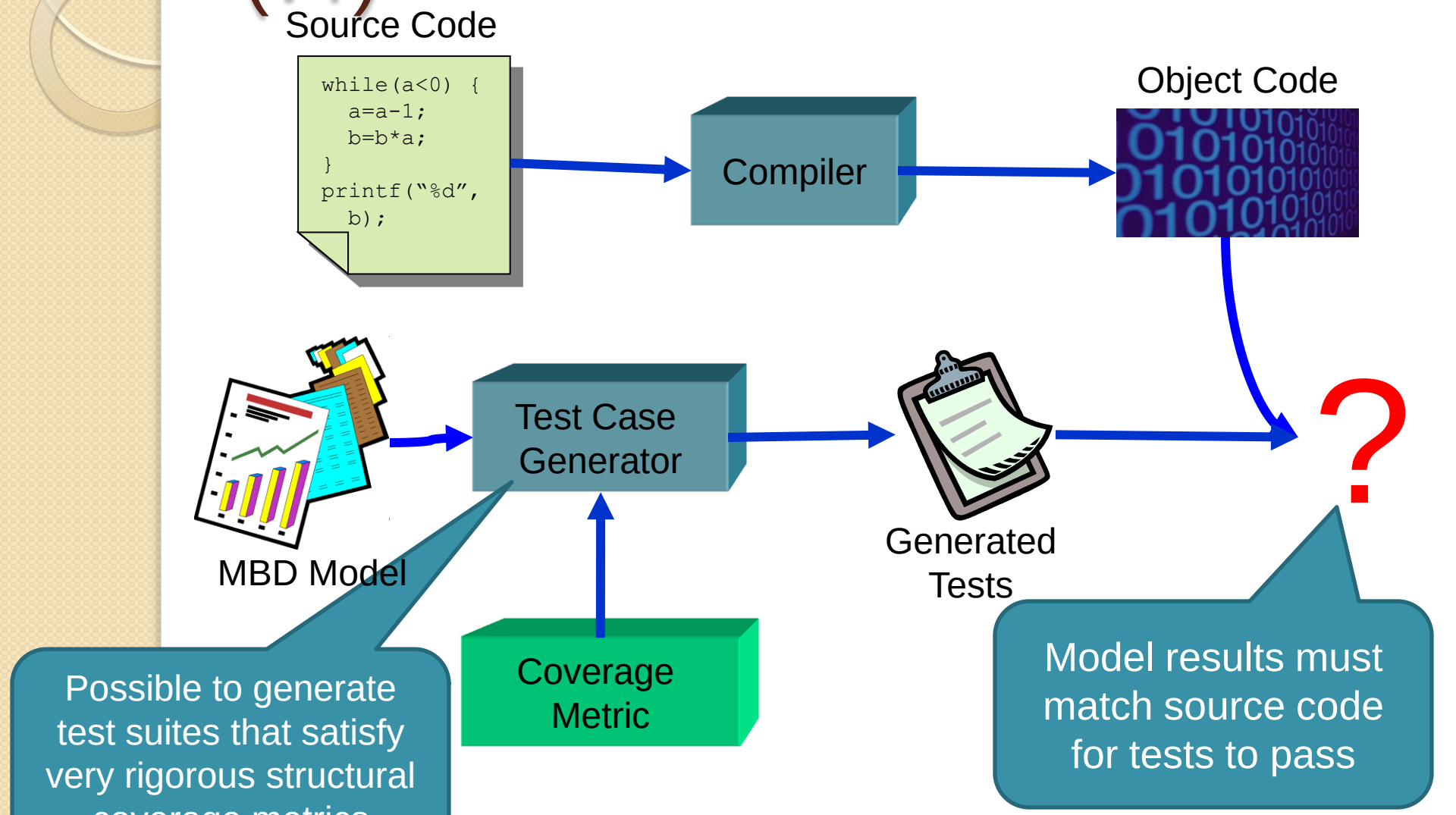
What We Need


Theorem Proving
Perfect Ve
Exhaustive
(Infinit
Access to **Many** Tools and Techniques
Model Checking Temporal Properties Of Finite Systems
MCDC Testing
Data Flow Analysis
Simplistic Program Analysis
Typical Testing
Precise Analysis of Simple Syntactic Properties
Optimistic Inaccuracy
Pessimistic Inaccuracy
Simplified Properties

