



# When Software and Controls Collide

**Alwyn Goodloe**

**NASA Langley Research Center**

**29 October 2011**



# Motivation

- **I recently spent about six months on a FAA task investigating how the use of adaptive systems in civil aviation impacts software assurance**
- **I have had a number of interesting discussions:**
  - Discussing software assurance with controls people yielded a “controls discussion”
  - Discussing controls with software people yielded a “software what are controls discussion”
- **Similar issues arose with the DO-178 Model-Based Design Supplement**



# Open Discussion

- **The purpose of this talk is to drive a discussion about the interaction between the controls and software community**
- **Please feel free to speak up**
  - Tell me when I'm wrong
  - Add your own comments
  - Take the discussion off in a different direction



# Confession

- **I come at things from a software perspective**
- **I've seen a transfer function or two, but**
  - I have never used Simulink
  - I think anything of interest can be expressed in the  $\lambda$ -calculus
    - Yes, I like functional programming 😊
  - I have never taken a class in dynamics
    - Unless we are talking about operational semantics
  - I think logic, universal algebra, lattice theory, and number theory is applied math
    - Differential equations? Why do that pure math stuff!
  - When you say “the plant” I think of something hanging on my balcony



# Software Folks View of Control Engineers

- **Don't get discrete math**
- **Why do that automata thing instead of an ODE**
- **Think all software = Matlab/Simulink**
  - You push a button and produce C
    - “There was this grad student a couple of years ago who could actually wrote some C code”
- **Think computers are things that execute Simulink**
- **Assume real numbers and floating point are the same**
  - Actually we software people are guilty of this too



# Control Engineers View of S/W Folks

- **S/W folks just code**
- **S/W folks don't think about physical world**
  - Actually, we usually don't
- **S/W folks don't know any applied math**
  - Hey, I said we know some number theory
- **Shouldn't the S/W people be doing numerical analysis**
- **If the S/W guys can't answer my Matlab questions what good are they?**
- **Simulation is verification so why don't software people see the light here**
  - I verified it in Simulink thus it's safe



# Turning On Us Software Folks

- **Our verification tools are aimed at our own algorithms and data structures**
  - Discrete math. Graphs, trees, etc.
  - Often requires spec in temporal or first-order logic
- **Functional correctness is often needed in control software in civil aviation**
  - More than safety
- **Real-time not discrete time**
- **Model checking, SMT solving, abstract interpretation pretty much falls down in this area**



# In Civil Aviation Avionics

- **The problem is usually the physically possible, but logically unanticipated**
- **You must test with real hardware and any and all conceivable hardware failures**
- **Our formal tools often fall down and simulation is not helpful so it is still about testing**



# More On Our Tools

- **Hybrid systems tools have had some success, but seem stuck on very simple linear differential equations and not so helpful when dealing with actual code**
- **Interactive theorem provers are very flexible, but very very steep learning curve**
  - We can do controls in PVS or Coq, but does that help
- **Control engineers are only going to use Simulink so is there a return on building tools aimed at this domain requiring different background**
  - Alur and others are looking at applying formal methods to the simulations



# Outline

- **I will give a briefly discuss some of the work I did on the FAA task**
- **A brief introduction to the DO-178 process governing civil aviation**
- **The impedance mismatch in talking with controls folks about civil aviation software**
- **What we found out and still need to learn**
- **Some observations about the Model-Based Design supplement**



# Adaptive Controls Rock

- **Aircraft can safely operate after sustaining severe damage**
- **The control theory is way beyond gain scheduling**
- **How to show it is safe**
  - Lyapunov analysis
  - Simulink simulations
    - Monte-Carlo simulations
- **How controls people produce software**
  - Autocode from simulink
- **Build, test, fly**
- **DoD processes are somewhat like this**



# Is it ready for civil aviation?



**No Way In \*&%\$ That's Going on MY Airplane!**



# What we mean by “adapt”

- ***Adapt***: to change behavior using an active feedback process to achieve a goal in the presence of changes in the system or its environment
- **Adaptive methods are introduced into system design to achieve a goal**
  - enhance performance or efficiency
  - maintain desirable behavioral traits
    - example: robust control systems
- **Ultimately, adaptation is realized through software**
  - important to understand impact on system requirements and design through to software implementation and verification



# Can we assure them?

- **From a software perspective:**



- ❖ What are the unique safety concerns and assurance challenges, *especially for software*, for adaptive systems?
  - ❖ Is the guidance in DO-178B/C sufficient to address adaptive systems?
  - ❖ Is something else needed for certain types of adaptive systems?
- **I spent several months trying to get the questions right**



