# Advanced Verification and Validation Procedures and Tools for the Certification of Learning Systems in Aerospace Applications

**Stephen Jacklin**
**NASA Ames Research Center**

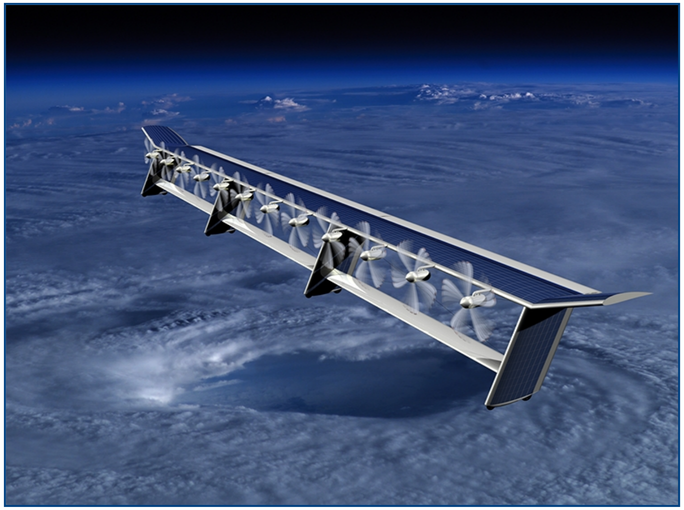
Aerospace Control and Guidance Systems Committee Meeting No. 98
October 11-13, 2006
Williamsburg, VA

# Presentation Outline

- **Adaptive control system architecture**
- **Learning systems and V&V problems**
- **V&V methodologies, tools, and procedures for (partial) solution**
  - Emphasis on methods that may be helpful toward meeting future certification requirements

# Why Adaptive Flight Control?

- **Adaptive flight control systems are critically needed**
  - To retain vehicle handling qualities in the event of damage
  - To maintain robust control in unknown or changing conditions
  - To enable autonomous flight control of aircraft and spacecraft
  - To adapt to changing mission requirements

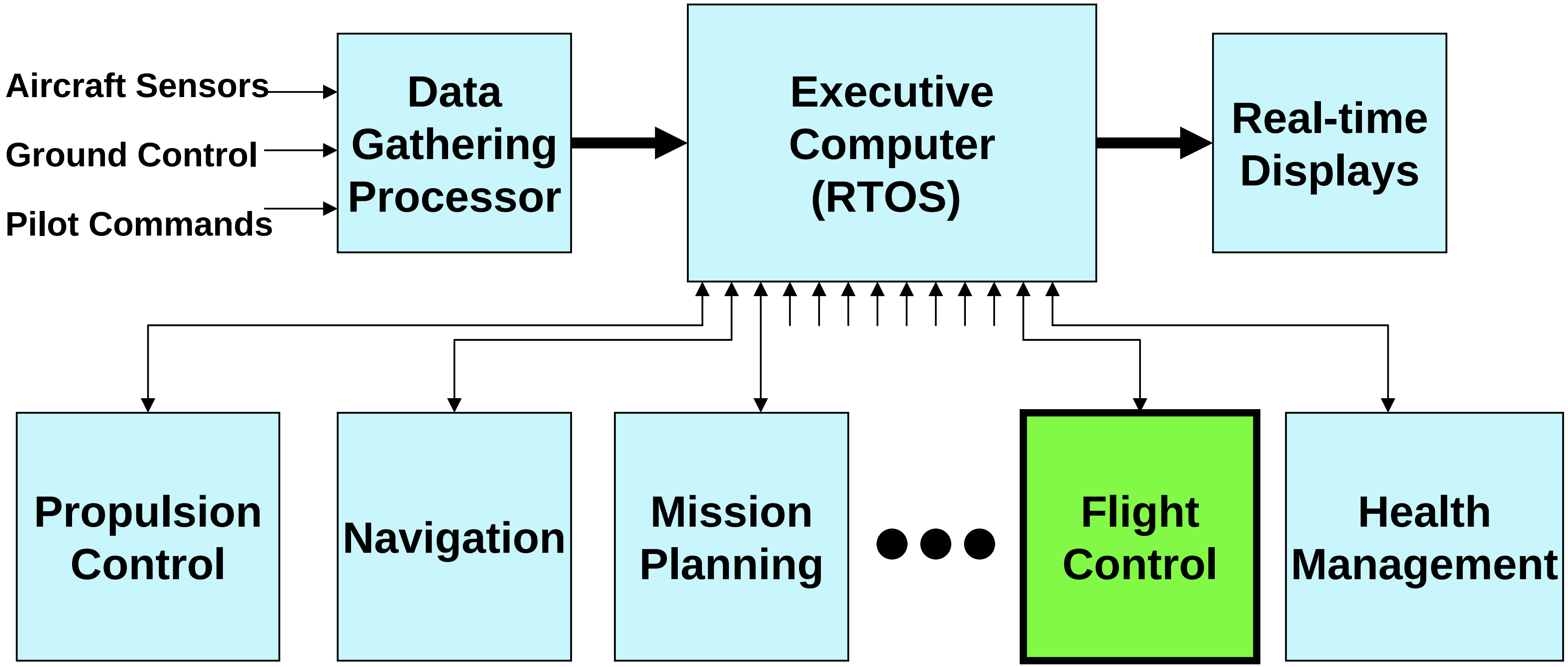



- **Many compelling applications**
  - Neural adaptive flight control
  - Fault tolerant flight control
  - Damage adaptive control

# Verification and Validation of Adaptive Controllers

- **Adaptive controllers won't be part of the future unless they can be proven to be highly safe and reliable**
- **Rigorous methods for adaptive system V&V are needed**
  - To ensure control system failures do not occur
  - To ensure control systems perform their intended function
  - To ensure against unintended or undesired functionality
- **FAA certification regulations do not yet address adaptive systems**
  - RTCA DO-178C still under development
- **V&V of adaptive control systems for aerospace applications complicated by hybrid systems**

# Outer-Loop Controller Architecture



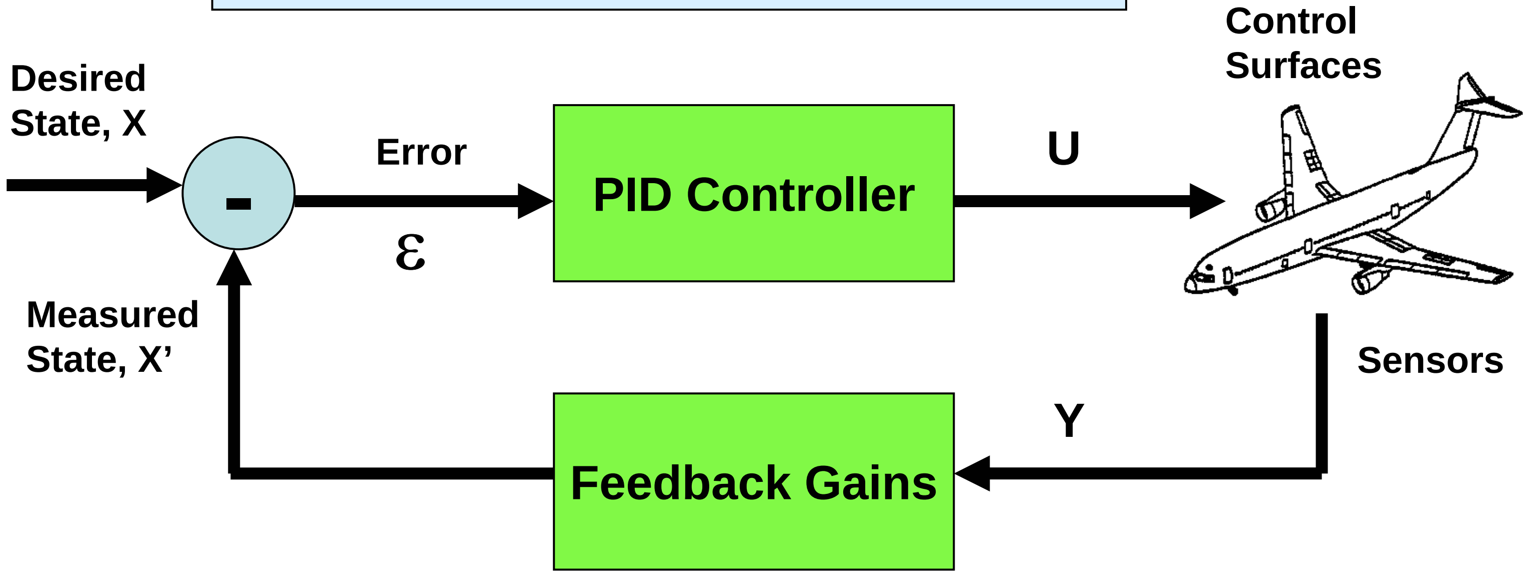

Outer-loop controller (Real-Time Operating System) controls lower-level computers and controllers