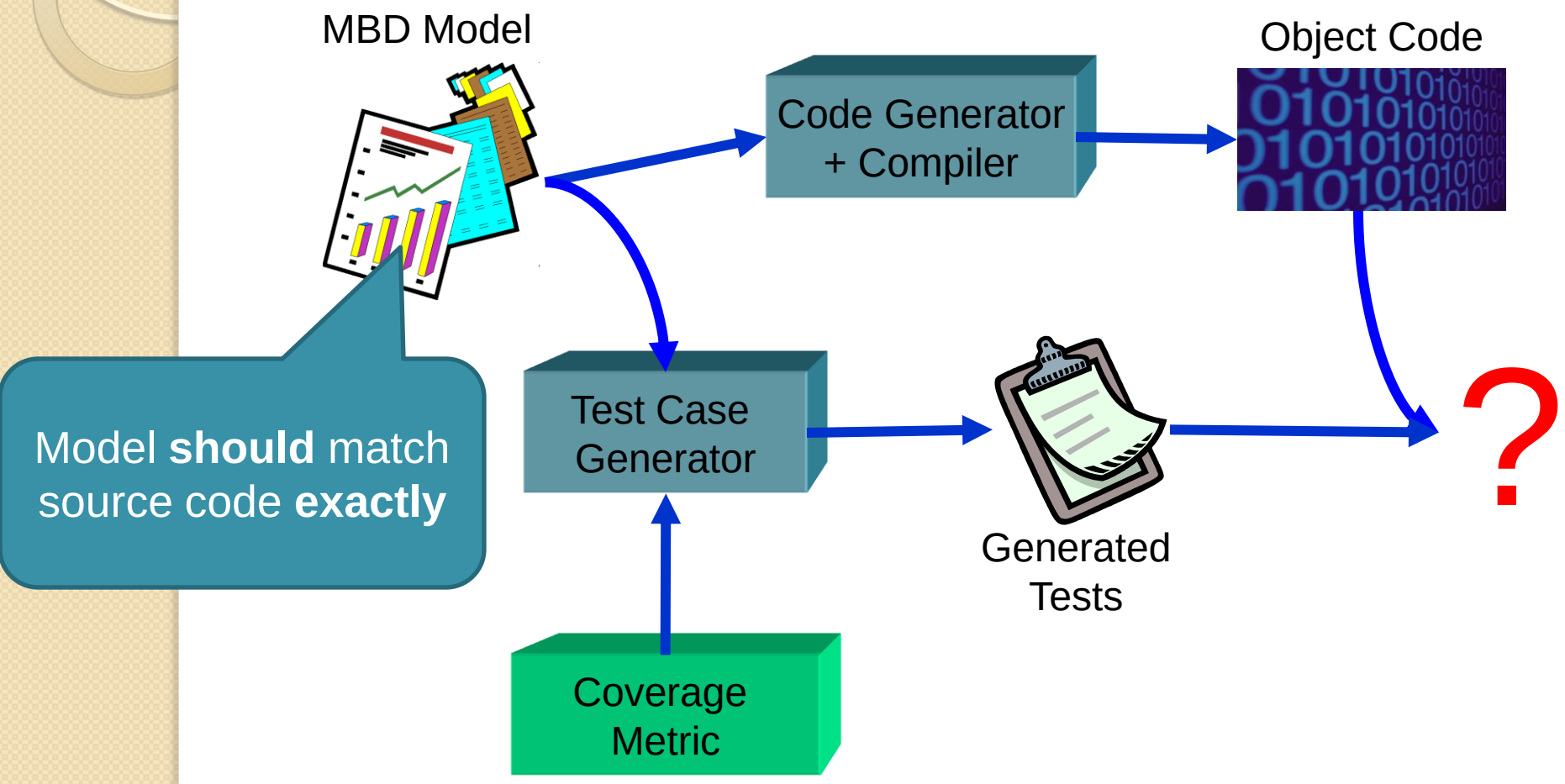
MBD Is a **V&V-Enabling** Technology

- Strong simulation and analysis capabilities built into most tools
  - Demo: Stateflow Elevator
    - (Help: Stateflow/Demos/Large-Scale Modeling/Modeling an Elevator System)
- Even stronger simulation capabilities in external tools
  - Demo: Reactis step simulation with Microwave
- Allows straightforward “Build a little, test a little” philosophy
  - Consistent with incremental development philosophy

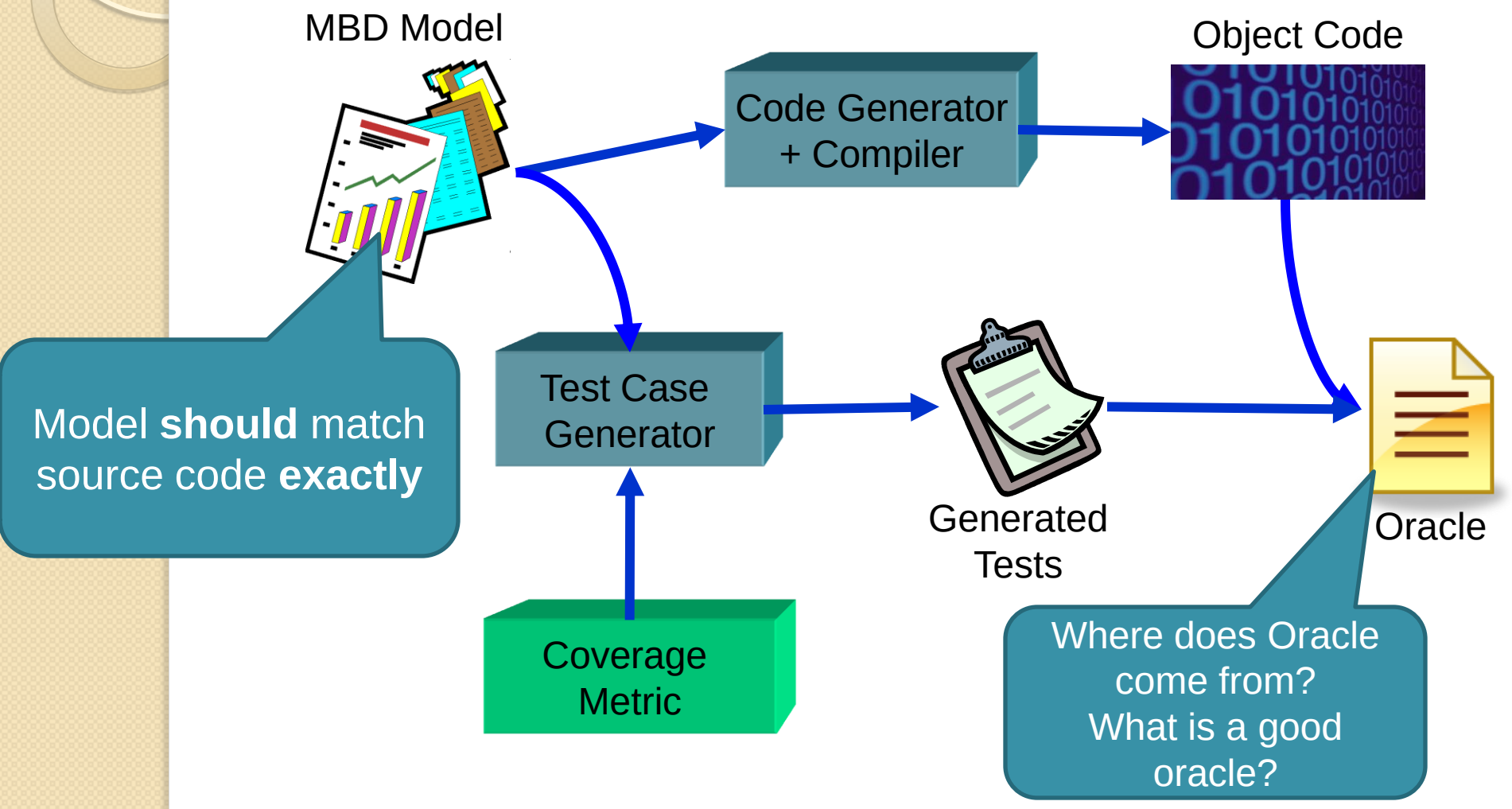
Model-Driven Test Generation (v1)



