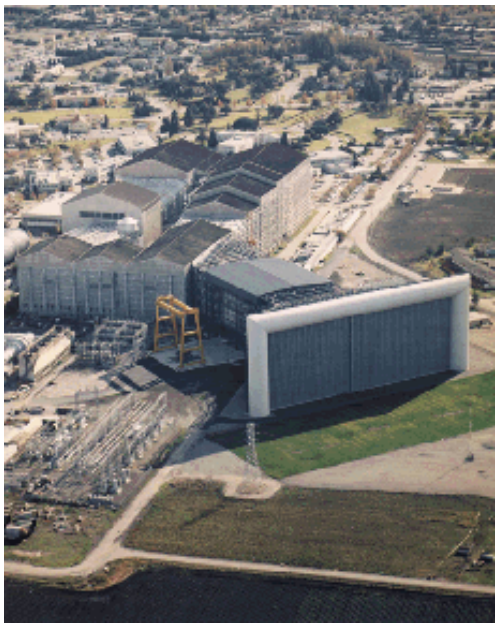
Verification & Validation for PHM

Dr. Guillaume Brat, NASA ARC

NASA Ames Research Center, Code TI

Who am I?

- Deputy Area Lead for Robust Software Engineering in the Intelligent Systems division at the NASA Ames Research Center
  - Our goal is to increase the reliability and robustness of NASA software, and the productivity of its software engineering, through the research, development, application, and transfer of automated software engineering technology that scales to meet NASA's software challenges.
  - We draw upon many techniques from Computer Science (eg, program verification, automated reasoning, model checking, static analysis, symbolic evaluation, and machine learning) and apply them to the verification and validation of software, as well as code generation.



- Bio nuggets:
  - Ph.D. from the ECE Dept. at The University of Texas at Austin in 1998
  - Principal Scientist for the FY'10 V&V of Flight Critical Systems study for the Aviation Safety program in NASA ARMD

Outline

- Setting the stage
- A few definitions
- Current V&V trends
- What about V&V for PHM?
- Conclusion: need more research

Setting the stage

Focus on aeronautics

Impact: Cost, and Constraints on Innovation

Size Comparisons of Embedded Software

System	Lines of Code
Mars Reconnaissance Orbiter	545K
Orion Primary Flight Sys.	1.2M
F-22 Raptor	1.7M
Seawolf Submarine Combat System AN/BSY-2	3.6M
Boeing 777	4M
Boeing 787	6.5M
F-35 Joint Strike Fighter	5.7M
Typical GM car in 2010	100M

NASA Study

Flight Software Complexity, 4/23/2009

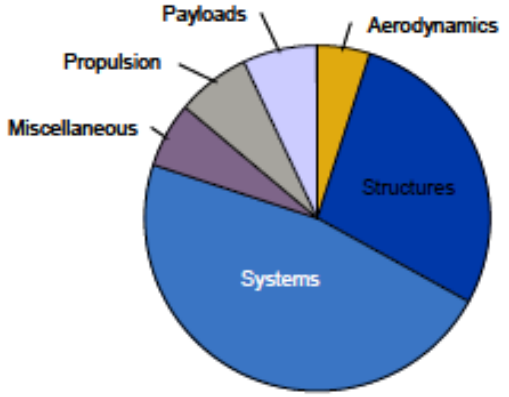
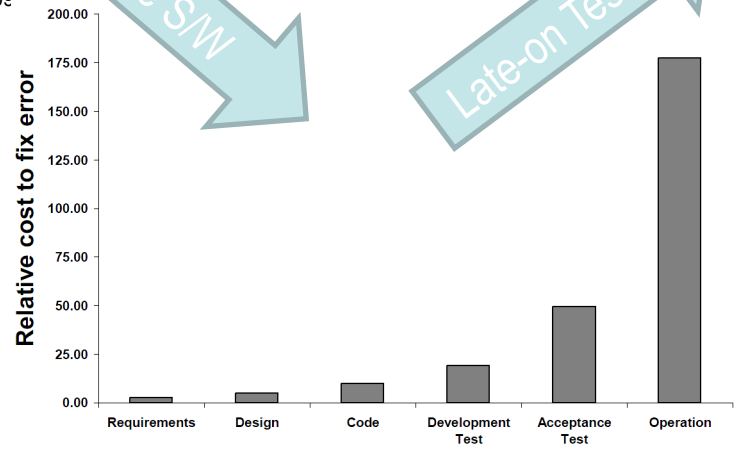


Fig. 1 - Typical Transport Aircraft Development Cost Distribution – Current Generation

Winter, D. (VP, Engineering & IT, Boeing PW) House Committee on Science and Technology, July 31, 2008



Boehm, B. 1981 *Software Engineering Economics*, as cited in DAA, 2008

Software Certification

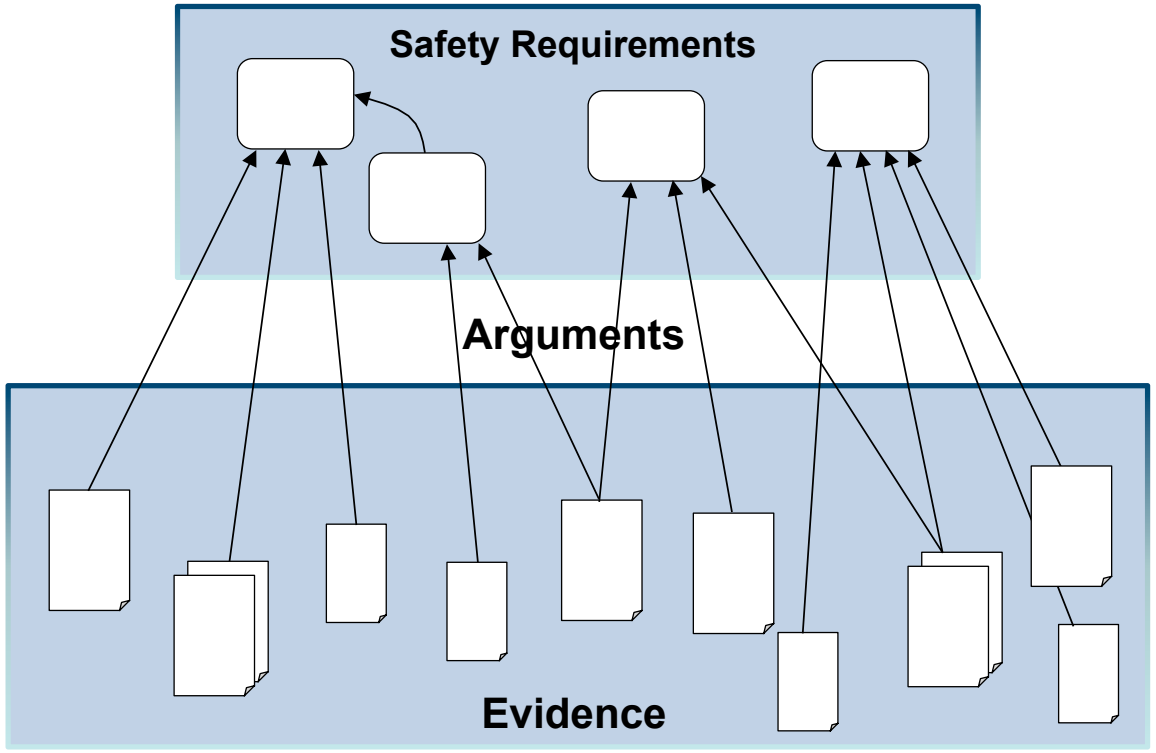
- **DO-178B / ED-12B**: Software Considerations in Airborne Systems and Equipment Certification
  - Prepared by RTCA SC-167 and EUROCAE WG-12

- Main V&V elements:
  - Requires structural testing (with code coverage analysis)
    - testing of executable object code
  - Strong emphasis also on requirement-based testing
  - Review of all requirements, design and code
    - Need to ensure traceability between requirements, code, and the tests that verify that the code satisfy the requirements

Software Certification Evolution

- **DO-178C**
  - Prepared by RTCA SC-205 and EUROCAE WG-12
- Among other things, it emphasizes:
  - Model-based design and verification
    - New guidance for model execution/simulation
    - Attempting to define model coverage metrics
  - Formal methods
    - Define new objectives/activities/documentation (abstractions, assumptions)
    - Avoid common errors (false hypotheses)