# Inner-Loop Controller Architecture

- **These are continuous adaptive control systems**

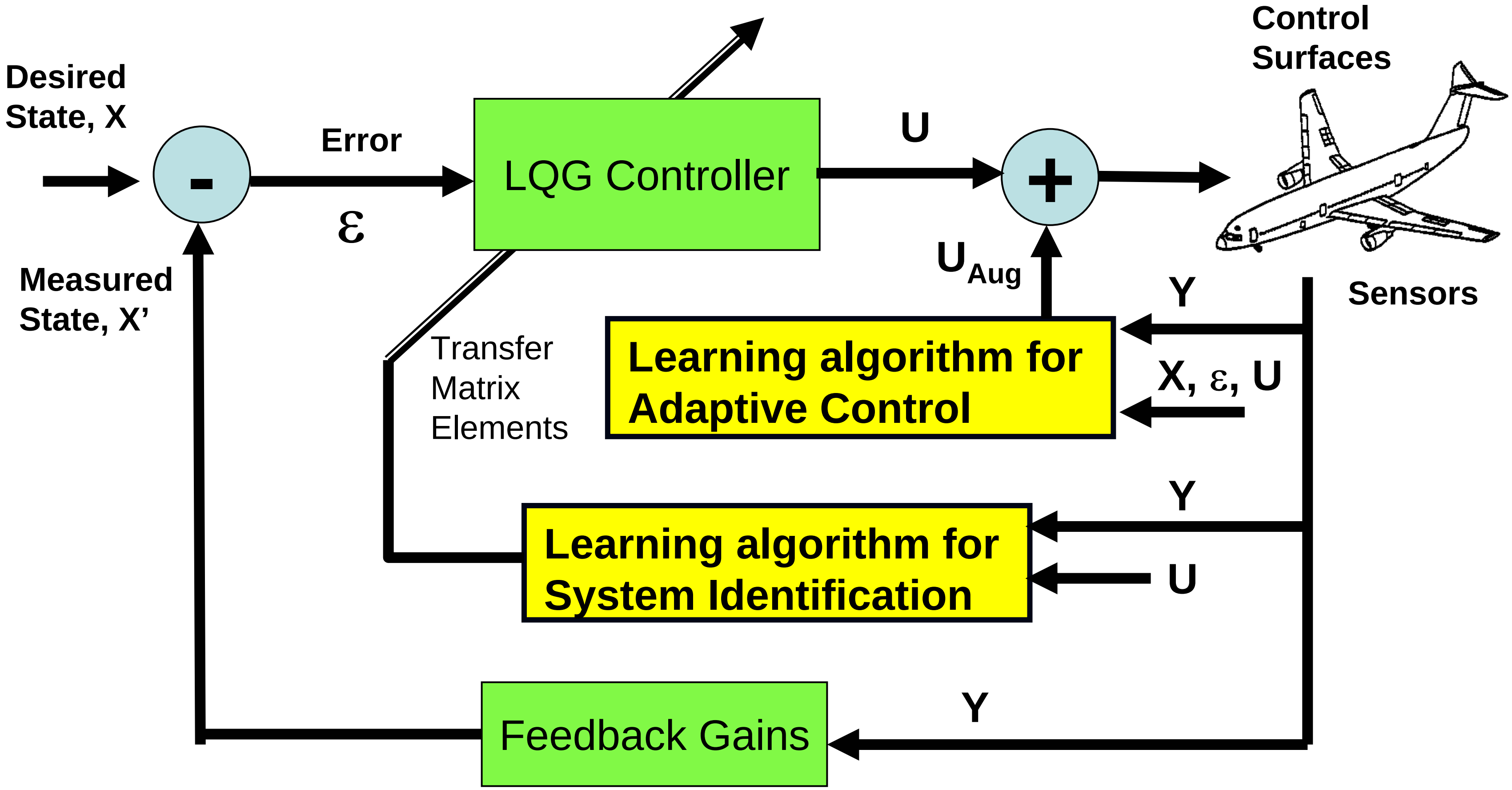
$$\frac{dx}{dt} = Ax + Bu, \qquad y = Cx + Du$$

$$u = K_P \varepsilon + K_I \int (\varepsilon) dt + K_D \left( \frac{d\varepsilon}{dt} \right)$$



# Many Ways to Introduce Learning

**Control augmentation and system identification are two**



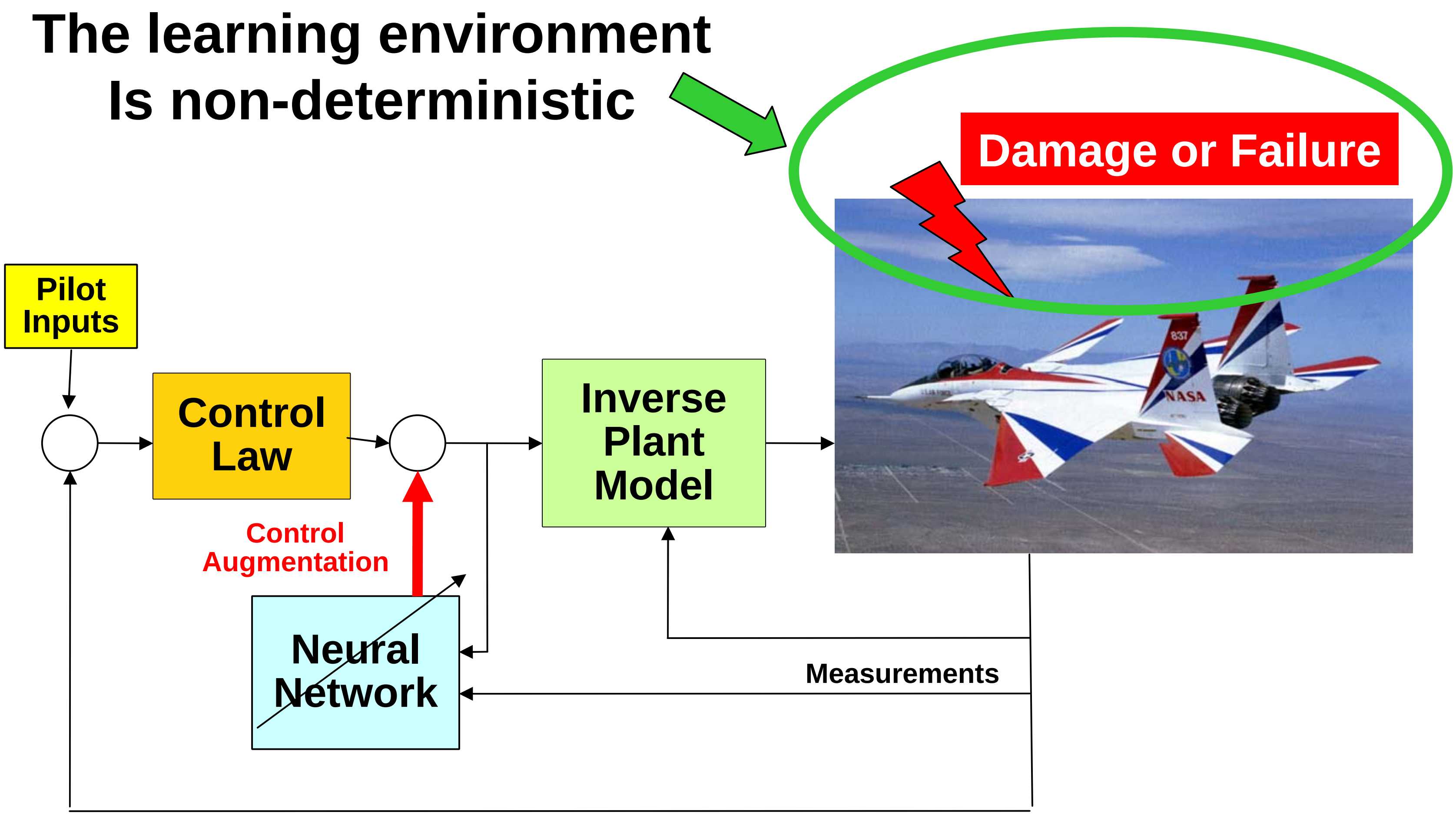
# V&V Problems for Learning Systems

- **Learning systems face some special problems for certification**
- **Continuous System Problems**
  - Non-determinism
  - Learning accuracy
  - Learning stability
- **Finite State System Problems**
  - Shear size of the programs ( > 20 million SLOC)
  - Coverage testing
  - Regression testing of modified software