# Machine Learning

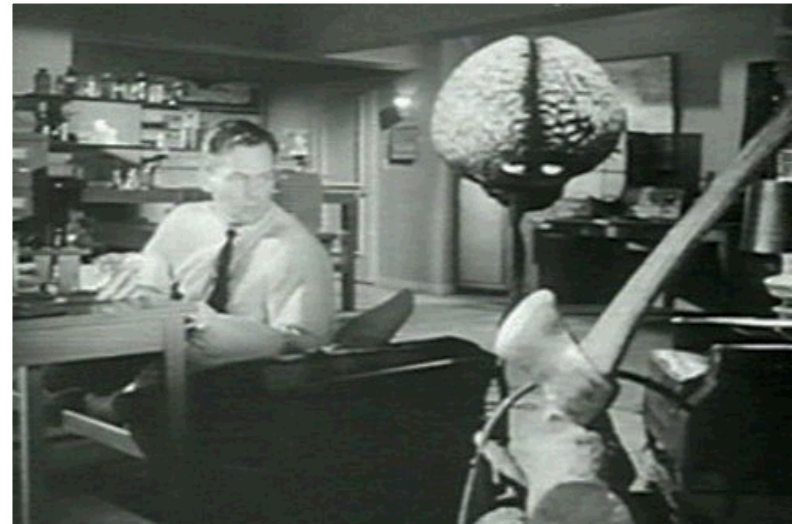
- **Most of the approaches we looked at fall under the rubric of Machine Learning**
- **Machine Learning – typically uses statistical techniques to automatically learn to recognize complex patterns and make intelligent decisions based on data**
  - think of it as applied statistics rather than “AI”
  - the theory and practice of machine learning is advancing at an extraordinary pace





# Learned Behavior

- **Machine learning algorithms do not behave like a traditional mathematical function**
  - generalize based on experience
  - if the system sees input  $x$ , what should its output  $y$  be?
- **Output is not always predictable based on input!**





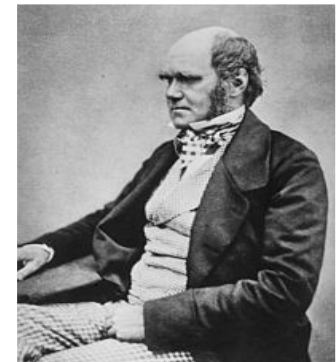
# Dependence on Training Data

- **Common problems with training data**
  - **Undertraining occurs when the resulting model poorly represents the underlying relationship**
    - inaccurate measurements
    - too little data
    - many holes
  - **Overtraining occurs when a statistical model describes random error or noise instead of the underlying relationship**
    - an over-trained model will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data



# Other Approaches for Adaptation

- **Reflection** - Dynamically change software systems based on performance criteria
  - replicate modules or swap in new module with different performance metrics
    - Data Centers
- **Genetic algorithm** - a stochastic search method that mimics the process of natural evolution such as inheritance, mutation, selection, and crossover
  - used for optimization
  - adaptation is done via an iterative process of selecting the best solutions
    - Designing efficient circuits





# Variations

- **Adaptive methods vary in how they “learn” and how they use what they have learned to change behavior**
- **Learning might only be used to help inform the design of a system**
- **Learning might be used to change behavior in real-time based on changing conditions**
- **All of these things influence the ability to accurately predict the changed behavior**



# Adapting Early

- **Apply adaptive mechanisms to produce a better design**
- **Performed solely at the systems level**
- **No adaptation during operations**
  - that is, the output is predictable
- **Still must produce unambiguous systems level requirements of the resulting design**
- **Conventional assurance techniques/process**
- **Example:**
  - fuel measurement system
    - Applied neural nets resulting in a simpler system