Place of V&V in certification



- V&V provides evidences for a certification case

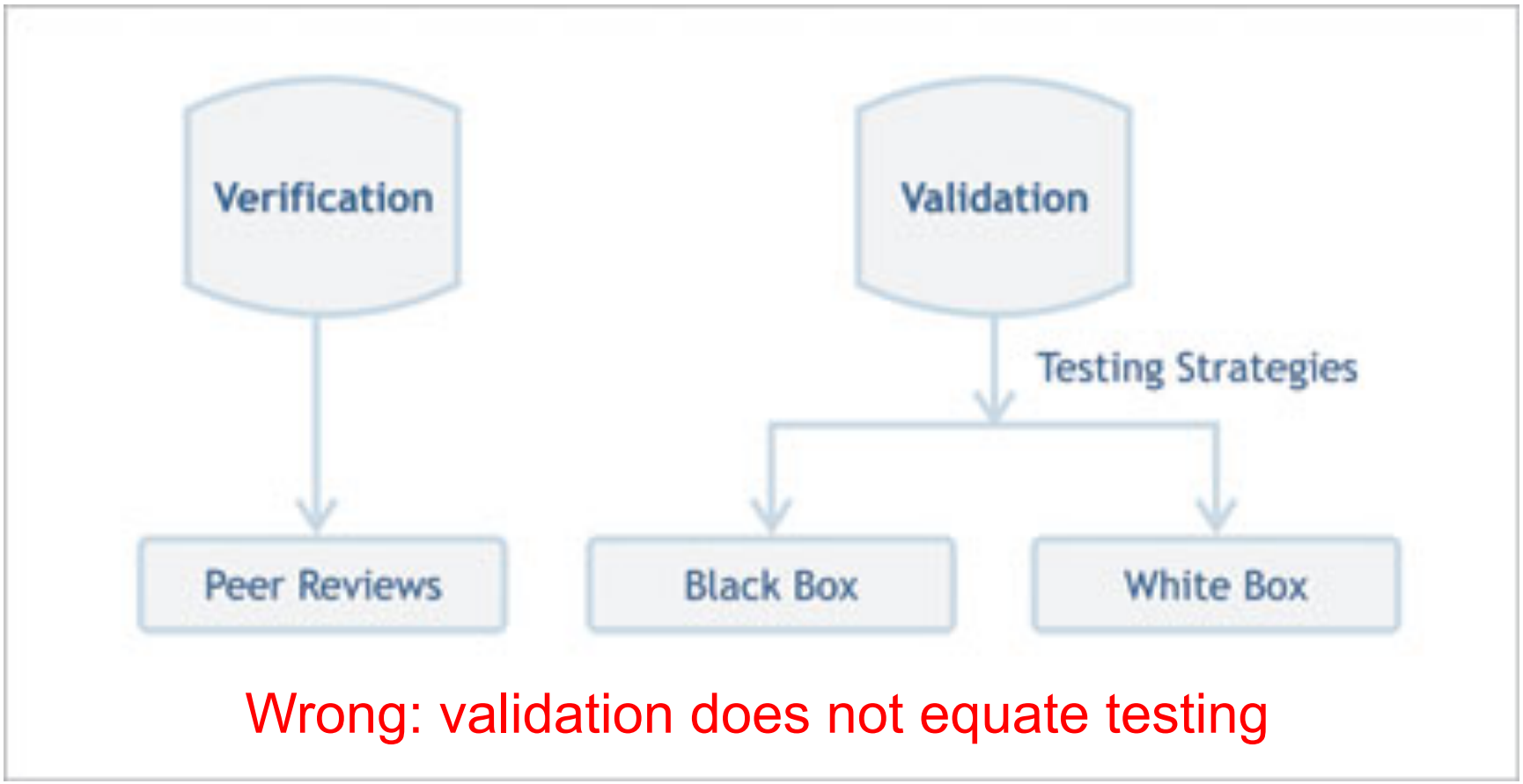
A few definitions

V&V Definitions

- Verification is a process that is used to evaluate whether or not a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase.
  - Are we building it right?

- Validation is the process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements.
  - Have we built the right thing?

V&V Misconception



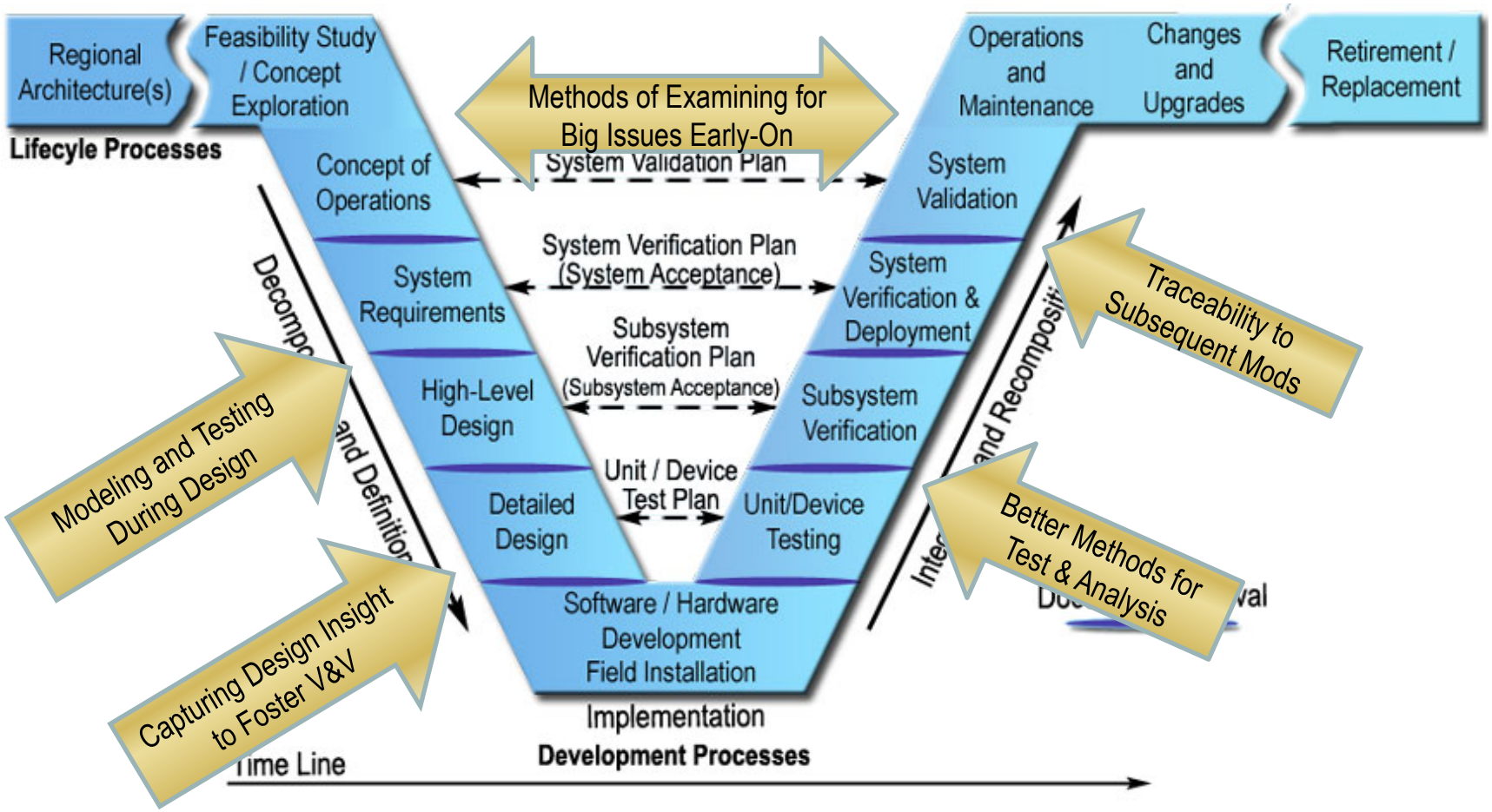
Wrong: validation does not equate testing

In the research world

- During the research activity, we're mostly concerned with validation:
  - Is the new algorithm solving the right problems?
  - How do we convince people that it is the case?
  - We typically do just enough verification to make sure that we can trust the simulation results.
- During technology transfer, we're very concerned with verification:
  - Is the software correct with respect to the specifications?
  - We are still concerned with validation because nothing replaces validation on the final product.