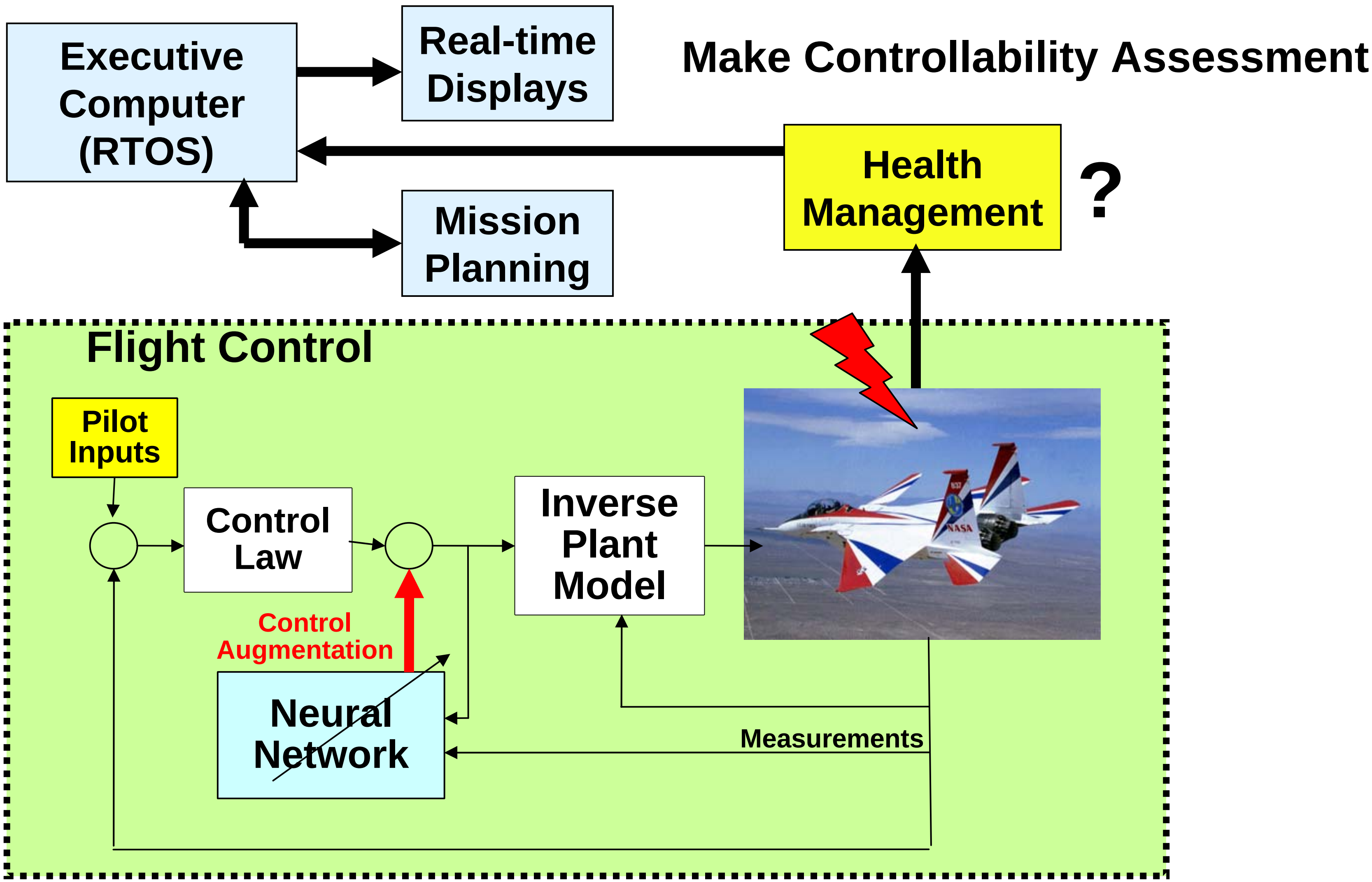
# Non-Determinism

- **An important certification concern is that fielded software duplicate its behavior as tested**
- **A reasonable requirement for adaptive systems comprised of mathematical algorithms**
  - Should behave in a predictable manner if started from the same initial conditions and then given a specified input history
  - But, a valid concern is the need to restart controller in flight
- **Only partially reasonable to account for non-determinism of the environment**
  - Adaptive systems intentionally designed to maintain controller performance in unforeseen situations
    - **Vehicle damage**
    - **Unexpected operating conditions**

# Most Controllers Are Deterministic



# Health Monitoring Can Help

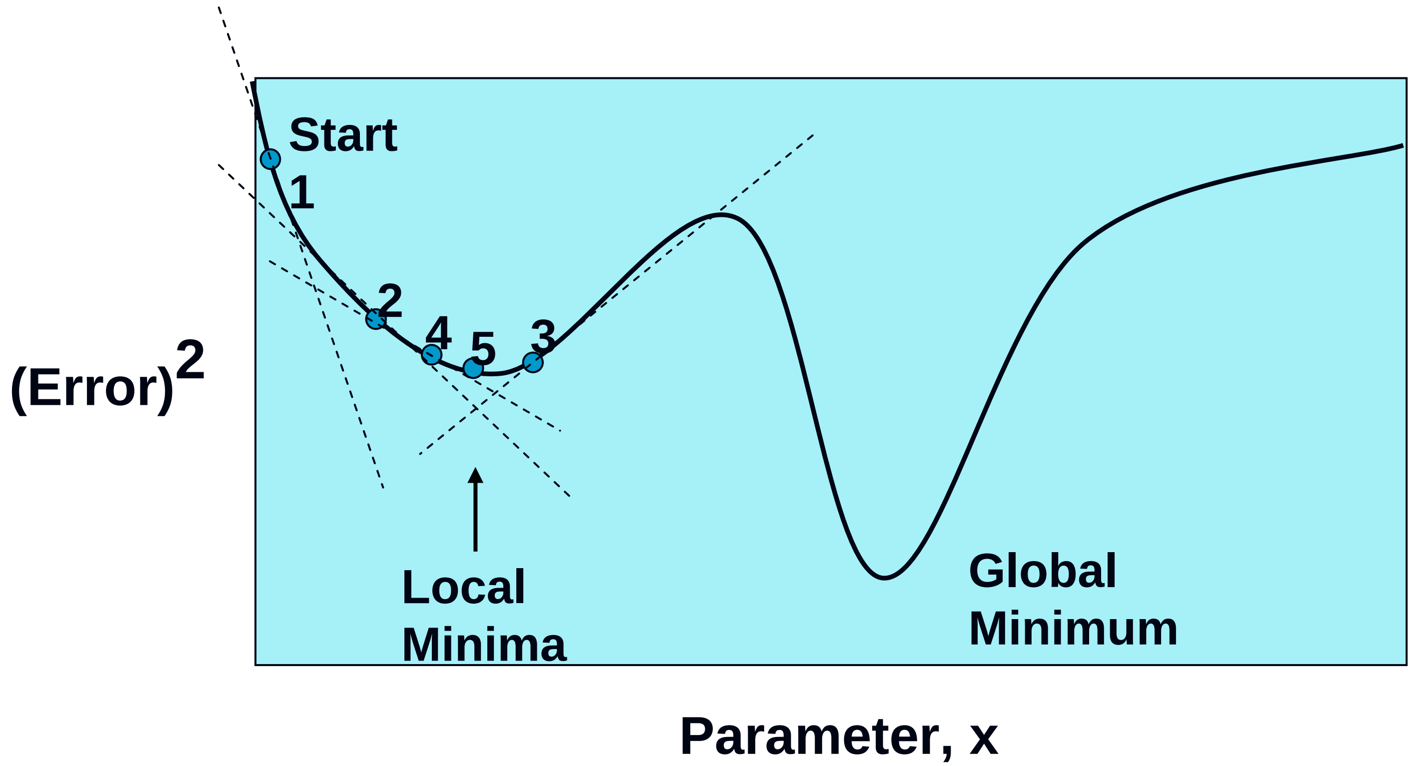


# Learning Accuracy Issues

- **Most fundamental problem in adaptive control is "Can the inner-loop systems learn ?"**
- **Many simple gradient-based search methods abound (**steepest descent, back propagation**)**
- **Methods that use 2$^{nd}$ derivative (Hessian) are faster (**Conjugate Gradient, Levenberg-Marquardt)
- **Currently, no analytical or formal method exists to guarantee learning convergence to the globally optimal values within a given time**