# Example 1: Fuel Measurement

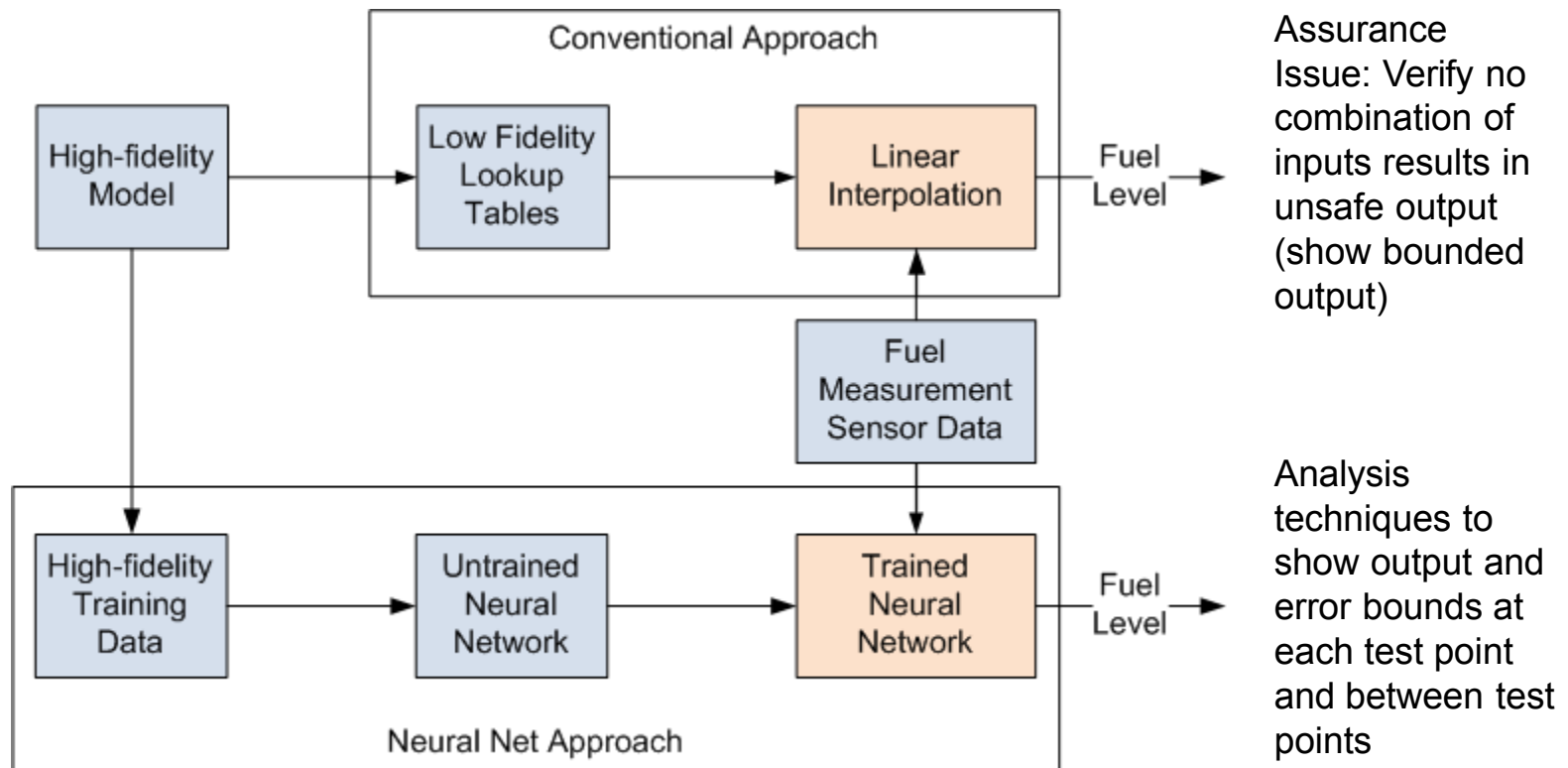
- **Estimate fuel level in distributed tanks**
- **Often uses conservative approximations that result in increased fuel consumption**
- **Complex tank geometries, fuel orientation due to flight attitude → Multidimensional nonlinear problem**
- **Conventional approach**
  - Generate a series of lookup tables from high-fidelity model
  - Simple linear interpolation extracts measurements from table





# Applying Neural Networks to Fuel Measurement

- **Goodrich Corp and Barron Associates project to replace lookup tables with neural networks**



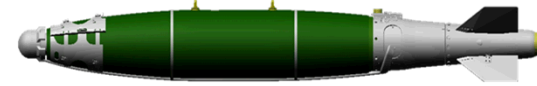


# Adapting in Operation

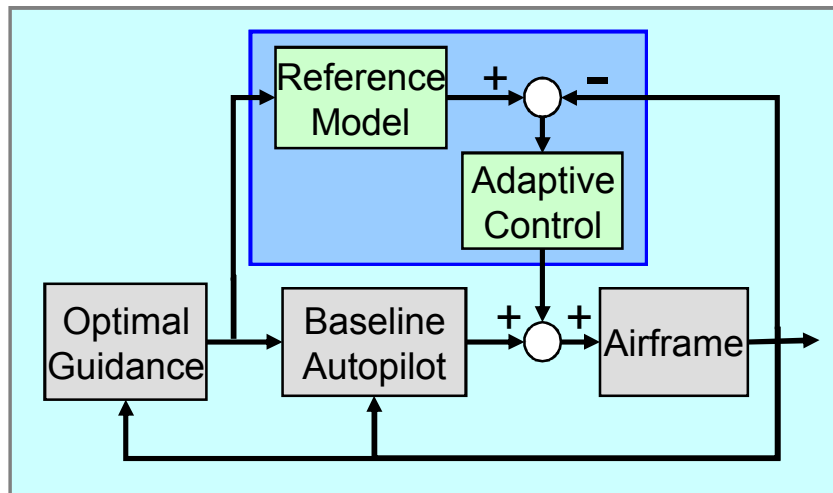
- **Applying adaptive mechanisms that continue to learn in operation**
- **To enhance performance, improve robustness or controllability, .....**
- **Has systems level and software level implications**
- **May have additional requirements associated with learning**
- **Most obvious example – adaptive controls**