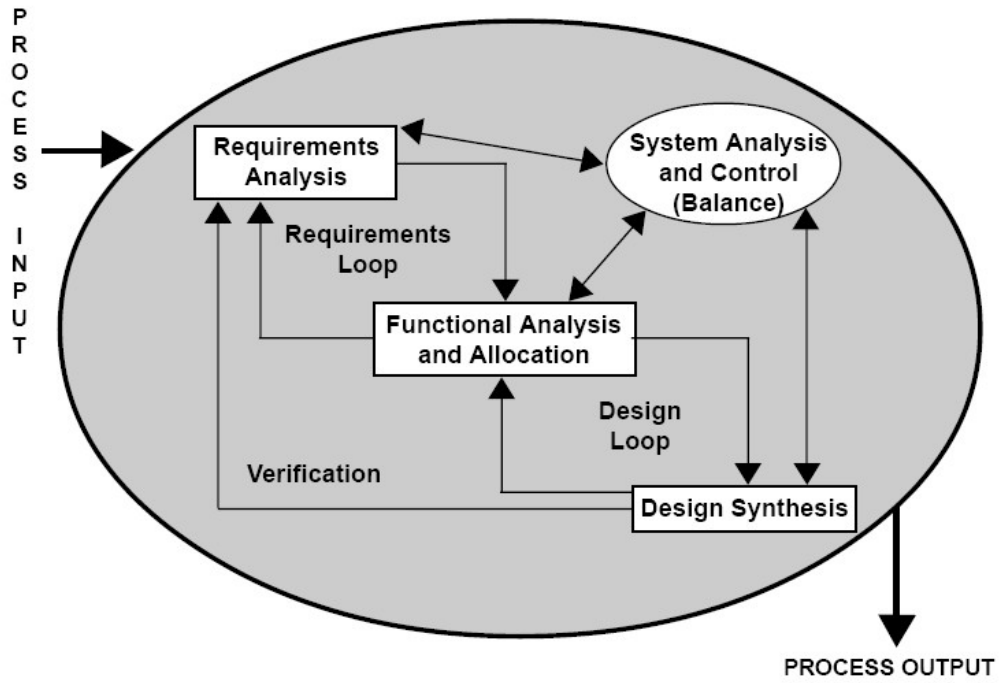
Current V&V trends

V&V and the Lifecycle Model



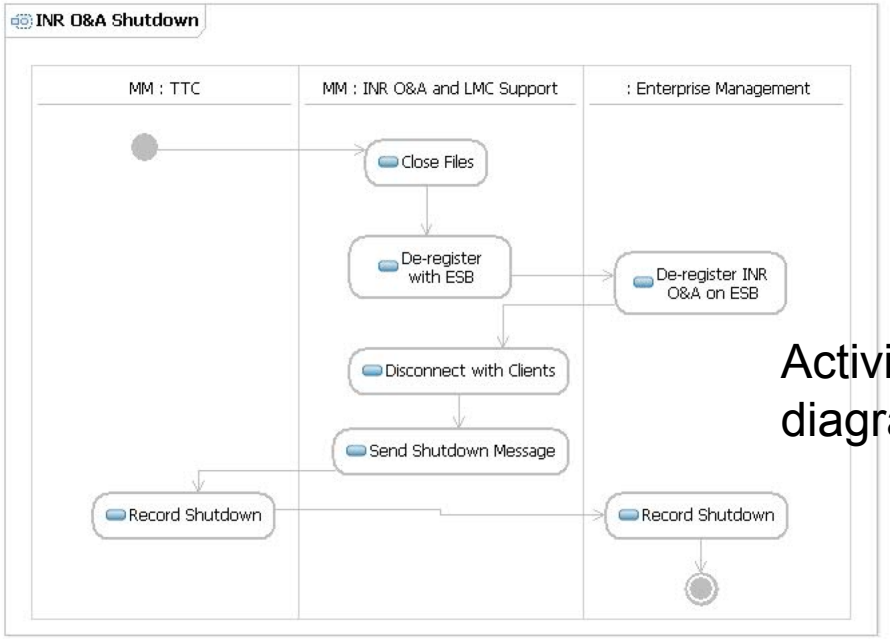
Requirements

- **Requirements analysis** determine the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders.



Requirements

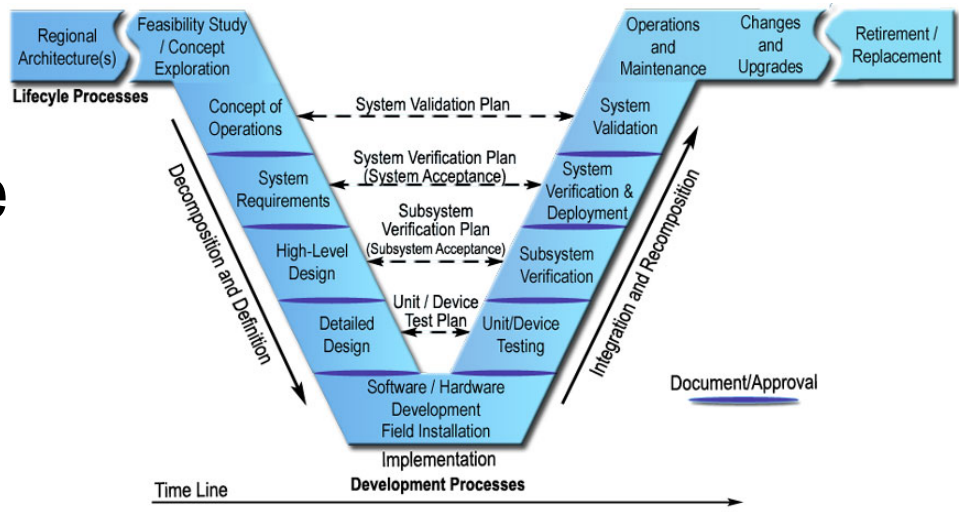
- Requirements can be
  - architectural,
  - structural,
  - behavioral,
  - functional, and
  - non-functional.



Activity diagrams

3.1.1	The Spacecraft Navigation Sub-system shall stop performing Spacecraft Navigation upon receipt of a Shutdown Ground Command.	Spacecraft Navigation

Requirements



- Requirements must be
  - documented,
  - measurable,
  - testable,
  - defined to a level of detail sufficient for system design, and,
  - traceable all the way to code and tests.
- Requirements are mostly verified through peer reviews
  - Some checks (conflicts, ambiguities) can be automated if requirements are expressed formally.

Tabular representation: (RSML^{-e}, SCR)

2.3 Flight Director (FD)

The Flight Director (FD) displays the pitch and roll guidance commands to the pilot and copilot on the Primary Flight Display. This component defines when the Flight Director guidance cues are turned on and off.

Definitions of Values to be Imported

MACRO

When_Turn_FD_On

Condition:

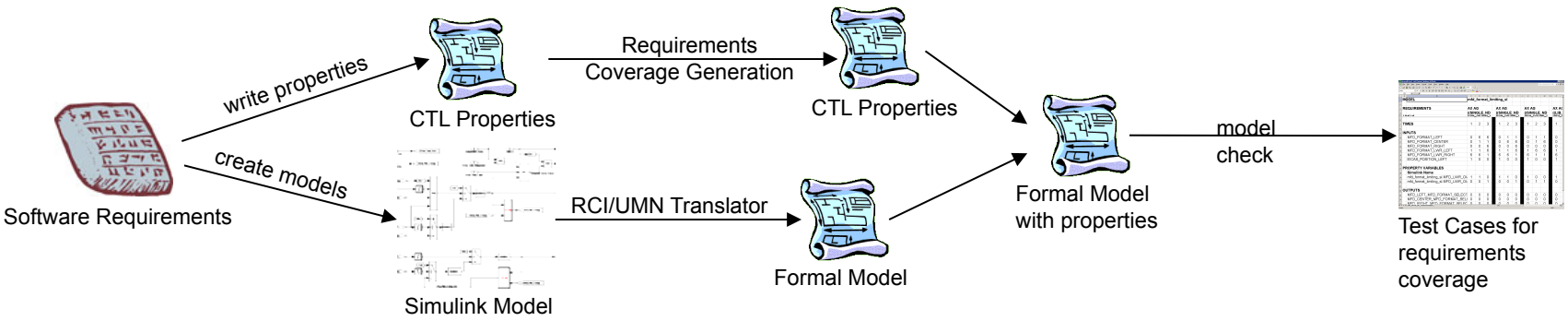
								OR
AND	When_FD_Switch_Pressed$_{m\text{-}96}$()	T	·	·	·	·	·	·
	When(AP$_{v\text{-}129}$ =Engaged)	·	T	·	·	·	·	·
	When(Overspeed$_{v\text{-}118}$)	·	·	T	·	·	·	·
	When_GA_Switch_Pressed$_{m\text{-}102}$()	·	·	·	T	·	·	·
	When_Lateral_Mode_Manually_Selected$_{m\text{-}23}$()	·	·	·	·	T	·	·
	When_Vertical_Mode_Manually_Selected$_{m\text{-}24}$()	·	·	·	·	·	T	·
	When_Pilot_Flying_Transfer$_{m\text{-}26}$()	·	·	·	·	·	·	T
	Pilot_Flying$_{v\text{-}26}$ =THIS_SIDE$_{LEFT}$	·	·	·	·	·	·	T
	Were_Modes_On$_{m\text{-}31}$()	·	·	·	·	·	·	T

Purpose: This event defines when the onside FD is to be turned on (i.e., displayed on the PFD).

Models and Testing

The goal of Model-Based Testing (MBT) is to reduce costly test development.

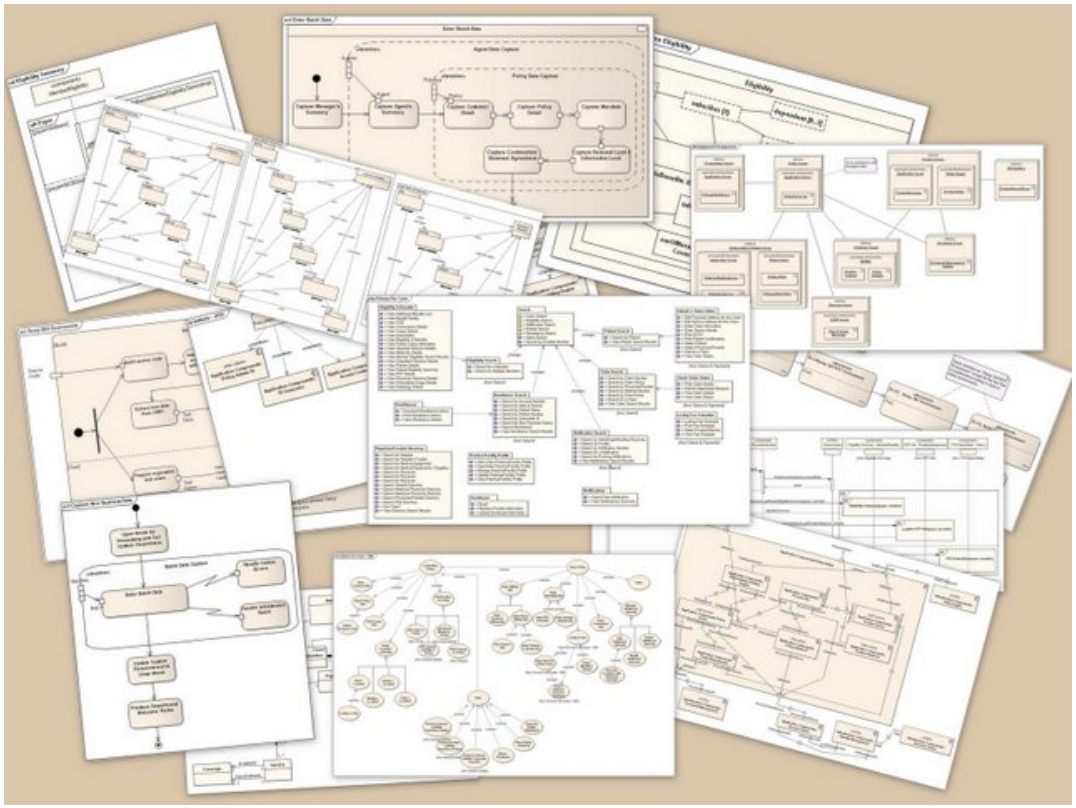
The use of model checking enables the QA engineers to test contractor code developed from the models