# Local Convergence Problem

- **Learning process may converge to a local optimum rather than the global optimum**



# Subtle Convergence Problems

- Suppose we have an inverse controller whereby we desire to adaptively identify the inverse transfer matrix "T" at each step

$$\Delta u = [T]\Delta y$$

$$T_{k+1} = T_k + Update$$

$$Update \approx K_{Learn}\left(\Delta u_k - [T_k]\Delta y_k\right)$$

- What happens as

$$\left(\Delta u_k - [T_k]\Delta y_k\right) \longrightarrow \text{Zero ?}$$

# Sensor Noise Corrupts Learning

- **Answer: although initially good for control, measurement noise soon corrupts the system identification process if left on-going**
  - Small changes in the output, when dominated by noise, obscure the effect of small control changes

$$\left(\Delta u_k - \left[T_k\right]\left(\Delta y_k + noise\right)\right)$$

- **Persistent excitation or disabling of learning when not required, or both, are two possible solutions**

# Inner-loop Controller Stability

- **Stability of adaptive inner-loop controllers is well understood**
  - Bode, Root Locus, Nyquist Chart, Nichols, etc.
- **Stability of controllers with learning is a more difficult problem**
  - Liapunov theory can show learning errors are bounded, but applying Lyapunov theory is difficult
  - Liapunov theory can’t explicitly calculate the maximum learning error bounds from theory without incorporation of the real, detailed system dynamics

# Simulation Remains a Vital Step for Convergence and Stability Testing

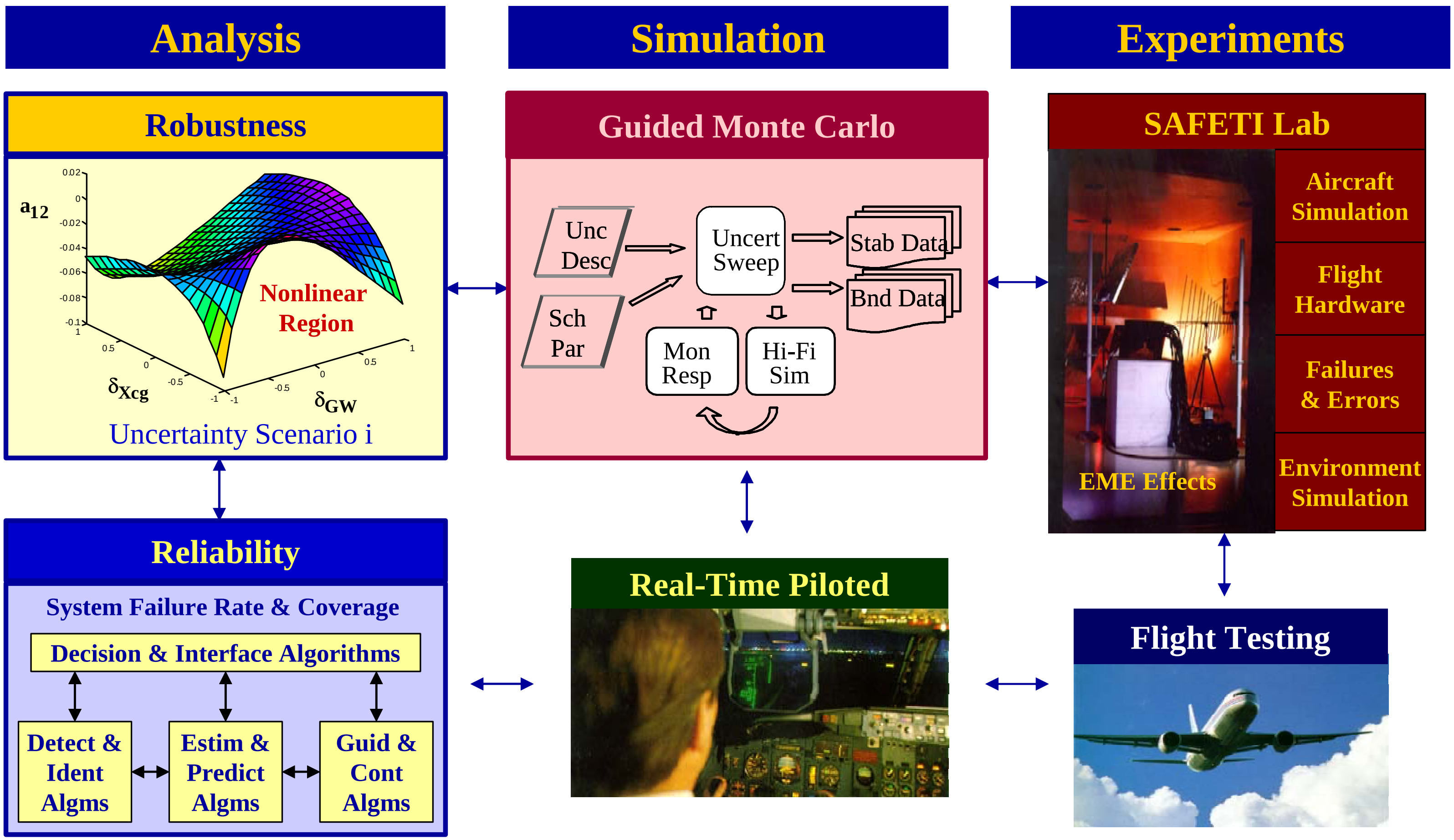
- Traditional V&V starts with analysis of convergence and stability, but then usually rapidly transitions to simulation

| Model Fidelity | Simulation Type & Test Bed |
|---|---|
| Low | Desk Top Computer (Matlab-Simulink) |
| Low-Medium | Work Station (nonlinear models) |
| Medium | Simulation with target flight computer |
| Medium-High | Hardware-in-the-loop (cockpit + FC) |
| Medium-High | Aircraft-in-the-loop simulator (ground) |
| High | Motion-based simulation |

# Simulation is a Useful Tool

- Explore initial concept feasibility studies
- Evaluate and compare learning algorithms
- Find stable learning rule update gains
- Evaluate overall controller stability
- Determine how much learning happens at each step of the simulation
- Evaluate learning convergence tendencies and convergence speed
- Evaluate ad hoc fixes that are difficult to analyze
- Evaluate human-machine interactions