# Adaptive Controls Example



- **Use an adaptive mechanism to improve accuracy of existing munitions**
  - retrofitted onto existing autopilot (baseline autopilot unchanged)
  - Joint Direct Attack Munition (Boeing's JDAM)
    - Adaptive augmentation of Optimal Pitch and Roll-Yaw autopilots



Assurance Issue:  
Verify stability of  
composed system  
(Lyapunov, Monte-Carlo  
simulation, etc.)

Nothing akin to FAA s/w  
assurance



# Civil Aviation Processes

- **Vendors build aircraft**
- **The FAA provides oversight**
  - An aircraft must be certified by the FAA in order to fly
- **Vendors must abide by regulations in order to get certified**
  - Often process based regulations
- **In-house or consulting Designated Engineering Representatives (DER) approve or recommend approval of technical data to the FAA**



# DER

- **A FAA designee, a DER may approve engineering technical data within the limits of his/her authority by means of FAA Form 8110-3 and, when authorized by the ACO, may witness FAA compliance tests and perform compliance inspections**
- **DER's will follow the procedures of FAA Order 8110.4, Type Certification Process**



# Systems and Software

- **The design and analysis of the system and software are treated separately with well defined interfaces**
- **Often different groups of people work on the different components**
- **Not unusual for systems and software do be done by different companies and/or different locations**
  - Each group having a unique skill set/knowledge set
    - Systems guys design the controller
    - Software guys know how to get the software to move the actuators



# The Software Team

- **Typically the software team has a real mix of backgrounds**
- **Some have undergraduate degrees in**
  - Aerospace or mechanical engineering
  - Electrical engineers who know microcontrollers, etc.
  - Computer science majors often focusing on testing and operating system level issues
- **If they took controls, it would only be the basic course, they are not control engineers**
- **It is often the case they can read a simulink diagram and implement it**
- **They make the controller work on the actual hardware platform**



# What Gets Done Where

- **Our task focused on the software level as governed by DO 178**
- **DO 178 is about correctness not safety**
- **Talking with controls people was only partially helpful because they wanted to talk about things outside DO 178 scope**
  - The dynamics and control of the aircraft is not addressed at software level
    - Stability is safety and NOT addressed at S/W (178) level
- **If a system level issue is detected, it is sent back to the systems level**



# What do they do at s/w level

- **The software people IMPLEMENT the specification passed from the systems level**
  - Software design implementation documents are produced
  - Traceability to systems level requirements is maintained
  - The SOFTWARE is analyzed and tested
  - Does it meet the requirements
- **The s/w people deal with the specifics of the hardware platform, the operating system, etc.**
- **Verification is intended to show that the software matches the specification**
  - They won't deal with systems properties like stability