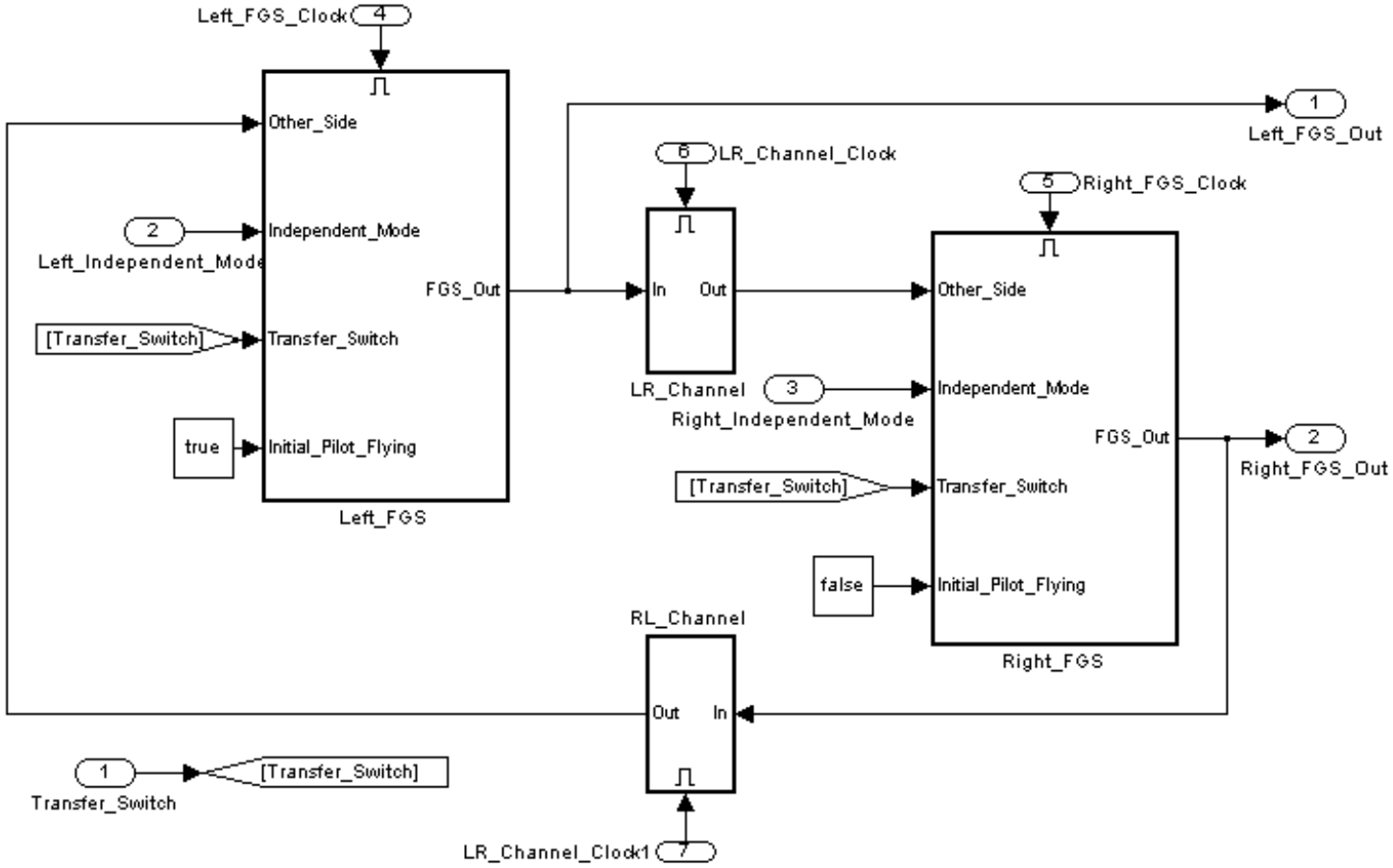
Design

- Software design is a process of problem-solving and planning for a software solution that meets the requirements.

UML diagrams



Simulink model

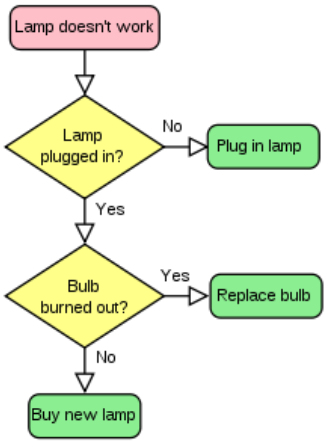


Model-based Development Examples

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

Model-based development process

- More and more, code is automatically generated from design models.
- Verification can be built into the auto-coding process.

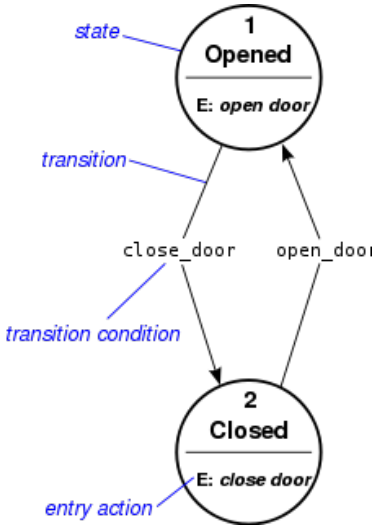


Flow chart

Program

```
void add(Object o) {
 buffer[head] = o;
 head = (head+1)%size;
}

Object take() {
 …
 tail=(tail+1)%size;
 return buffer[tail];
}
```

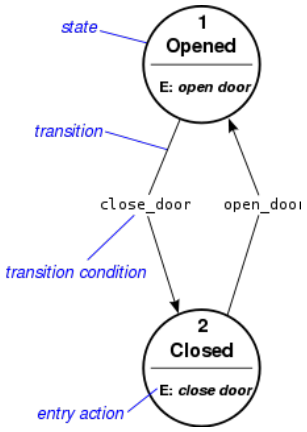


Finite state machine

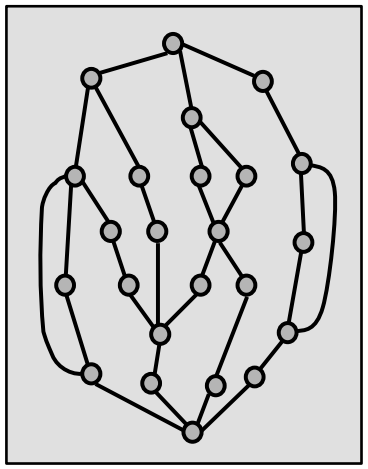
- It's very important to verify and validate models

Model Checking

Model



model checker

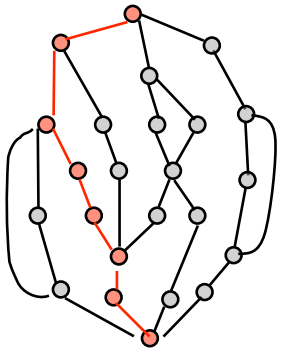


NO + counterexample:
(provides a violating execution)

YES (property holds)