# Integrated Simulation Labs



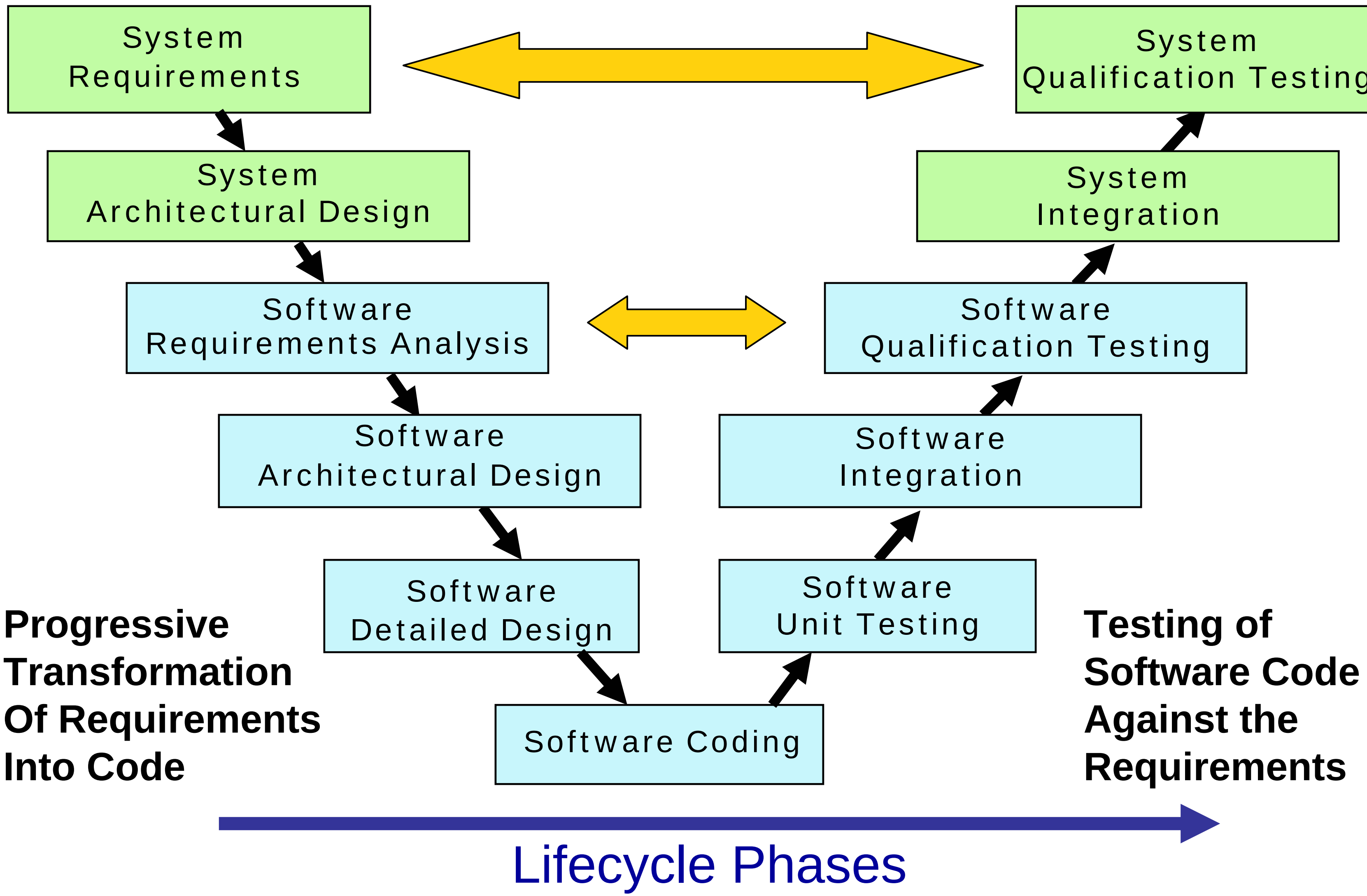

## NASA Langley SAFETI Laboratory

# Problems with Simulation and Testing

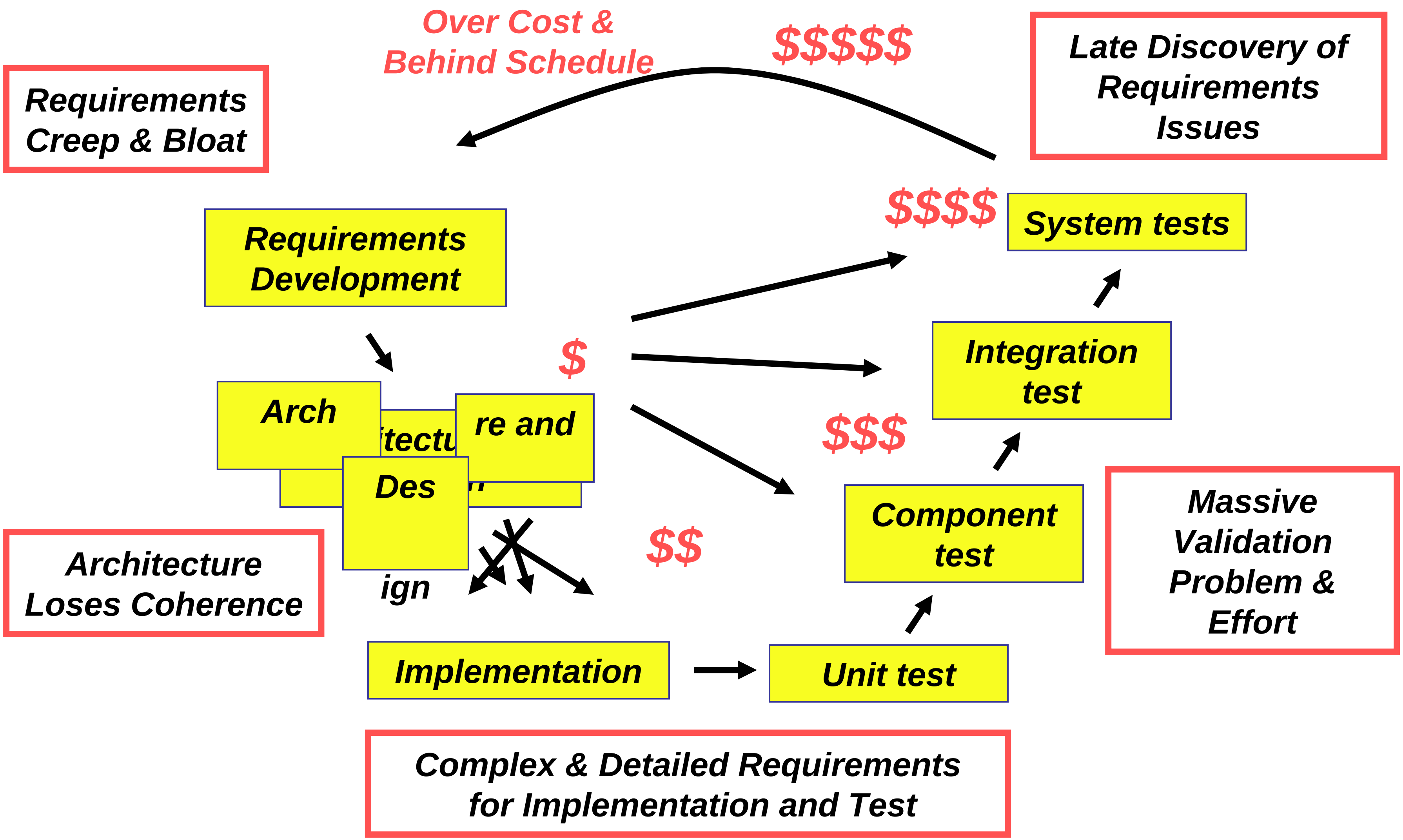
- **Testing can never prove the absence of errors**
  - Totally relies on the engineer to identify all test scenarios
- **Providing adequate test case coverage is time-consuming and hard to prove**
  - DO-178B requires complete MC/DC testing to evaluate all possible outcomes given all possible inputs
- **High-fidelity simulation gets progressively more expensive as fidelity increases**
  - Cost and lack of time may cause some desired code modifications to be rejected, and code function limited instead

What else can be done?