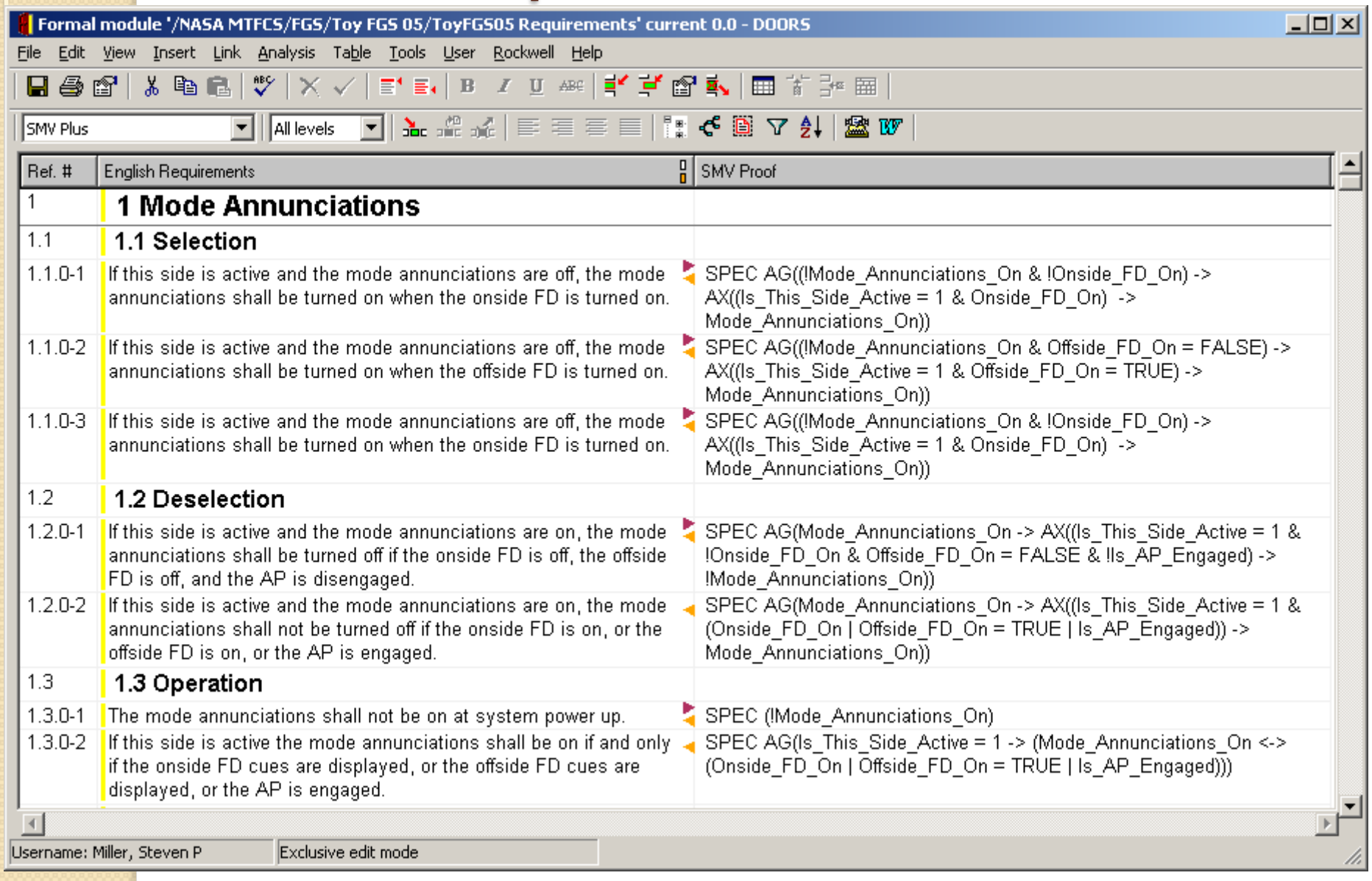
Model-Driven Test Generation (v2)



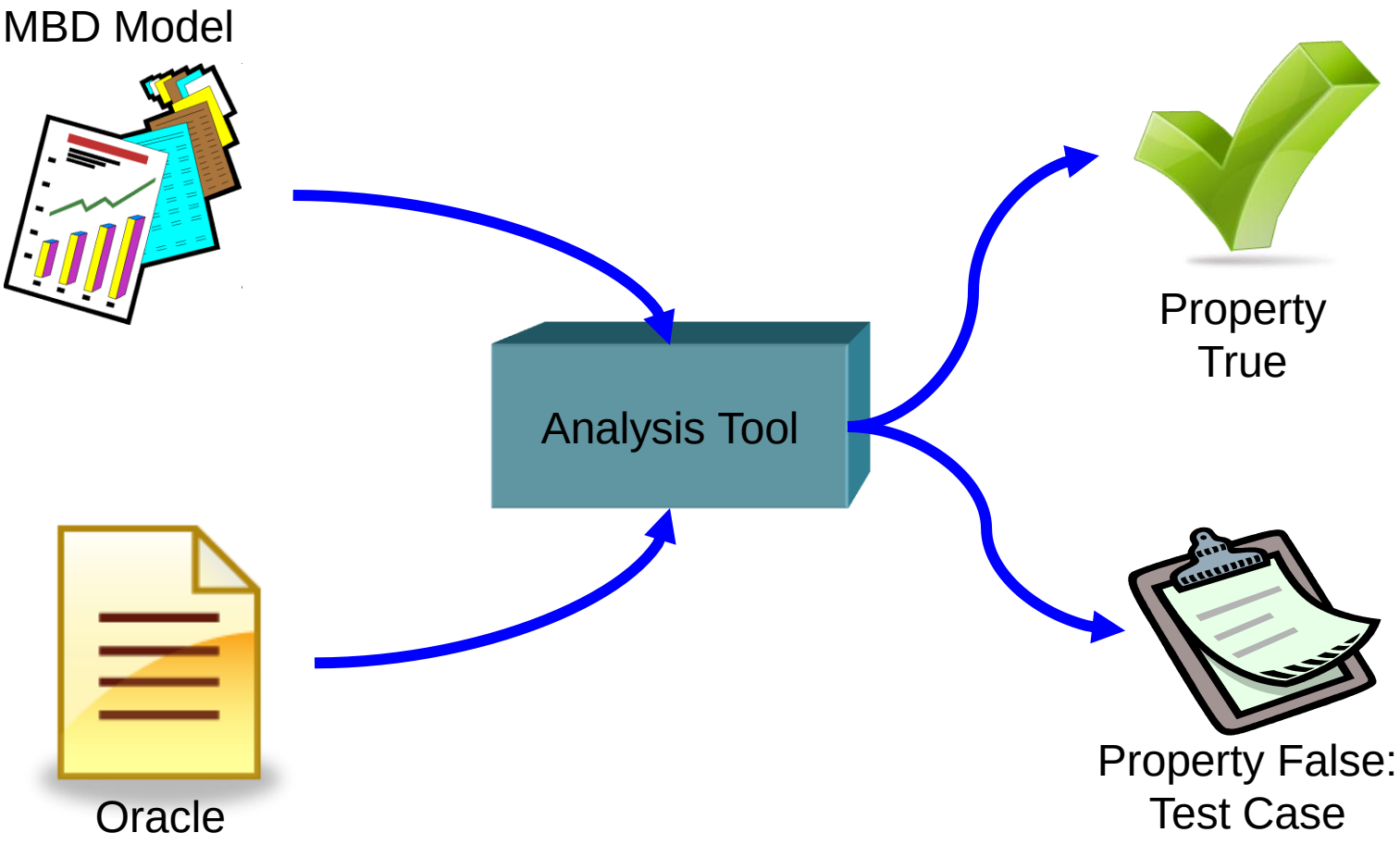
Model-Driven Test Generation (v2)