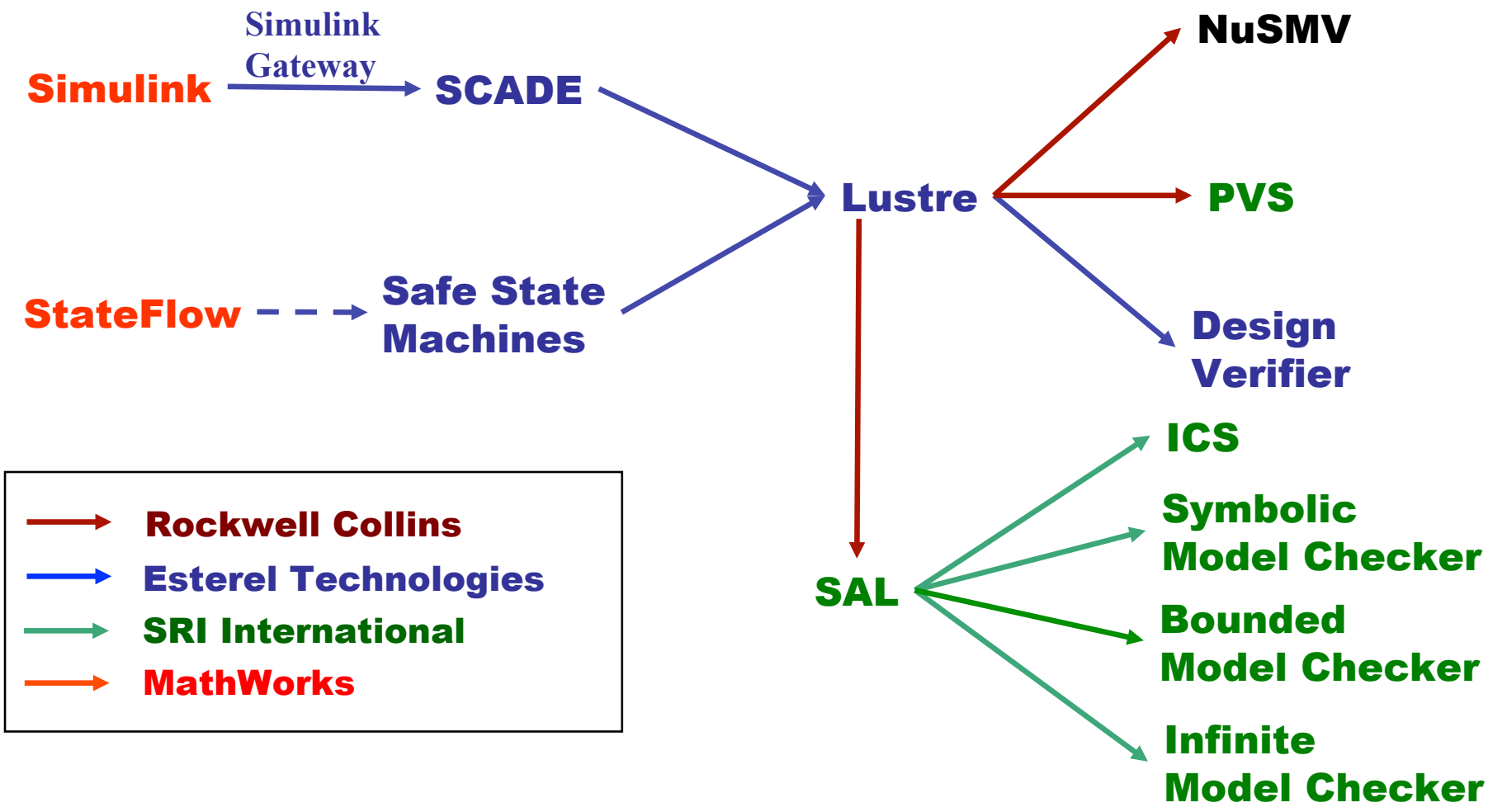
safety property

always(ϕ **or** ψ)



Testing

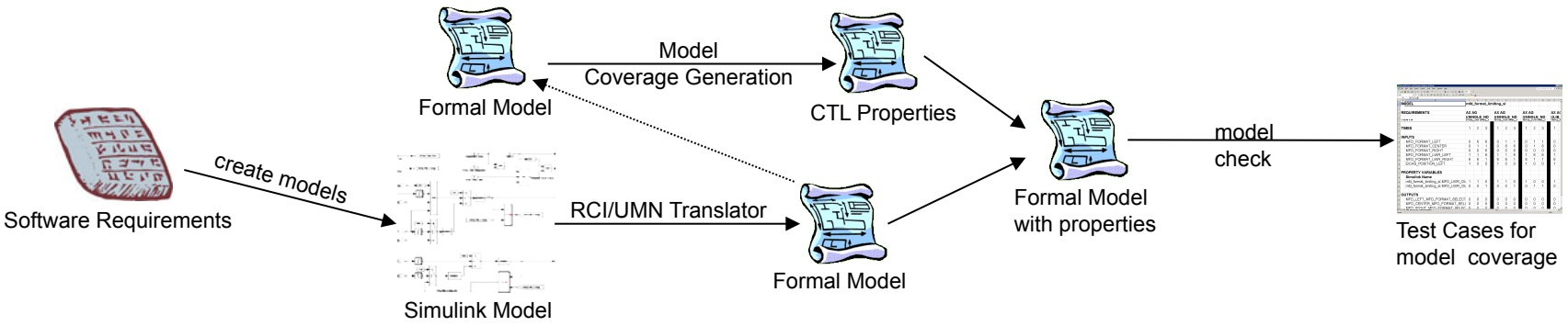
Formal Requirement/design V&V: example



Models and Testing

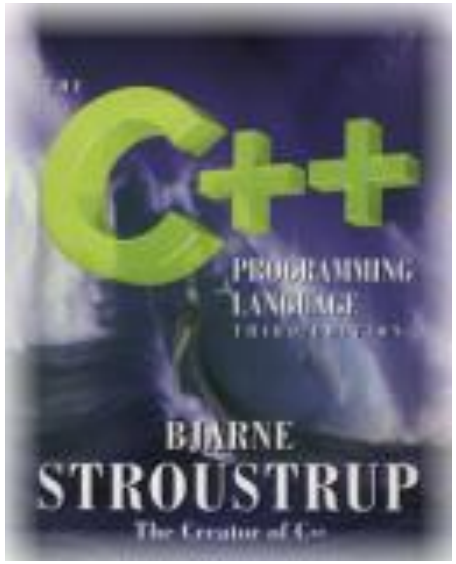
The goal of Model-Based Testing (MBT) is to reduce costly test development.

The use of model checking enables the QA engineers to test contractor code developed from the models



Coding

- Coding is the process of designing, writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs.
- Most embedded systems are coded in C and C++.

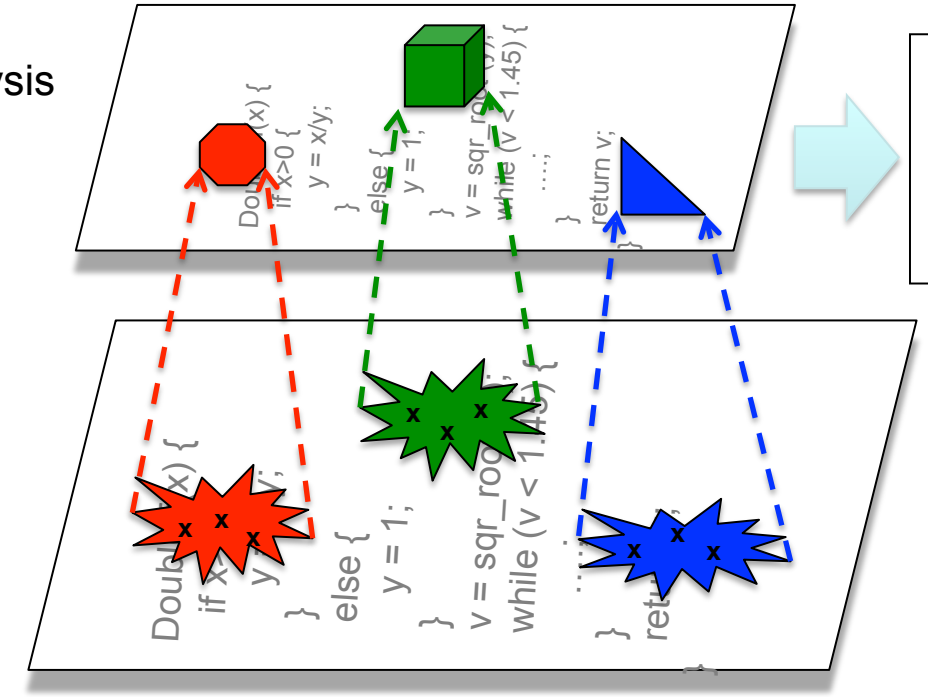


- The main V&V means for coding are:
  - Dynamic analysis, or testing, and,
  - Static analysis.

Static Analysis

The goal of static analysis is to exercise all data ranges for all paths

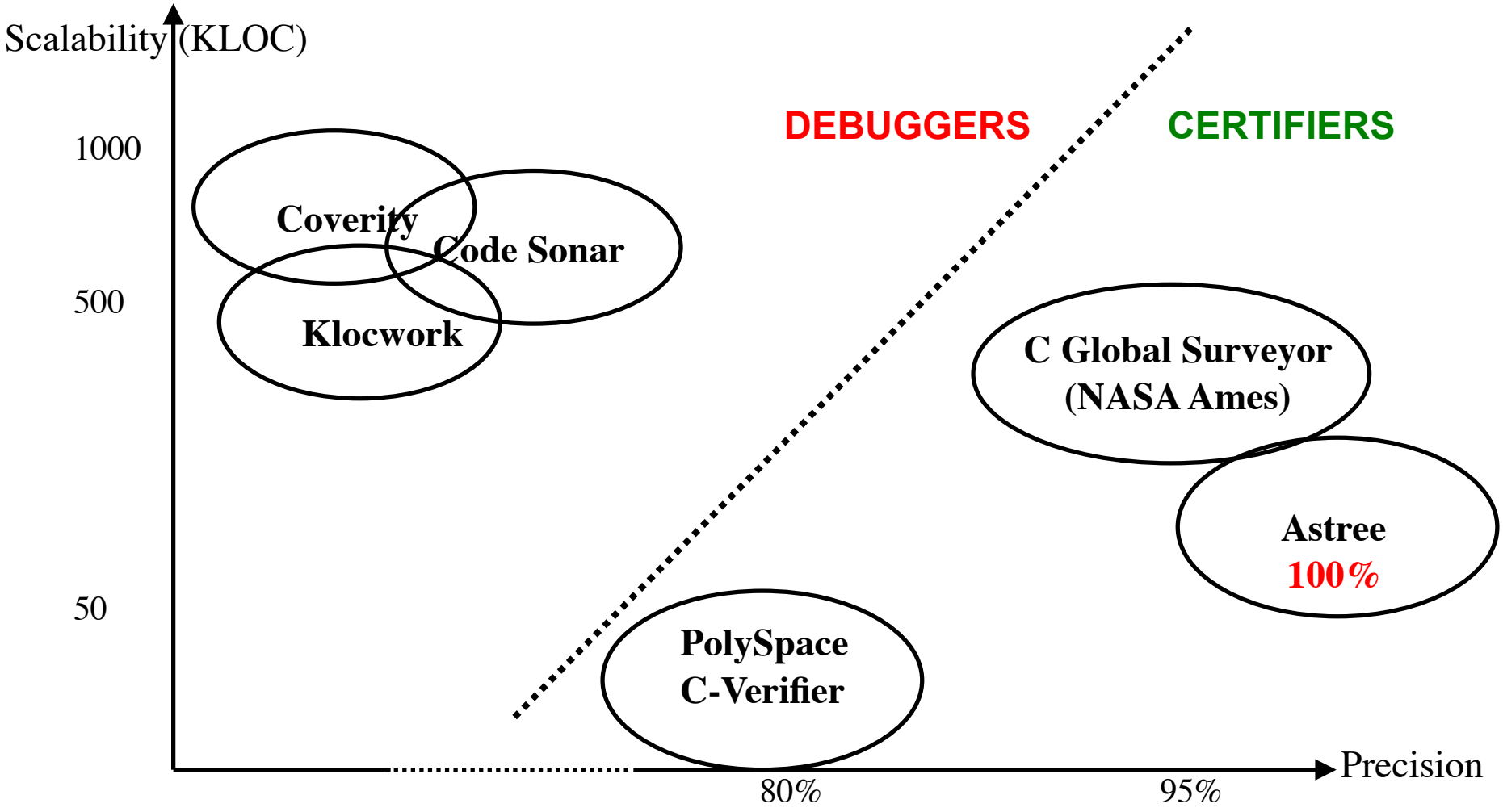
Testing exercises some data points and some paths