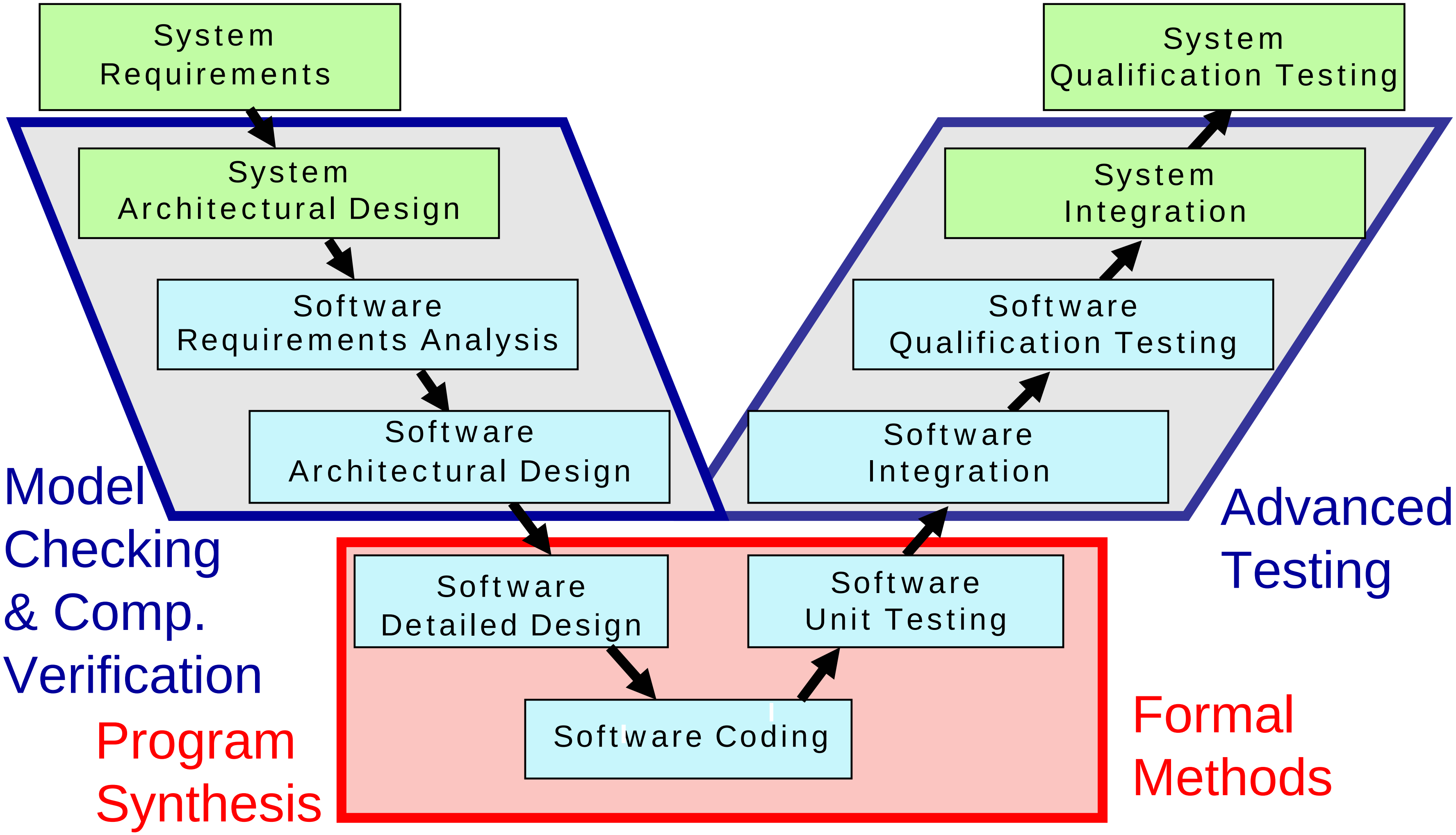
# Look at the Software Development Process



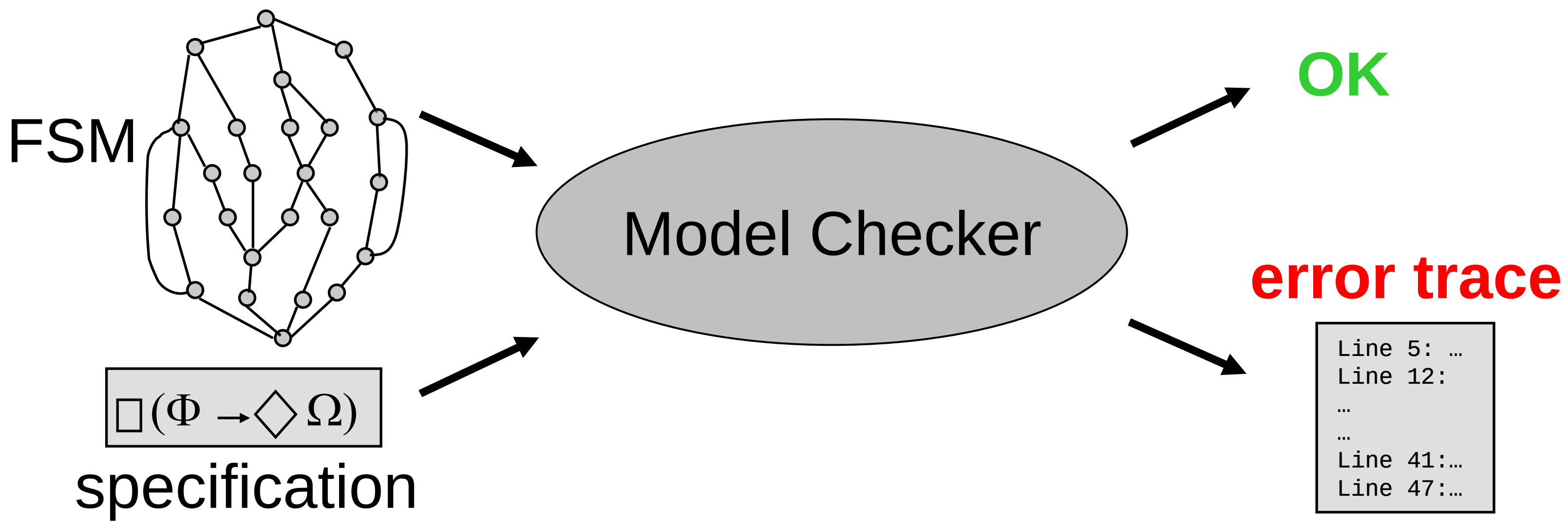
# What Goes Wrong?



# Automated V&V Processes



# What is Model Checking?




- **Programs represented as individual state machines with V&V properties specified in temporal logic**
  - exhaustively explores all executions in a systematic way
  - handles millions of combinations – hard to perform by humans
  - reports errors as traces and simulates them on system models