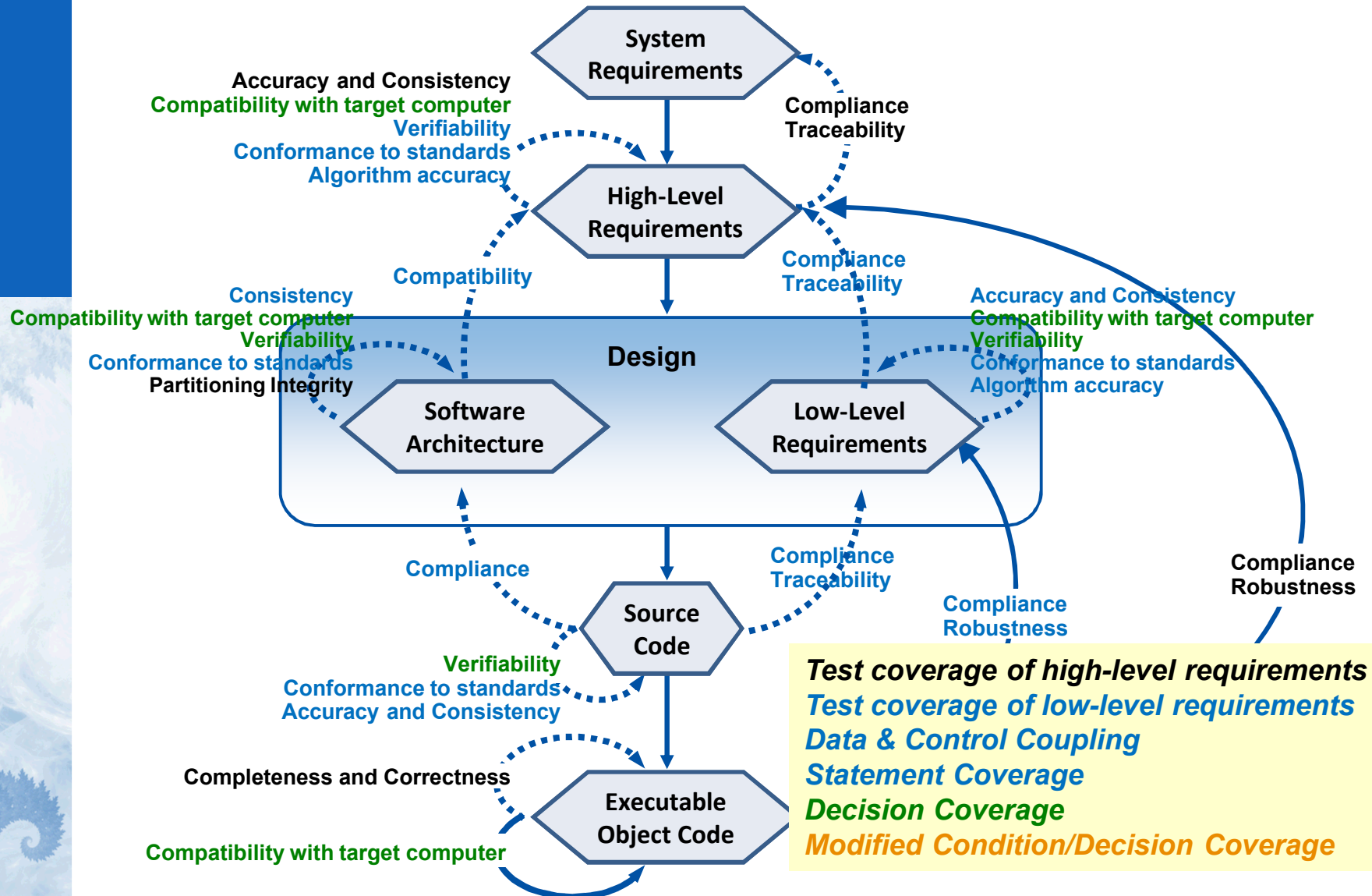


# What the Control Researchers Said

- **Many control researchers were helpful in explaining**
  - The nature of adaptive controls
  - Why it wasn't really so scary
  - A lot about the different techniques to analyze stability
  - A lot about flight testing
- **They could not tell us what the system level requirements for an adaptive controller would look like**
  - How would these differ from system conventional gain scheduling
  - Maybe they are simpler in some aspects
- **Why is this so important?**