Static analysis is well-suited for catching runtime errors, e.g.:

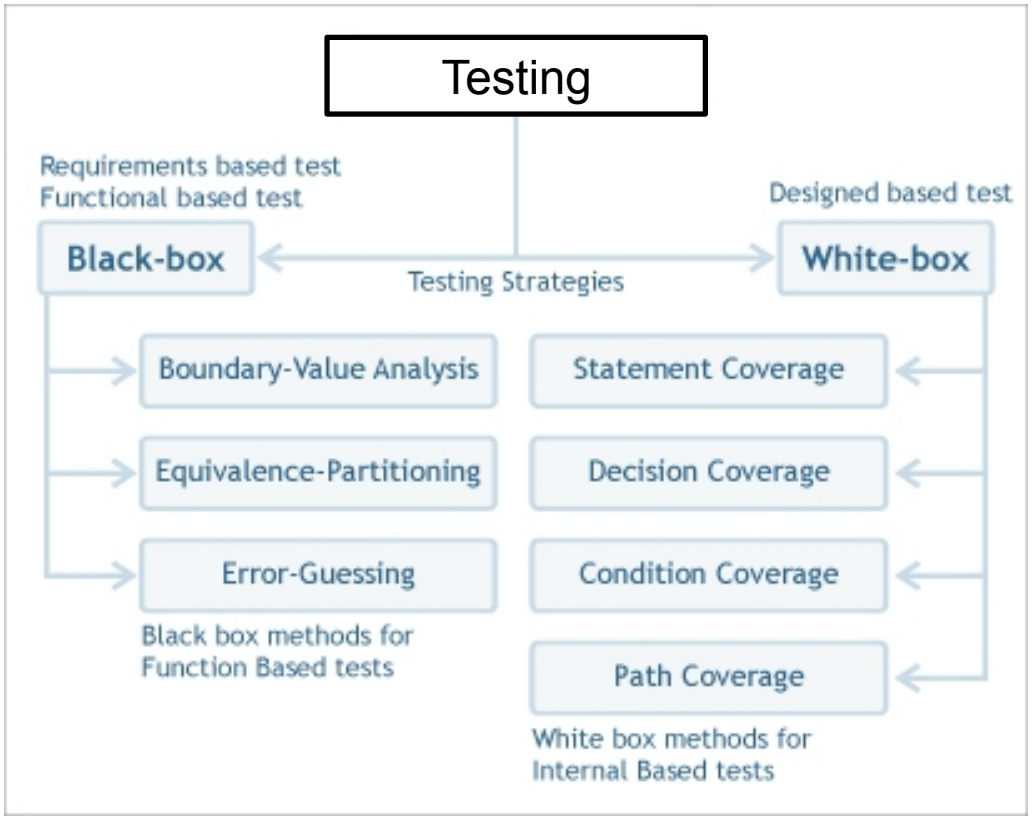
- Array-out-bound accesses
- Un-initialized variables/pointers
- Overflow/Underflow
- Invalid arithmetic operations

Static Analysis Challenge



Software testing

- Testing is the most common means of verifying software.

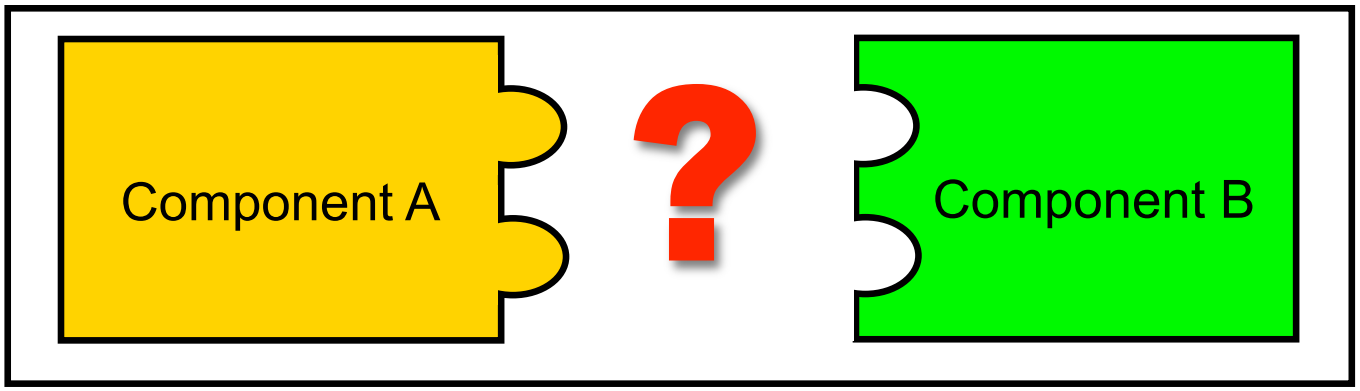


Unit Testing

- Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level.
  - They are usually written by developers as they work on code, to ensure that the specific function is working as expected.
- One function might have multiple tests, to
  - validate nominal cases
  - exercise corner cases or
  - verify that off-nominal cases are caught by exception handlers.
- Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Integration Testing

- Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design.



- Traditionally, larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

Compositional Verification

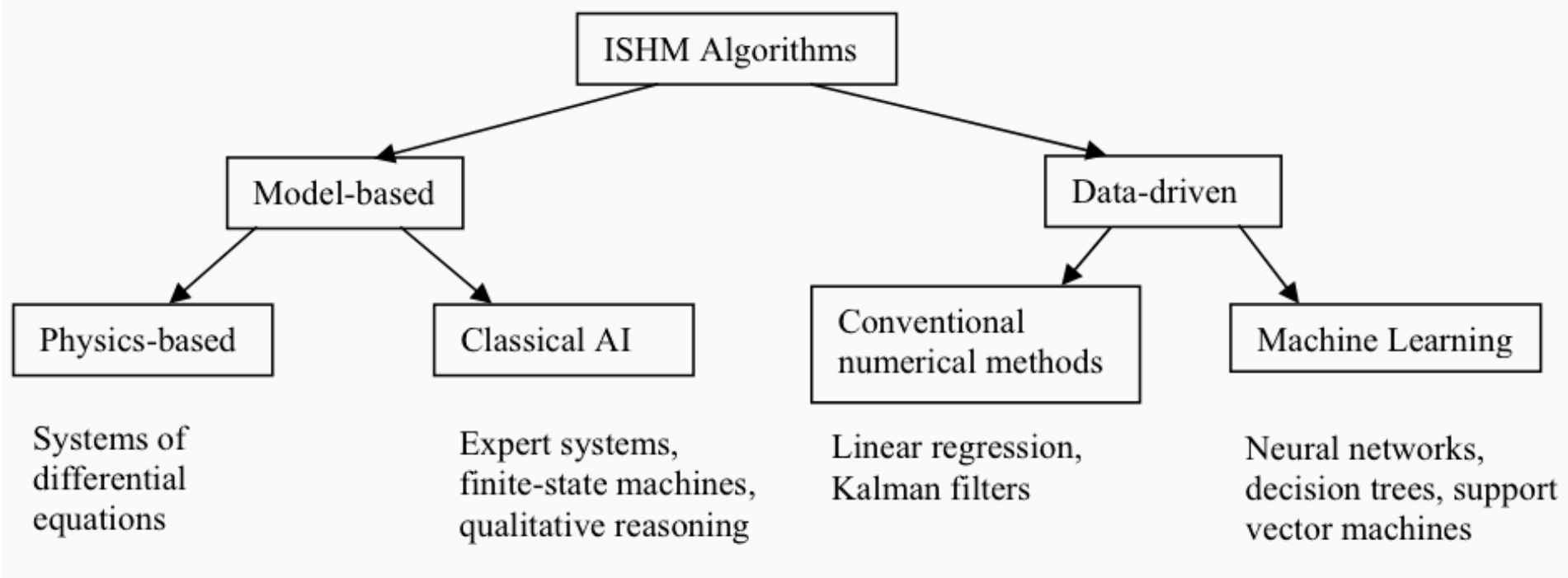
- Use system's natural decomposition into components to break-up the verification task
  - Divide-and-Conquer approach
- Components typically satisfy requirements in specific contexts / environments
  - safety assumptions about contexts
- System safety derives from the ability to compose the components' contexts at the system level