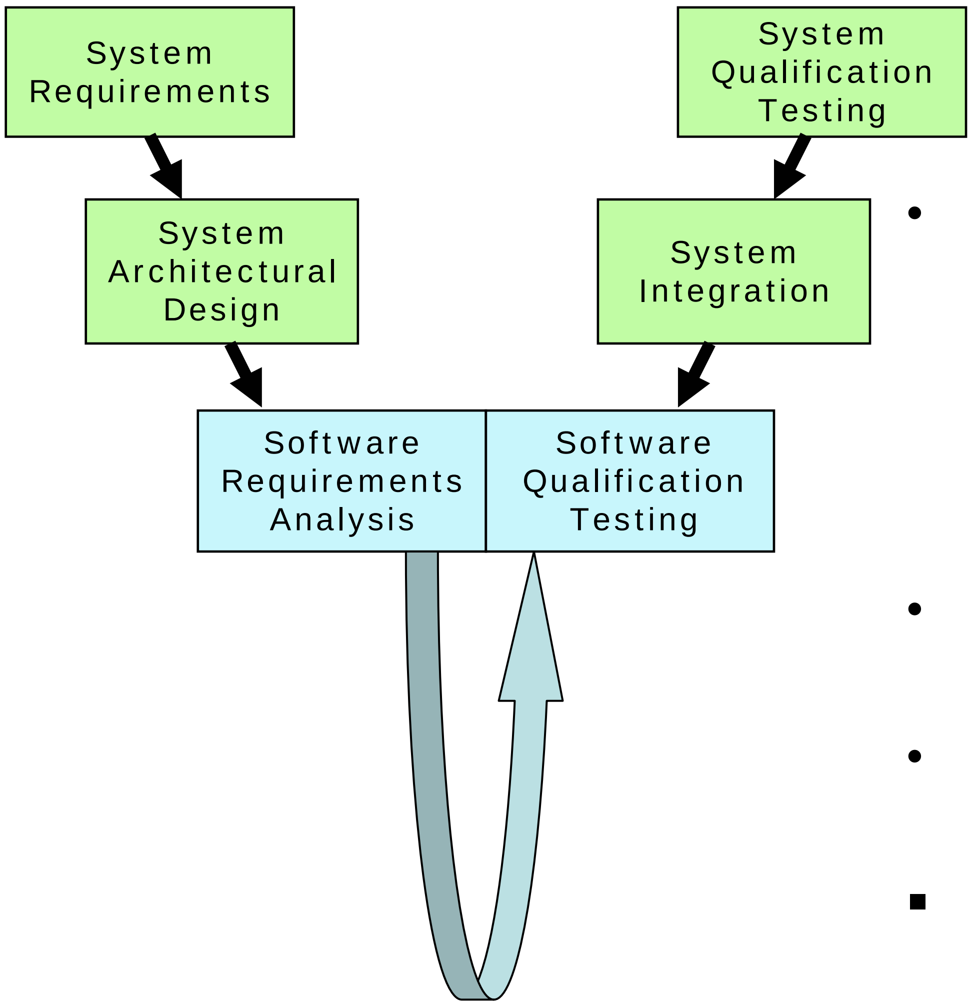
# Model Checking

- **Model checking programs can explore all reachable (finite) states (JPF2, NuSMV, SPIN)**
  - **Good tool for mode logic V&V of outer-loop controller**
  - **Can analyze up to about $10^{120}$ states (paths)**
  - **Very useful for analyzing multi-threaded programs**
- **Symbolic analysis of code allows analysis of all feasible execution paths for any valid input data**
  - **Inputs are symbolic and hence covers *all* concrete inputs**
- **Inputs are constrained by the code structure rather than by the tester’s imagination**
- **A model allows rapid V&V of software patches and revisions**
- **Model checking may be extended to do V&V continuous domain of adaptive systems**

# Compositional Verification

- Model checking by itself does not scale well due to the large computer memory requirements
- Compositional verification allows large systems to be logically divided at their interfaces
  - System modeled using language such as UML 2.0
  - Each subsystem can be checked individually
  - All interfaces are verified in the process
- A large benefit is that integration errors are detected early - at the design phase
  - Hundreds of pages of requirements can be analyzed for early detection of defects, prior to coding
  - Allows many problems to be identified up front as a result of just doing the modeling
  - Model facilitates quick analysis of software upgrades

# Program Synthesis: From V to Y




- **Code generation can be used throughout the software process**
  - **Model analysis**
  - **Coding (multiple platforms)**
  - **Testing (continuous domain)**
- **Generate *efficient* and *documented* code**
- **Automated support for safety certification**
- **Reduce schedule, costs, errors**

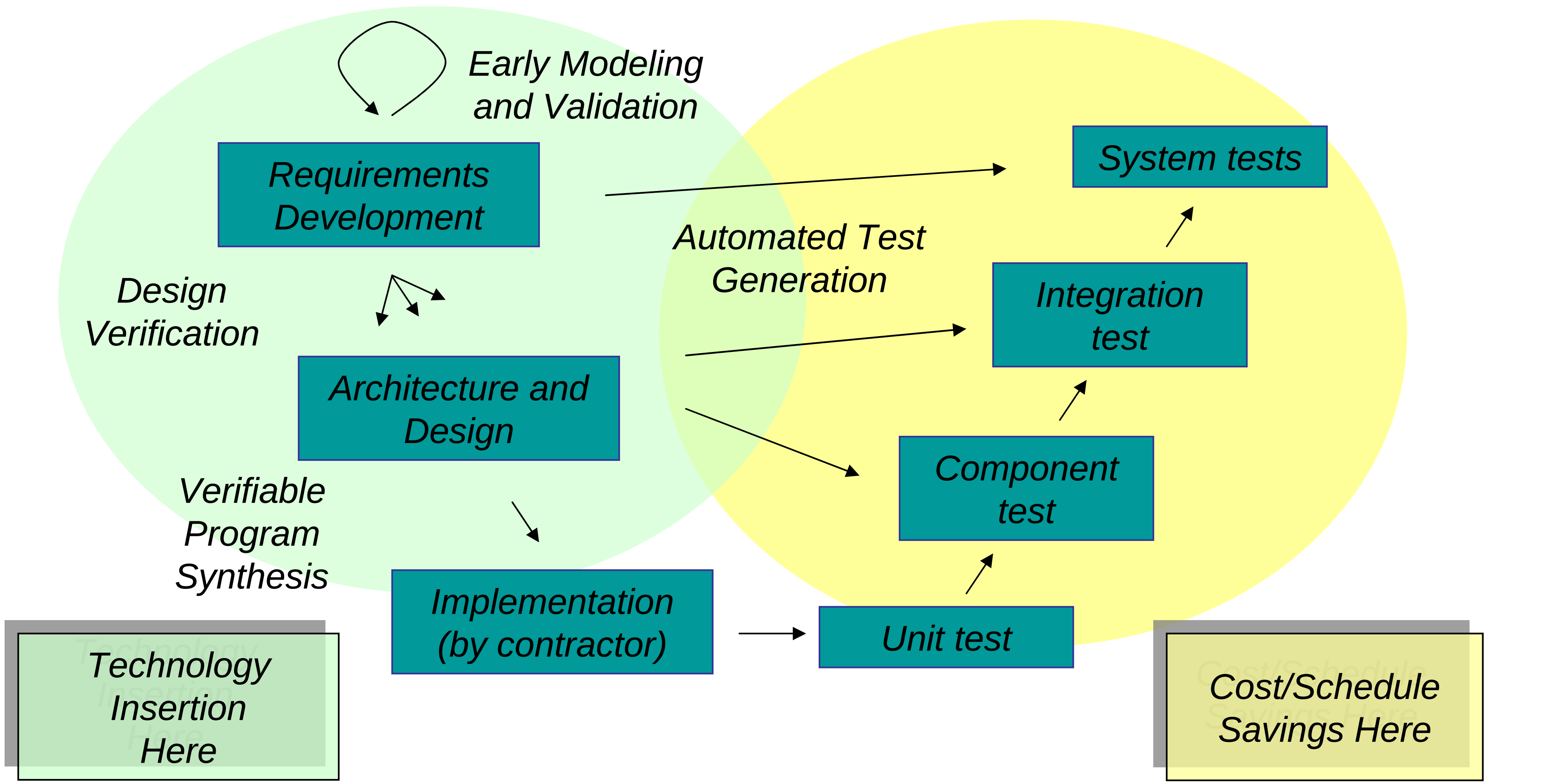
# Automatic Generation of Certified Code

- These tools can generate adaptive software programs that are easier to certify
- AutoFilter for Kalman Filter generation
  - Ewen Denney & Bernd Fischer – NASA Ames
- Safety Critical Application Development Environment (SCADE) - Esterol
  - Used for auto-pilot code generation from precise formal rules and specifications
  - Also generates readable and traceable code
- Boeing Zbra compiler
  - V. Santhanam, Boeing Wichita
  - Subset of Ada language for safety-critical applications
  - Built-in anomaly checker to look for real-time code problems

# Automated Code Review

- **Static analysis is a formal method used to look programming errors in an automated way**
- **Coverity, Polyspace, Parasoft, Clint, UNO, CGS, …..**
- **Can catch a lot of relatively simple mistatkes**
  - Buffer over-runs -- Over-flows
  - Un-initialized variables -- Unreachable code
  - Out-of-bounds array ref -- Mixed mode comp
  - Wrong use of formal parameters
- **Very fast, but can generate many false-positives**