Use Requirements as Oracle

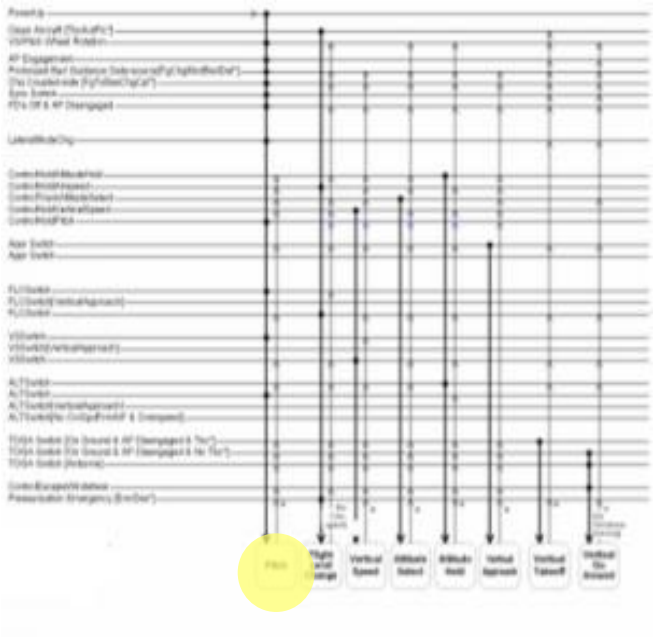


Static Analysis and Model Checking



FCS 5000 Flight Control Mode Logic

Mode Controller A



Example Requirement

Mode A1 => Mode B1

Counterexample Found in Less than Two Minutes

Found 27 Errors

Modeled in Simulink

Translated to NuSMV

6.8 x 10^{21} Reachable States

Mode Controller B

ADGS 2100 Adaptive Display and Guidance System

Modeled in Simulink

Translated to NuSMV

4,295 Subsystems

16,117 Simulink Blocks

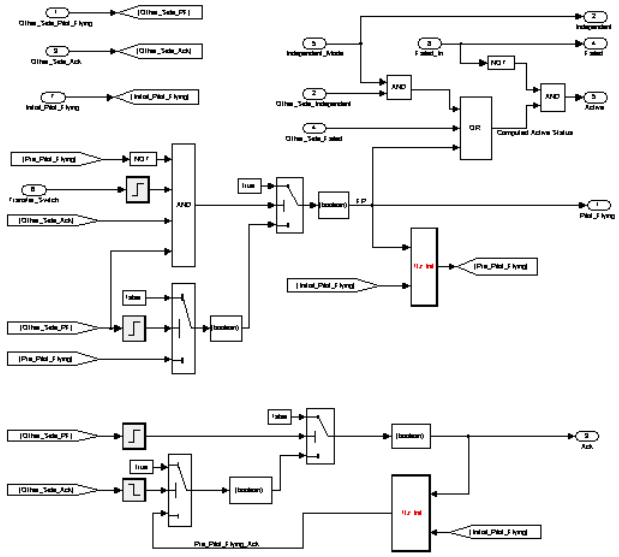
Over 10^{37} Reachable States

Example Requirement:

Drive the Maximum Number of Display Units Given the Available Graphics Processors

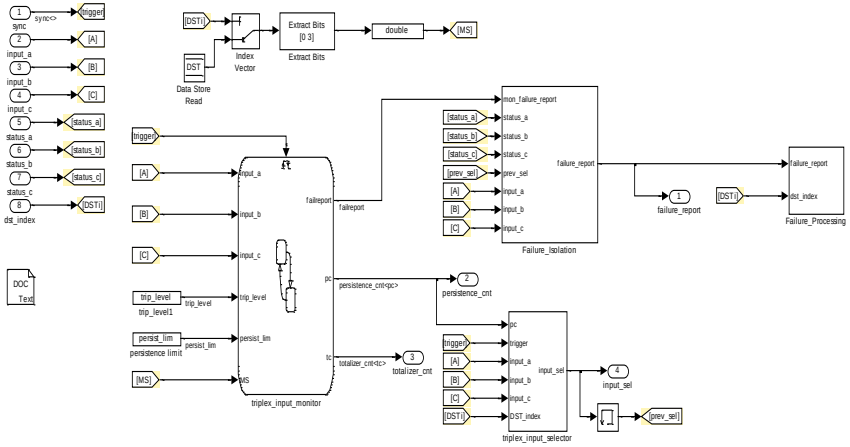
Counterexample Found in 5 Seconds

Checked 573 Properties - Found and Corrected 98 Errors in Early Design Models



CerTA FCS Phase I

- Sponsored by AFRL
  - Wright Patterson VA Directorate
- Compare FM & Testing
  - Testing team & FM team
- Lockheed Martin UAV
  - Adaptive Flight Control System
  - Redundancy Management Logic
  - Modeled in Simulink
  - Translated to NuSMV model checker



Phase I Results

	Effort (% total)	Errors Found
Testing	60%	0
Model-Checking	40%	12

MBD Formal Analysis Efforts

Rockwell Collins — Examples of Using Formal Methods

Rockwell Collins — Examples of Using Formal Methods

Rockwell Collins — Examples of Formal Methods

Rockwell Collins — High Speed Encryptor

Rockwell Collins — Examples of Formal Methods

CerTA FCS Phase II

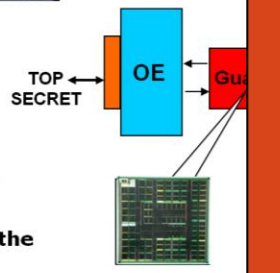
Rockwell Collins — Examples of Using Formal Methods

Turnstile High Integrity Guard

- High-assurance cross domain platform that provides secure communication between different security classification domains ranging from top secret to unclassified.