System V&V

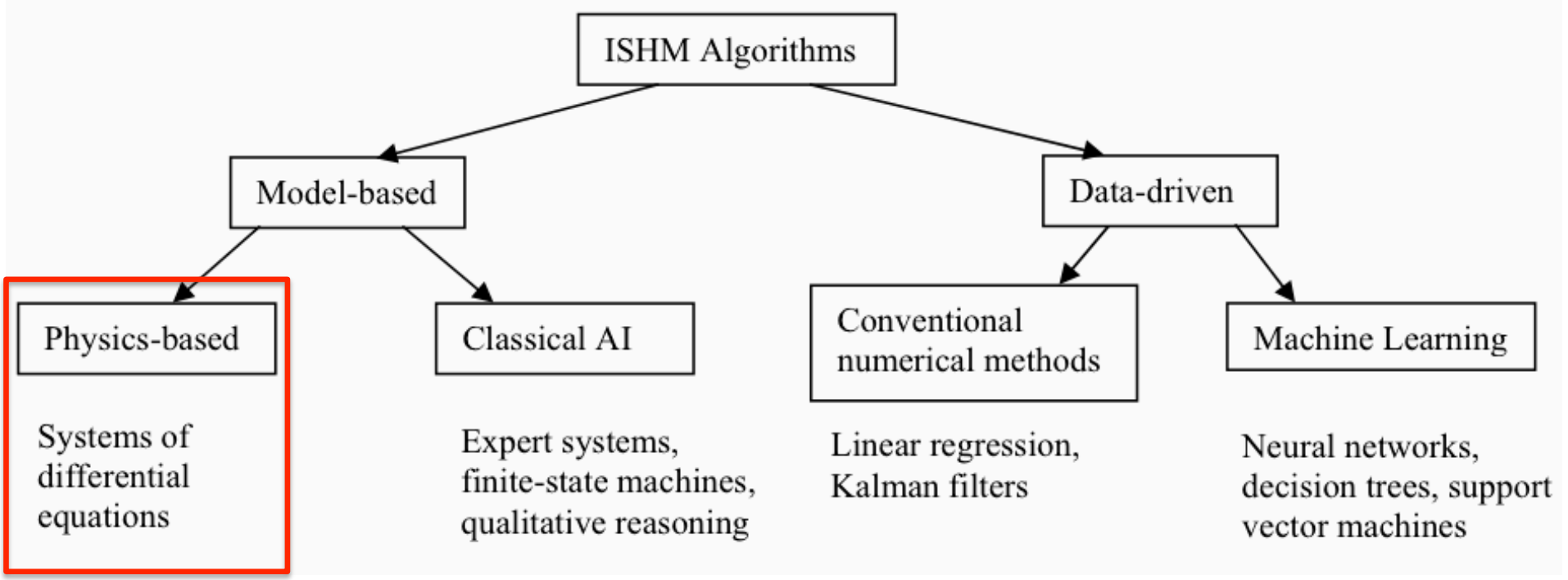
- System testing tests a completely integrated system to verify that it meets its requirements.
  - Hence, the importance of having requirement traceability all the way to system testing (for verification).
  - It also test beyond the bounds defined in the requirements and specification, and therefore the believed expectations of the customer , for validation.
- It can use test-beds with different level of fidelity, such as
  - Hardware-in-the-loop testing
  - Human-in-the-loop testing

What about V&V for PHM?

PHM algorithm classification

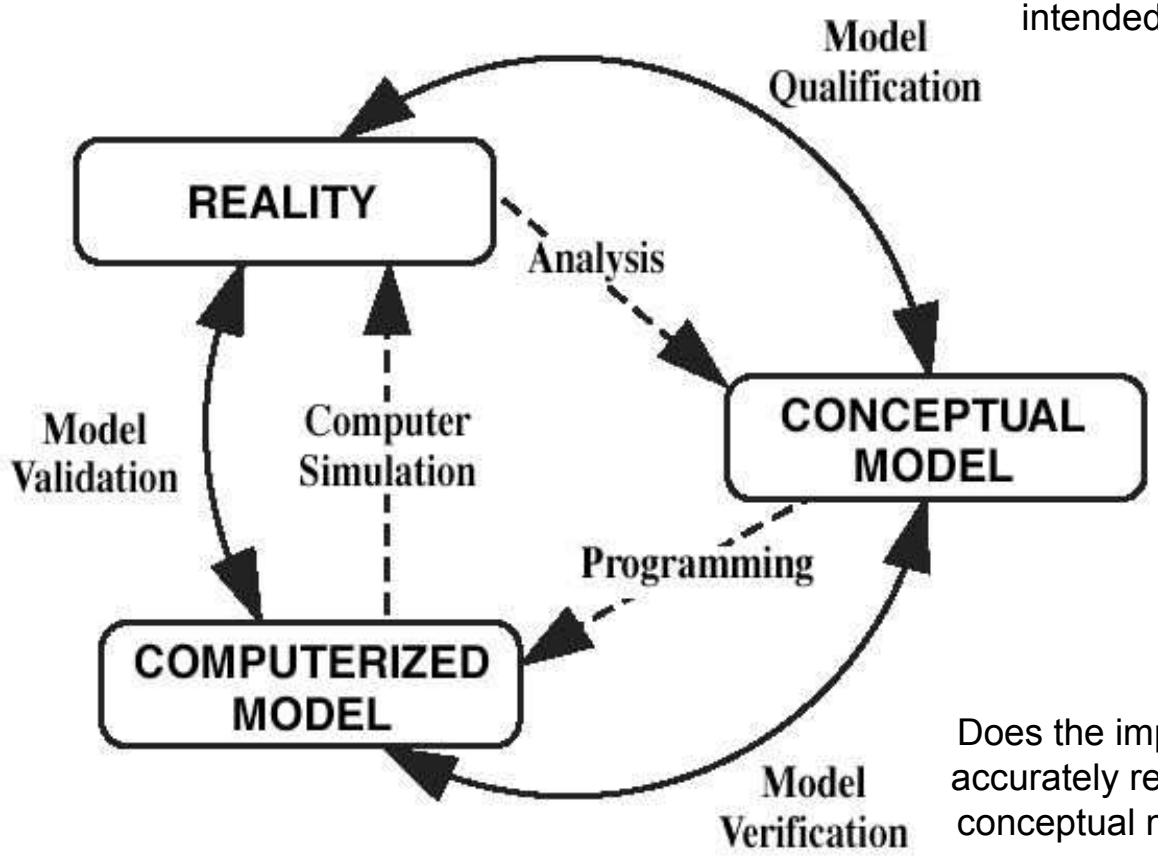


PHM algorithm classification



Physics-based models

Adequacy of the conceptual model with the domain of intended application



Is the computerized model an accurate representation of the real world for the perspective of its intended uses?

Does the implementation accurately represents the conceptual model and its solution?

Physics-based models

- Verification
  - determining that a model implementation accurately represents the developer's conceptual description of the model and the solution of the model
- Ideas
  - Certifiable code synthesis from equations
  - V&V for numerical analysis code (similar to flight control SW)
    - Numerical instabilities (floating point computation problems)
    - Complex data structure manipulation problems (buffer overflow)

Physics-based models

- Validation
  - determining the degree to which a physics model is an accurate representation of the real world from the perspective of its intended uses
    - Correctness of the differential equations
    - Usually done through simulations
    - Stability and convergence of the implemented algorithm
    - Analysis of the off-nominal cases

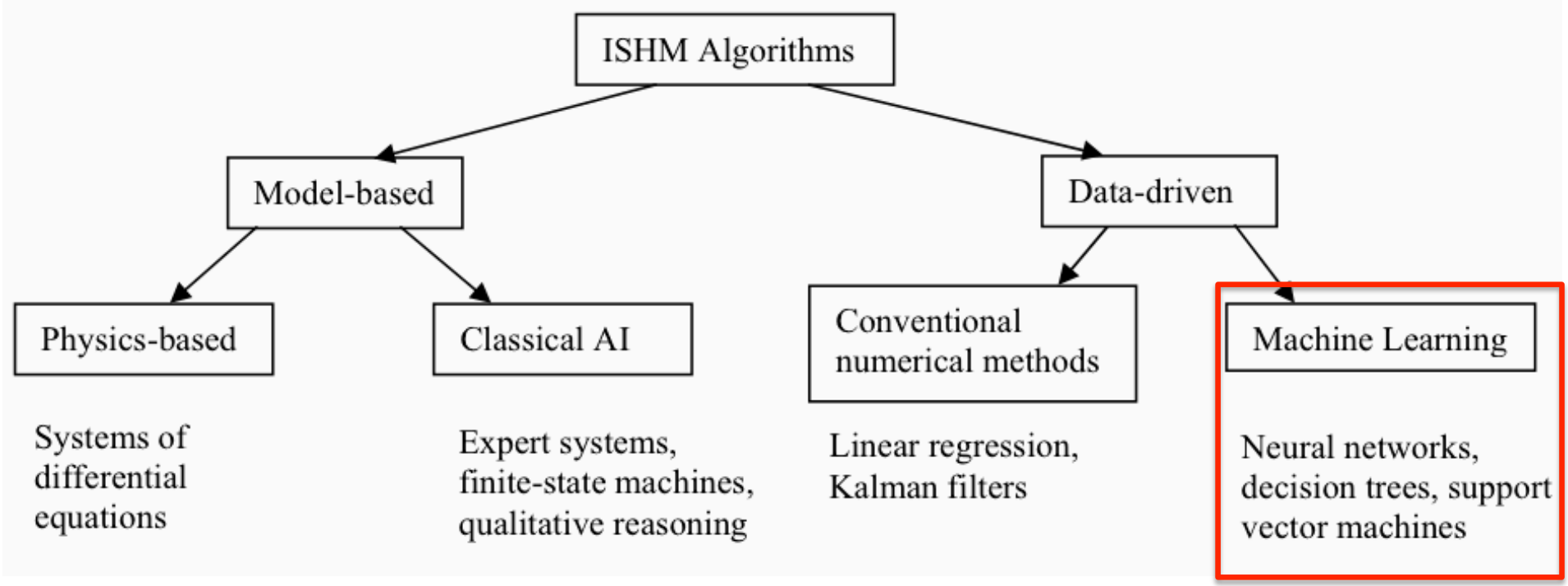
Physics-based models

- Related concepts:
  - Model interpolation
    - means application of a model to conditions bounded by the calibration and validation experiments
  - Model extrapolation
    - outside the range of model parameters tested, or
    - to conditions not tested (i.e., different geometries or boundary conditions),
    - to different physical phenomena for which the model acts as a surrogate
  - Model approximation
    - In the sense that we're looking for a bounding model for a physical phenomenon