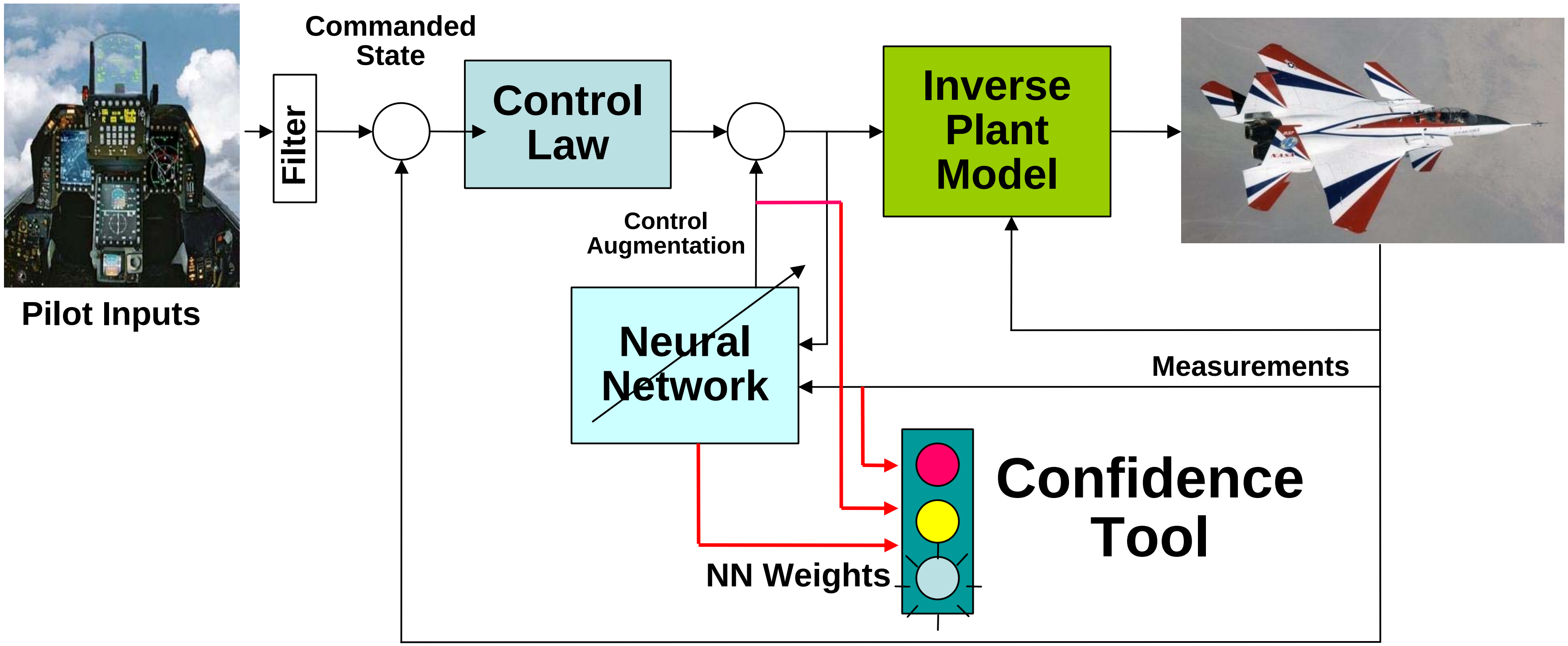
# Automated Methods Should Pay for Themselves



# Tools for In-Flight Software Assurance

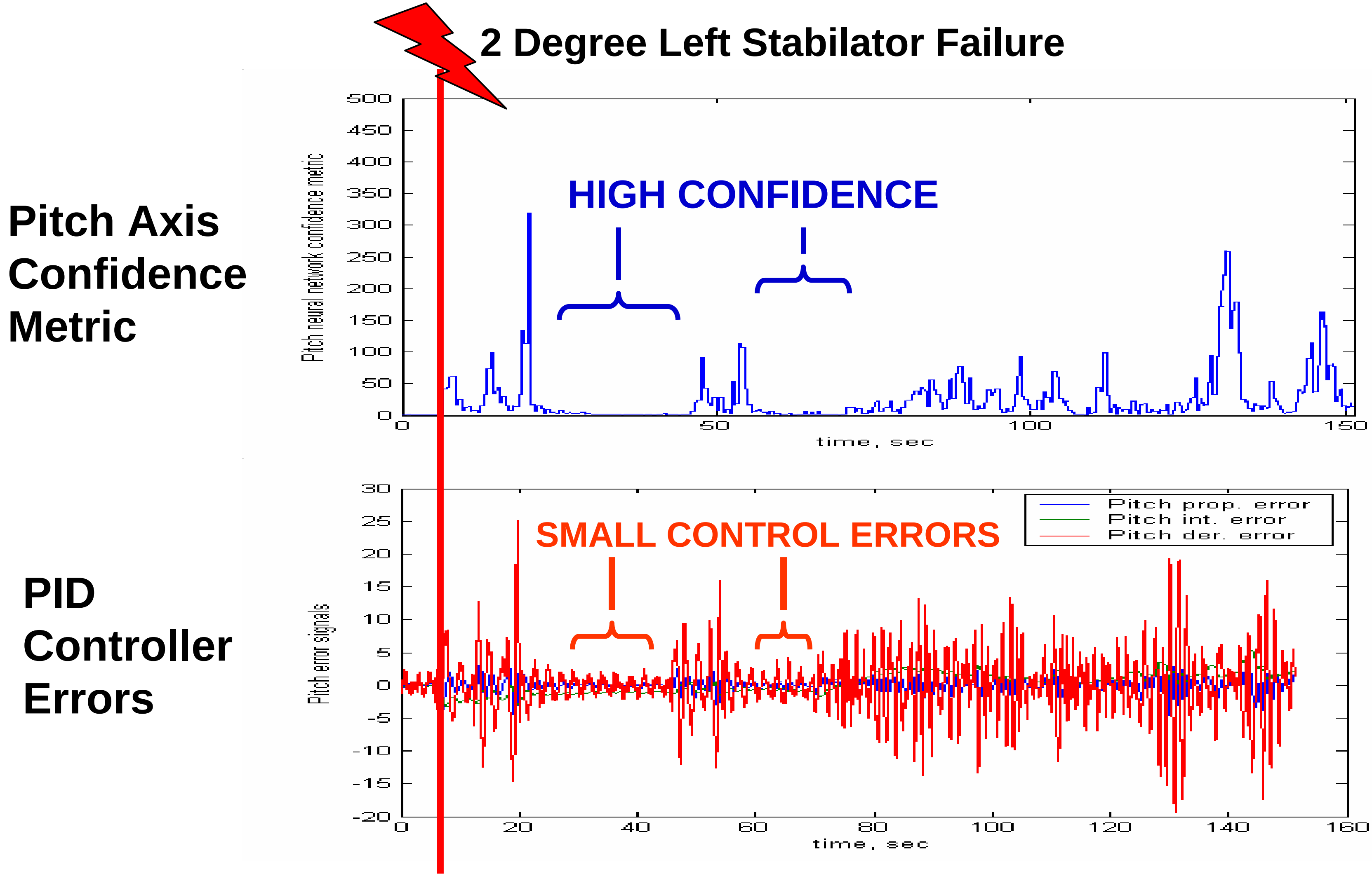
- **Health monitoring likely to be important**
- **Rule Extraction methods (NNRules, RuleX)**
  - A type of safety wrapper
  - English readable rules about the trained neural network (IF {cond 1 AND cond 2 ….} THEN Result)
  - Shown useful for pre-trained neural networks
- **NASA Ames Confidence Tool is an example**
  - Based on Bayesian probability theory
  - Looks at variance of neural network parameters to judge functionality of network …. without knowing correct value of network weights

# Tested on NASA Dryden IFCS F-15




**The Confidence Tool, based on a Bayesian approach, provides a measure of how well the neural network is performing at the moment**

# Confidence Tool Results



# Development of Process Guides

- **Necessary to ensure the methodologies developed for learning systems can become a well-defined process**
  - Critical in shaping new certification requirements, e.g., DO-178C, that may address adaptive systems
  - Document a body of knowledge with application examples and lessons learned
- **Should be comprehensive**
  - Outer-loop finite state executive programs
  - Inner-loop learning controllers
  - Clear guidance on tool utilization and options / benefits

# Summary Points

- **V&V tools and procedures are being developed that may be used to address critical certification issues**
  - Simulation remains the backbone of verification, but …
  - Automated tools for finite state executives and continuous systems are being developed, and
  - Tools to prove learning convergence and stability are available
  - Health management critically needed for learning systems
- **More work remains to be done**
  - Application of model checking to outer-loop controllers
  - Extend model checking to continuous systems
  - Pursue development of synthesis tools for certification
  - Development tools to prove on-line software assurance
  - Develop certification process guides for learning systems

# Contact Information

Stephen Jacklin
Intelligent Systems Division

NASA Ames Research Center
Moffett Field, CA
Mail Stop: 269-2

Phone: (650) 604-4567

Email: stephen.a.jacklin@nasa.gov