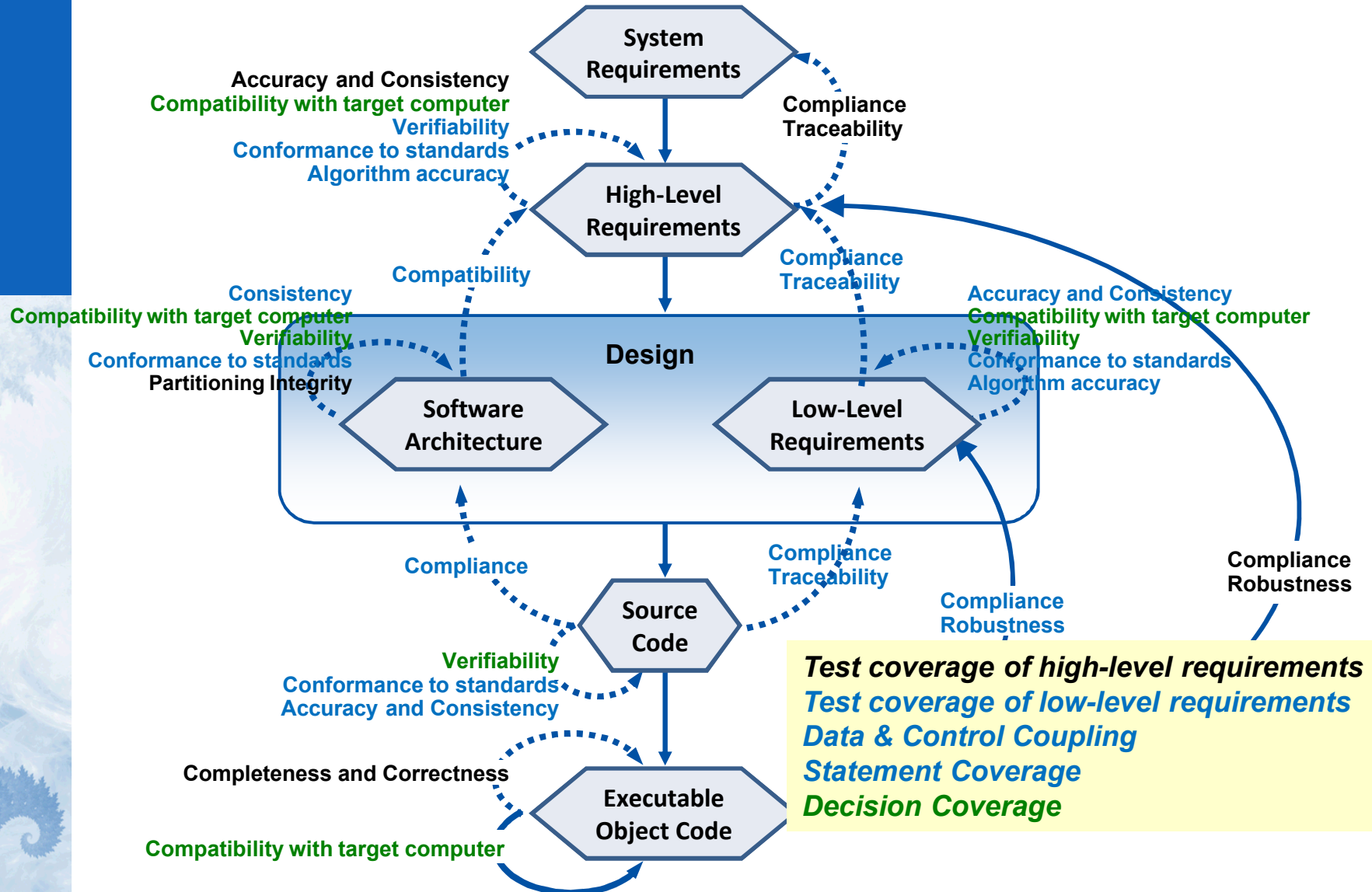


# DO-178B Verification Objectives for Level A



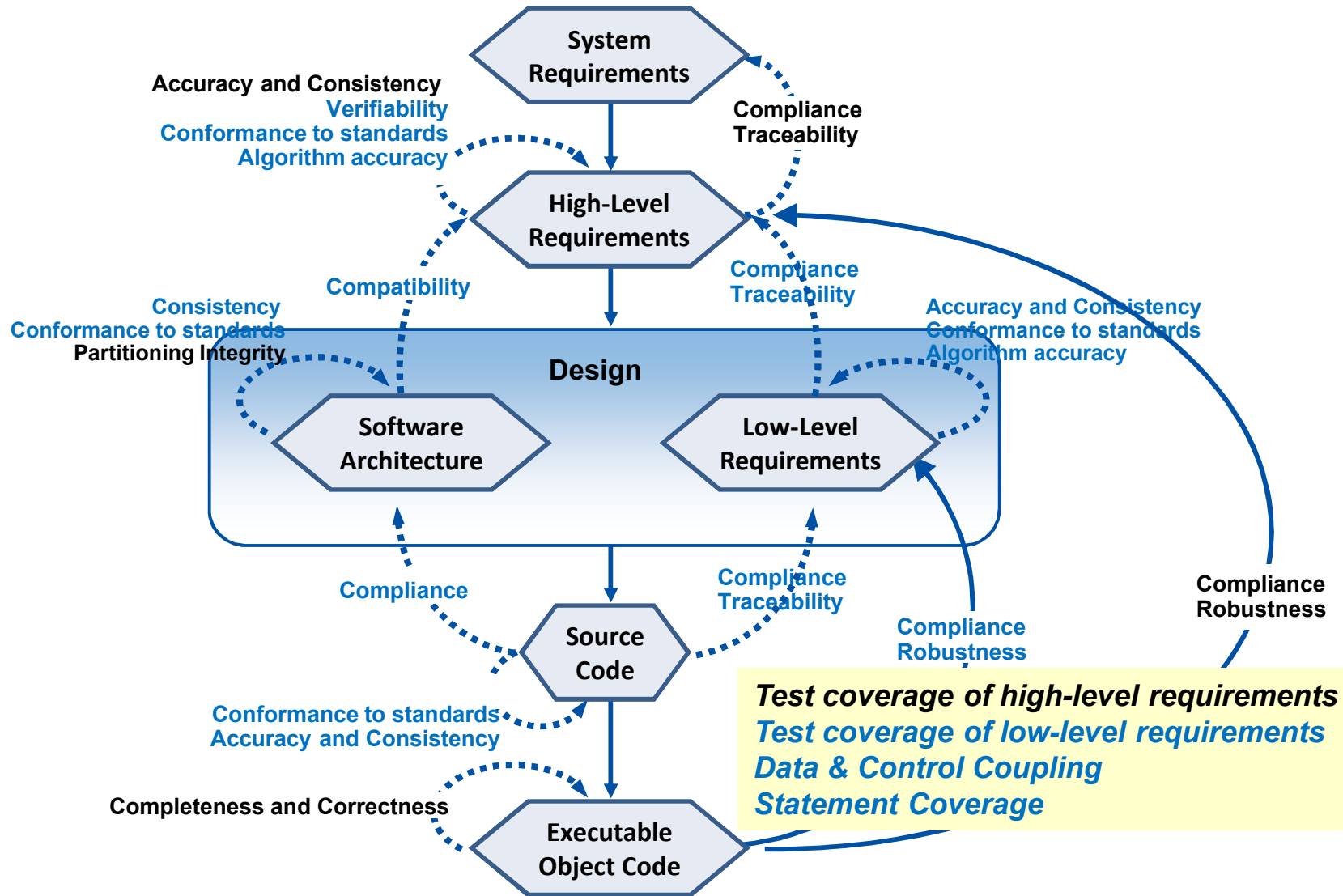


# DO-178B Verification Objectives for Level B



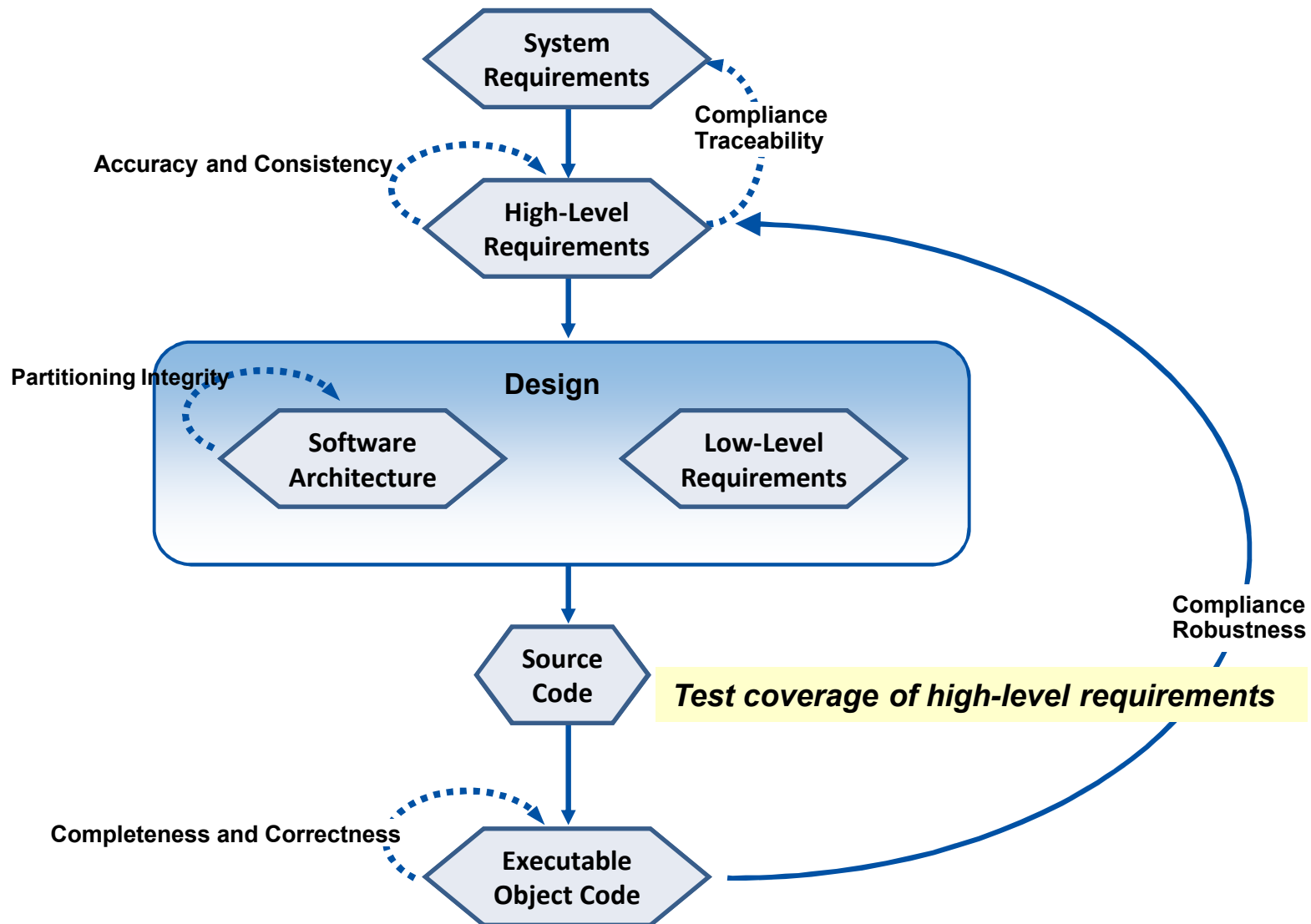


# DO-178B Verification Objectives for Level C





# DO-178B Verification Objectives for Level D





# Systems Level Safety Assurance

- **Sophisticated mathematical techniques are employed to show safety of nonlinear adaptive systems**
  - Lyapunov stability
- **Safety depends on time to converge to a solution**
  - More work needed here
- **Safety in machine learning is dependent on integrity, completeness, and accuracy of data**



# Software Level Safety

- **New algorithms and data structures will be employed in adaptive software**
  - Sparse data structures for machine learning
  - Some methods increase dynamism (dynamically growing neural nets, replicating resources, etc.)
  - Are these sufficiently similar to existing software structures?
- **What kind of tests are needed to verify adaptive software meets requirements?**
- **How do we plan and execute tests when the same inputs can yield different outputs given different histories?**



# Predictability Sacrificed

- **Safety and predictability not the same**
- **Historically regulations seek required levels of safety through predictability**
- **This is where the conflict comes in**
- **Both the software and controls community need to work much harder to bridge this gap**
- **It will have to be done with DERs and FAA officials in the loop**
  - **Probably means industrial participation**