Physics-based models V&V references

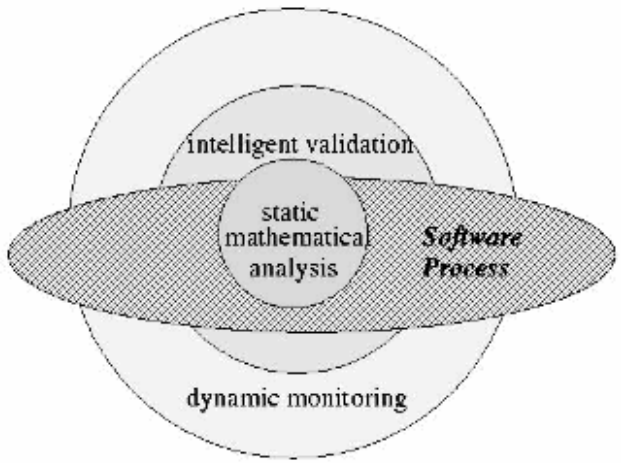
- **Statistical Validation of Engineering and Scientific Models: Bounds, Calibration, and Extrapolation** by Richard G. Hills and Kevin Dowding, SANDIA REPORTSAND2005-1826.
- **A General Strategy for Physics-Based Model Validation** by Didier Sornette, Anthony B. Davis, James R. Kamm, and Kayo Ide, in Proceedings of in Computational Methods in Transport, Granlibakken 2006.

PHM algorithm classification



Machine learning (Neural network) strategy

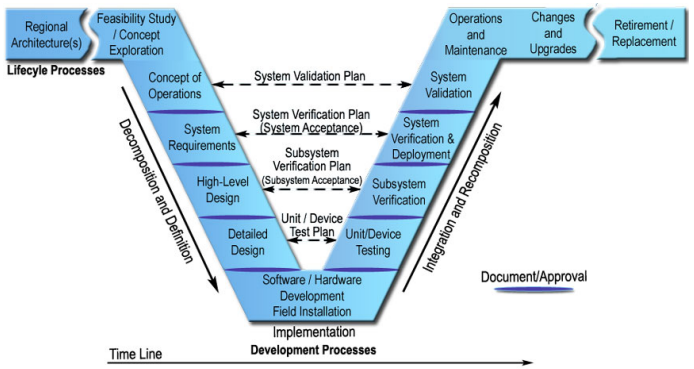
- The core-layer contains rigorous, mathematically sound results concerning robustness, stability, and convergence.
  - Current state-of-the-art provides relatively weak results in form of asymptotic guarantees.
    - Lyapunov proofs of (asymptotic) stability
    - Vapnik-Cherenovis-dimension arguments to reason about the NN's generalization abilities
- Testing techniques to provide convergence guarantees in the required short period of time
  - Analysis of numerical issues
  - tests for convergence and robustness of the system
- Run-time monitoring on NN performance



by Johann Schumann and Stacy Nelson, WOSS'02

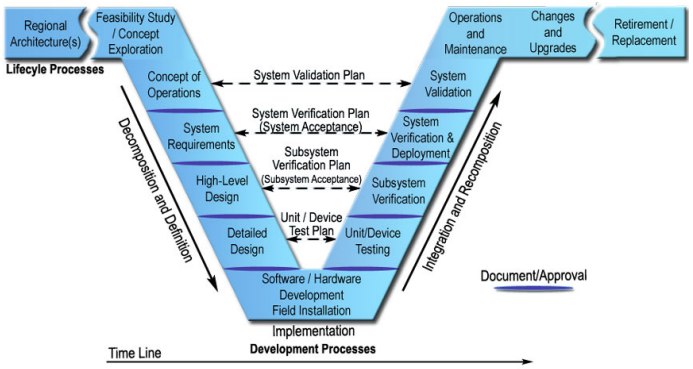
Machine learning

- *Requirements and Design*:
  - Documentation must include the specification for the NN and its architecture.
    - type of NN (feed-forward, Self Organizing Map, etc),
    - the learning algorithm (gradient descent, Least Means Squared, Levenberg-Marquardt, Newton's method, etc.)
    - all parameters of the NN architecture (e.g., number of layers and hidden nodes, activation functions, initialization)
    - a concise description of the inputs and outputs (including units and the expected and maximal range of values) and acceptable errors and training set(s) for PTNN must be provided.



- *Software Detailed Design* must include a description of precise code constructs required to implement the NN, including all data structures and algorithms (e.g., libraries for matrix operations).

Machine learning



- *Unit Testing* must include both black and white box testing for modularized NN code.
- *Software Integration* should verify that the NN interfaces with other software including proper inputs and outputs for the NN.
- *Software Qualification Testing* should ensure that the requirements are sufficiently detailed to adequately and accurately describe the NN.
- *System Integration Testing* should verify that the architectural design is detailed enough so, when implemented, the NN can interface with system hardware and software in various fidelity test-beds.
- *System Qualification Testing* should verify that the system requirements are sufficient enough to ensure that, when implemented, the NN will interface properly with the system in production.

Machine learning

- Other V&V concerns coming from the numerical aspect of a neural network:
  - general numeric properties, like scaling, conditioning, or sensitivity analysis,
  - properties/issues specifically related to the training algorithm (e.g., convergence, termination), and,
  - issues with respect to the actual implementation on a digital computer (e.g., round-off errors, accuracy of library functions).

Machine learning V&V references

- **Guidance for the Verification and Validation of Neural Networks** by Laura L. Pullum, Brian J. Taylor, Marjorie A. Darrah
- **Independent Verification and Validation of Neural Networks - Developing Practitioner Assistance** *By Dr. Laura L. Pullum, Dr. Marjorie A. Darrah, and Mr. Brian J. Taylor, Institute for Scientific Research, Inc.*, Software Tech New
- **Toward V&V of neural network based controllers** by Johann Schumann and Stacy Nelson
- **Validating A Neural Network-based Online Adaptive System** by Yan Liu, Dissertation submitted to the College of Engineering and Mineral Resources at West Virginia University, 2005
- **Verification and Validation of Adaptive and Intelligent Systems with Flight Test Results** by John Burken and Dick Larson, UCAUV 2009
- **Validation and Regulation of Medical Neural Networks** by David M. Rodvold. Molecular Urology. December 2001, 5(4): 141-145.

Conclusion: need more research?

SSAT Project under Aviation Safety program

Level 2 – *System Level*

Goal -- Validated multidisciplinary tools and techniques to ensure system safety in NextGen and enable proactive management of safety risk through predictive methods.

- **SSAA 2.1 Technical Challenges**
- **SSAA 2.2 Systems Analysis**
- **SSAA 2.3 *Partnerships and Outreach***
- **SSAA 2.4 Research Test and Integration**

Level 3 – *Themes*

- **SSAA 3.1 Verification & Validation of Flight Critical Systems**
- **SSAA 3.2 Data Mining and Knowledge Discovery**
- **SSAA 3.3 Human Systems Solutions**
- **SSAA 3.4 Prognostics and Decision Making**

Level 4 – *Foundational*

- **SSAA 4.1.1: Argument-Based Safety Assurance**
- **SSAA 4.1.2: Authority and Autonomy**
- **SSAA 4.1.3: Distributed Systems**
- **SSAA 4.1.4: Software Intensive Systems**

V&V of Prognostics Algos

- **SSAA 4.4.1: Decision Making under Uncertainty**
- **SSAA 4.4.2: Diagnostics**
- **SSAA 4.4.3: Prognostics**
- **SSAA 4.4.4: Software Health Management**

"Validated, proactive solutions for ensuring safety in flight and operations"

Technical Challenges: PDM

Prognostic Algorithm Design for Safety Assurance.

Explore designing new prognostics algorithms that are verifiable, thus removing obstacles to their certification and enabling their deployment by industry to take advantage of their safety benefits. (FY 25)

Goal:

Remove obstacles to the certification of prognostic algorithms. The non-linear and non-deterministic nature of prognostic algorithms requires industry to perform more costly, intensive testing than on traditional technologies.

Benefits:

- New class of verifiable prognostic algorithm
- Reduced the cost to deploy prognostics algorithms