Accreditable t

- Core guard application is based on the NSA certified AAMP7G.
- I/O processing is relegated to Offload Engines (OE) that do not have to be as highly trusted.
- System integrator can add function to the OE without compromising the guard function.
- Certification based on ACL2 theorem prover



Formal Analysis
of a Triplex Sensor Voter
in an Industrial Context

Michael Dierkes
Rockwell Collins France

FMICS 2011 workshop

August 30, 2011
Trento

Rockwell Collins

Outline of Presentation

Introduction

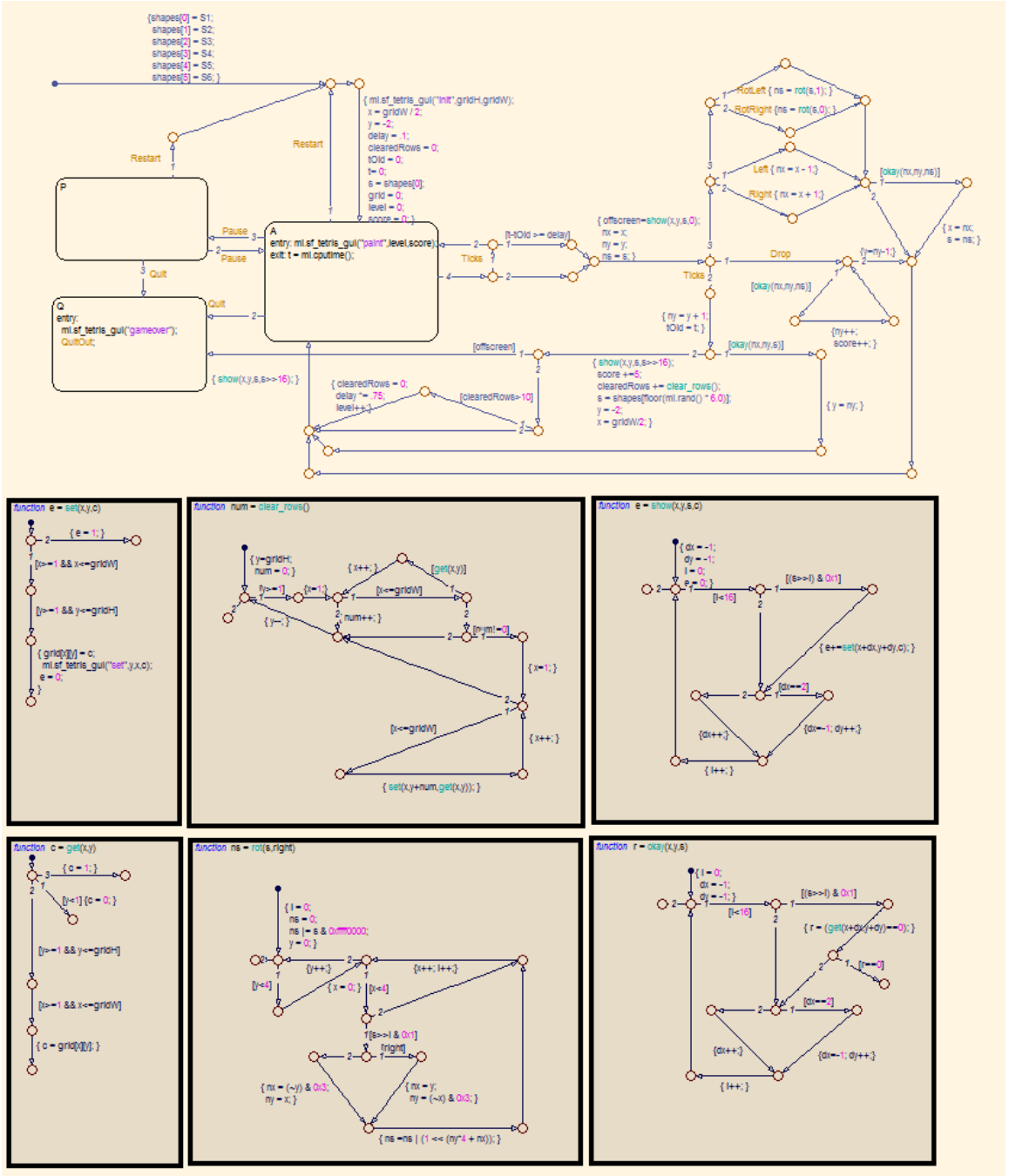
Why use Model-Based Development?
- Requirements
- Design
- Implementation: Code Generation
- Verification and Validation

Pitfalls

Problem 1:
Using Models Where They Don't Fit

If MBD notation doesn't provide a better representation of your problem than code, you're wasting your time.