# MBD Supplement

- **A new Model-Based Design (MBD) supplement for 178C**
- **The process is generally open**
  - Vendors send representatives
  - Did not need a background in civil aviation software
- **Several issues have arisen that are of great concern that are being fixed**
  - What is a model?
  - What is the role of simulation?
  - What should autocoders produce?



# What is a Model

- **In the Formal Methods supplement a model is something with a formal syntax and a mathematical semantics**
  - This is the only definition that makes sense to me
- **I have qualms about using any model lacking these two qualities in safety-critical systems**
  - How can you analyze anything without a precise semantics?
- **The MBD supplement still has an imprecise definition of model**



# Simulation I

- **Engineers today use simulation in many cases where they once used experiment or analysis**
- **The simulations can be used in this role because:**
  - The simulation executed a model that encodes knowledge about the physical world that has been validated over time
  - Efficacy of simulation rests heavily on the continuous nature of the mathematics used



# Simulation II

- **Simulation definitely has a role to play in the verification and validation of avionics software**
- **Validating the systems level and software design models**
- **Helping to generate tests for requirements coverage**
- **Simulation of components in a test harness**



# Simulation III

- **Simulation cannot replace testing and analysis of software implementations and DO 178 won't allow it**
- **But that's exactly what early versions of the MBD supplement tried to allow**
- **Simulate the model and you are good to go!**
- **The arguments showed little understanding of the discrete nature of software**



# Autocoding I

- **Control engineers are quite enamored of autocoding**
- **DoD allows autocoded programs on aircraft**
- **FAA only if the autocoder produces all the evidence and traceability required by DO-178C**
- **The tool itself must undergo qualification**
  - Which is very arduous



# Autocoding II

- **A lot of good work recently on**
  - Verified compiler (INRIA Paris/Princeton/PSU/Stevens/.....)
  - Translation verification using SMT solvers (Galois,...)
  - The automated code review documentation (NASA Ames)
- **To really be successful all this work requires a mathematical semantics of the specification language and the implementation language**
- **Vendors have a commercial interest not to be so precise**
  - Their customers (that's you control engineers) don't care
  - They don't want to go thru the qualification



# Learning to Talk To Each Other

- **We need to learn each other's language**
- **CS folks need to take signals, linear systems, and numerical analysis, and to work with hard real-time systems HW and SW**
  - Soldering irons, oscilloscope, and FPGAs
- **Controls folks need knowledge of logic, automata, algorithms, and semantics**
- **When one wants to affect control software used in civil aviation it is best for both software and controls folks to know the regulations governing the process**



# Questions?