MBD notations can be awful programming languages…



Model © The Mathworks, 2007

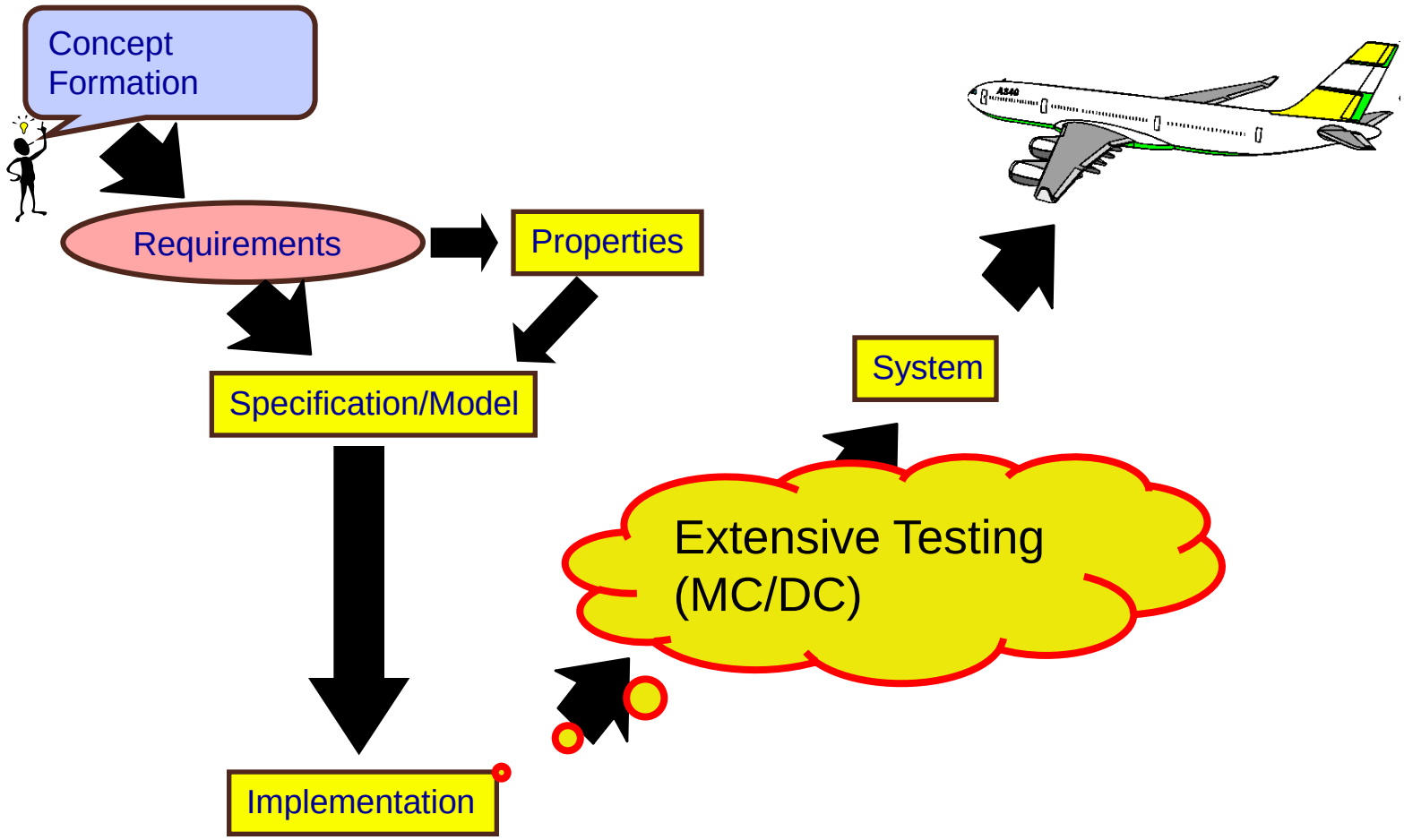
Remedies

- Perform honest assessment of where MBD notations can be used
  - They do not do everything
  - Recursive data structures are especially difficult to model.
  - Use models where they are a good representation.
- Create a partitioning strategy between models and code for applications that contain both complex mode logic and complex data.

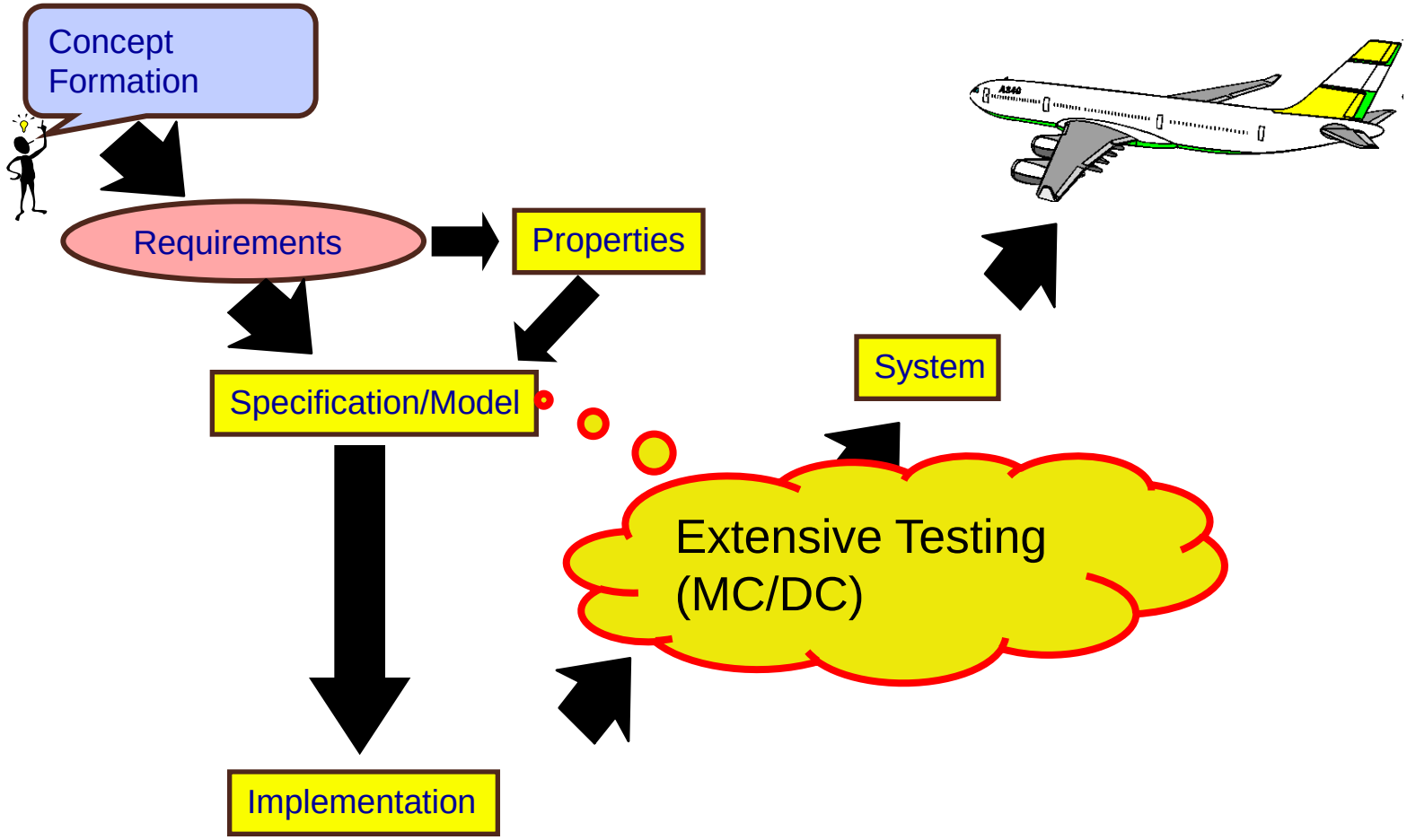
Problem 2
Believing Testing Can be Eliminated

Testing will always be a crucial (and costly) component

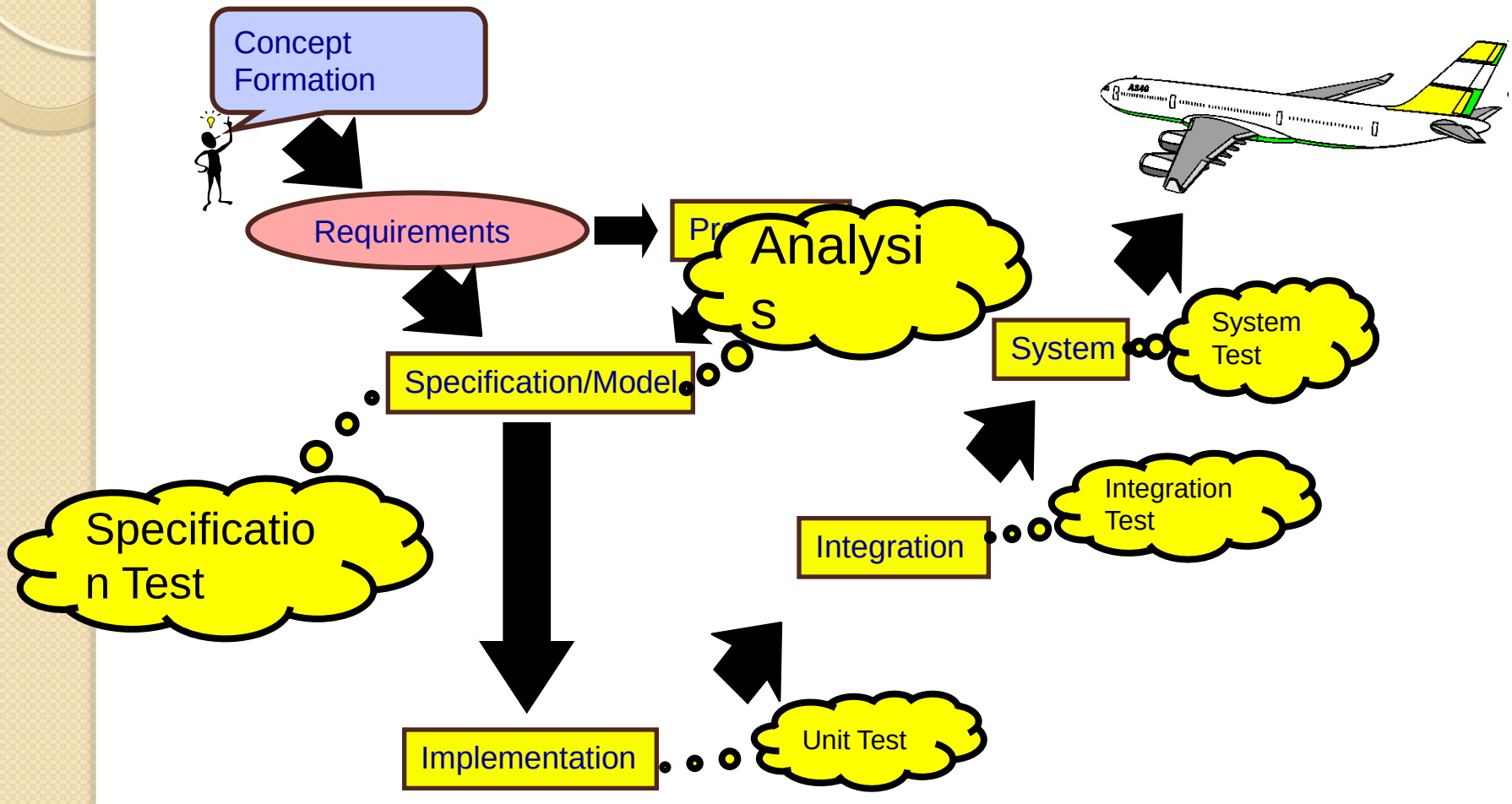
Testing Does not go Away



It Simply Moves



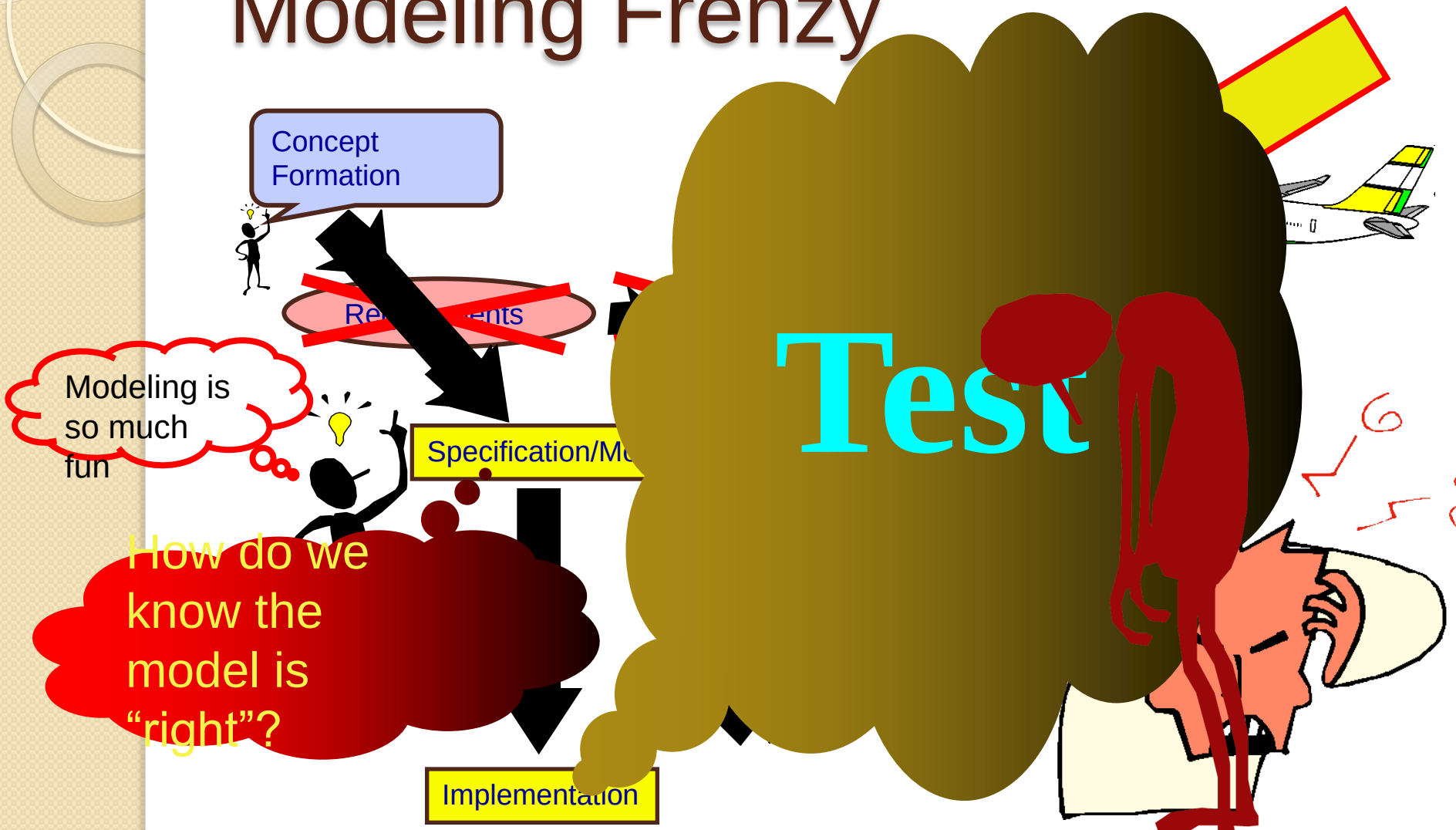
Do it the Right Way



Problem 3
Believing the Model is Everything

The model is never enough

Modeling Frenzy



Remedies

- **Recognize the Role of Software Requirements**
  - The model is not everything
- Development Methods for Model-Based Development Badly Needed
  - Model-Based Software Development Process
- Develop Tools and Techniques for Model, Properties, and Requirements Management
- Develop Inspection Checklists and Style Guidelines for Models

Problem 4
Trusting Verification

To really mess things up, you need formal verification

Property or Model: Who is Right?

The Mode Annunciations shall be turned on when the Flight Director is turned on

AG(Onside_FD_On -> Mode_Annunciations_On)

If this side is active, the Mode Annunciations shall be turned on when the Flight Director is turned on

AG((Is_This_Side_Active & Onside_FD_On)
-> Mode_Annunciations_On)

If this side is active and the Mode Annunciations are off, the Mode Annunciations shall be turned on when the Flight Director is turned on

AG(! Mode_Annunciations_On ->
AX ((Is_This_Side_Active & Onside_FD_On)
-> Mode_Annunciations_On)))

Remedies

- Develop techniques to determine adequacy of model and property set
  - How do we know they are any “good”
- Techniques for management of invariants
  - How do we validate the assumptions we make
- Methodology and guidance badly needed
  - Tools with training wheels
  - “Verification for Dummies”

All we need is one high-profile verified system
to fail spectacularly to set us back a decade or more

Conclusions

- MBD can significantly improve developer productivity, cost, schedule, and quality
- …or it can make your life miserable
- The important thing is to ***know why you're doing it!***
  - Know the limitations of what can be modeled using the DSNs
  - Know which capabilities you hope to use
    - Design and quality of models depends on this
- V & V receives the largest benefit of the MBD approach
  - Mature tools for test-case generation
  - Starting to see model checking built into commercial tools: SCADE Verifier, Simulink Design Verifier
- There are many other things to discuss! Versioning, diff, semantics, tool costs, training, structuring, vendor

Questions?

References

M. Whalen, D. Greve, L. Wagner, S. Miller, Model Checking Information Flow. In *Design and Verification of Microprocessor Systems for High-Assurance Applications.* D. Hardin, Ed. Springer, 2010.

M. Whalen, P. Godefroid, L. Mariani, A. Polini, N. Tillman, and W. Visser. FITE: Future Integrated Testing Environment. *Workshop on the Future of Software Engineering Research 2010 (FoSER),* Santa Fe, New Mexico, November 7-8, 2010.

S. Miller, M. Whalen, and D. Cofer. Software Model Checking Takes Off. *Communications of the ACM*, Volume 53, No 2, February 2010.

D. Hardin, T. D. Hiratzka, D. R. Johnson, L. Wagner, and M. Whalen. Development of Security Software: A High-Assurance Methodology. *Proceedings of the 11th International Conference of Formal Engineering Methods (ICFEM 2009)*, Rio de Janeiro, Brazil, December, 2009.

M. Whalen, D. Cofer, S. Miller, B. Krogh, and W. Storm. Integration of Formal Analysis into a Model-Based Software Development Process. *12th International Workshop on Industrial Critical Systems (FMICS 2007),* Berlin, Germany, July, 2007.

S. Miller, A. Tribble, M. Whalen, and M.P.E. Heimdahl. Proving the Shalls: Early Validation of Requirements through Formal Methods, *Journal of Software Tools for Technology Transfer*. Volume 8 Issue 4, August 2006.

M.P.E. Heimdahl, Y. Choi, and M. Whalen. Deviation Analysis: A New Use for Model Checking, *Automated Software Engineering*, Volume 12, Number 3, July, 2005.

M. Whalen, B. Fischer, and J. Schumann. Certifying Synthesized Code. *Proceedings of Formal Methods Europe 2002,* Copenhagen, Denmark, July 2002

M. Whalen, B. Fischer, and J. Schumann. AutoBayes/CC – Combining Program Synthesis with Automatic Code Certification. *Proceedings of Conference on Automated Deduction 18,*

Medical Cyber-Physical Systems

Improving patient treatment by coordinated systems of medical devices

Research directions:

- Medical device interoperability
- High-confidence development
  - Model-driven design
  - V&V, regulatory approval
- Smart alarms and decision support
- Physiological closed-loop control

Supported by NSF CNS-1035715

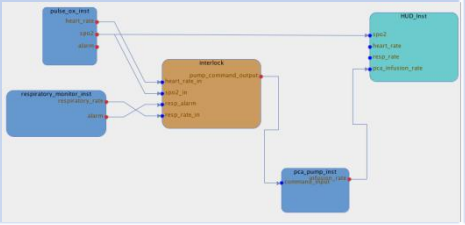
http://rtg.cis.upenn.edu/MDCPS/

Participants

- University of Pennsylvania
- U. Penn Hospital System
- University of Minnesota
- CIMIT/MGH

Coordination framework for medical devices

- Build high-confidence middleware
  - Rely on formal methods and static analysis
- Design a language for executable clinical scenarios
  - Specify information flows
  - Identify timing constraints
  - Ensure non-interference



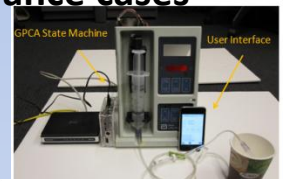
Model driven development and assurance cases

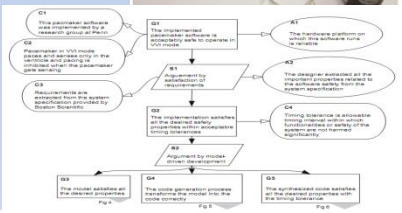
High-assurance development:

- Modeling, code synthesis
- Model-level verification, code-level validation

Assurance case construction reflects development process structure

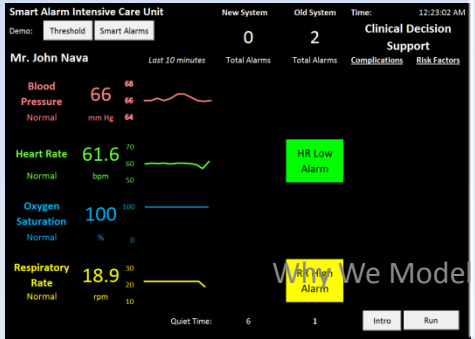
Applied to pacemaker, PCA pump





Smart alarm systems

- Reduction of irrelevant alarms for CABG patients
  - Based on aggregation of multiple vital signs and fuzzy logic
- On-going research:
  - Prediction of vasospasm in neuro-ICU patients



Networked Blood Glucose Control System

Safety-critical, closed-loop MCPS

Research issues:

- Identifying new risks and hazards
- Mitigation strategies
- Validation
- Control design

Pursue model-driven approach