Formal Methods in the Aerospace Industry: *Follow the Money*

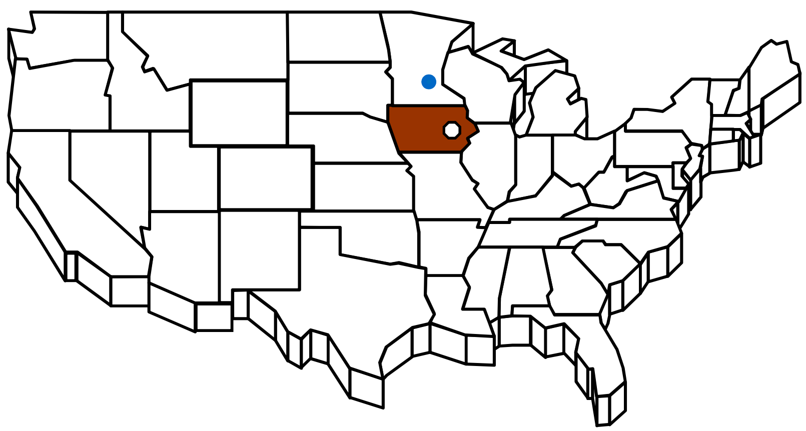
ICFEM
15 November 2012

Dr. Darren Cofer
cofer@ieee.org

Rockwell Collins

Rockwell Collins

Headquartered in Cedar Rapids, Iowa
20,000 Employees Worldwide
2012 Sales of ~$4.8 Billion



Commercial & Military Avionics Systems
Flight Control Systems & Displays
Heads Up Displays
Navigation & Landing Systems
Defense Communications
Weapons Data Links
Cryptographic Equipment

Domestic

California
Carlsbad
Cypress
Irvine
Los Angeles
Pomona
Poway
San Francisco
San Jose
Tustin
Florida
Melbourne
Miami
Orlando
Georgia
Atlanta
Warner Robins
Hawaii
Honolulu
Illinois
Chicago
Iowa
Bellevue
Coralville
Decorah
Manchester
Kansas
Wichita
Maryland
White Marsh
Massachusetts
Boston
Michigan
Ann Arbor
Detroit
Minnesota
Minneapolis
Missouri
Kansas City
St. Louis
New York
New York
North Carolina
Charlotte
Raleigh
Oklahoma
Midwest City
Tulsa
Oregon
Portland
Pennsylvania
Philadelphia
Pittsburgh
Texas
Dallas
Fort Worth
Richardson
Utah
Salt Lake City
Virginia
Sterling
Warrenton
Washington
Kirkland
Renton
Seattle
Washington, DC

International

Africa
Johannesburg, South Africa
Asia
Bangkok, Thailand
Beijing, China
Hong Kong
Hyderabad, India
Kuala Lumpur, Malaysia
Manila, Philippines
Moscow, Russia
Osaka, Japan
Shanghai, China
Singapore
Tokyo, Japan
Australia
Auckland, New Zealand
Brisbane, Australia
Melbourne, Australia
Sydney, Australia
Canada
Montreal
Ottawa
Europe
Amsterdam, Netherlands
Frankfurt, Germany
Heidelberg, Germany
London, England
Lyon, France
Manchester, England
Paris, France
Reading, England
Rome, Italy
Toulouse, France
Mexico
Mexicali
South America
Santiago, Chile
Sao Jose dos Campos, Brazil
Sao Paulo, Brazil

Advanced Technology Center
Trusted Systems group
(Formal Methods researchers)

Outline

- Problem
  - Verification of high-assurance complex systems
- Why use formal methods
  - Cost, safety, certification
- Formal methods for verification
  - Model checking
- Formal methods for certification
  - DO-178C / DO-333
- What's next
  - Compositional reasoning

Domain – avionics

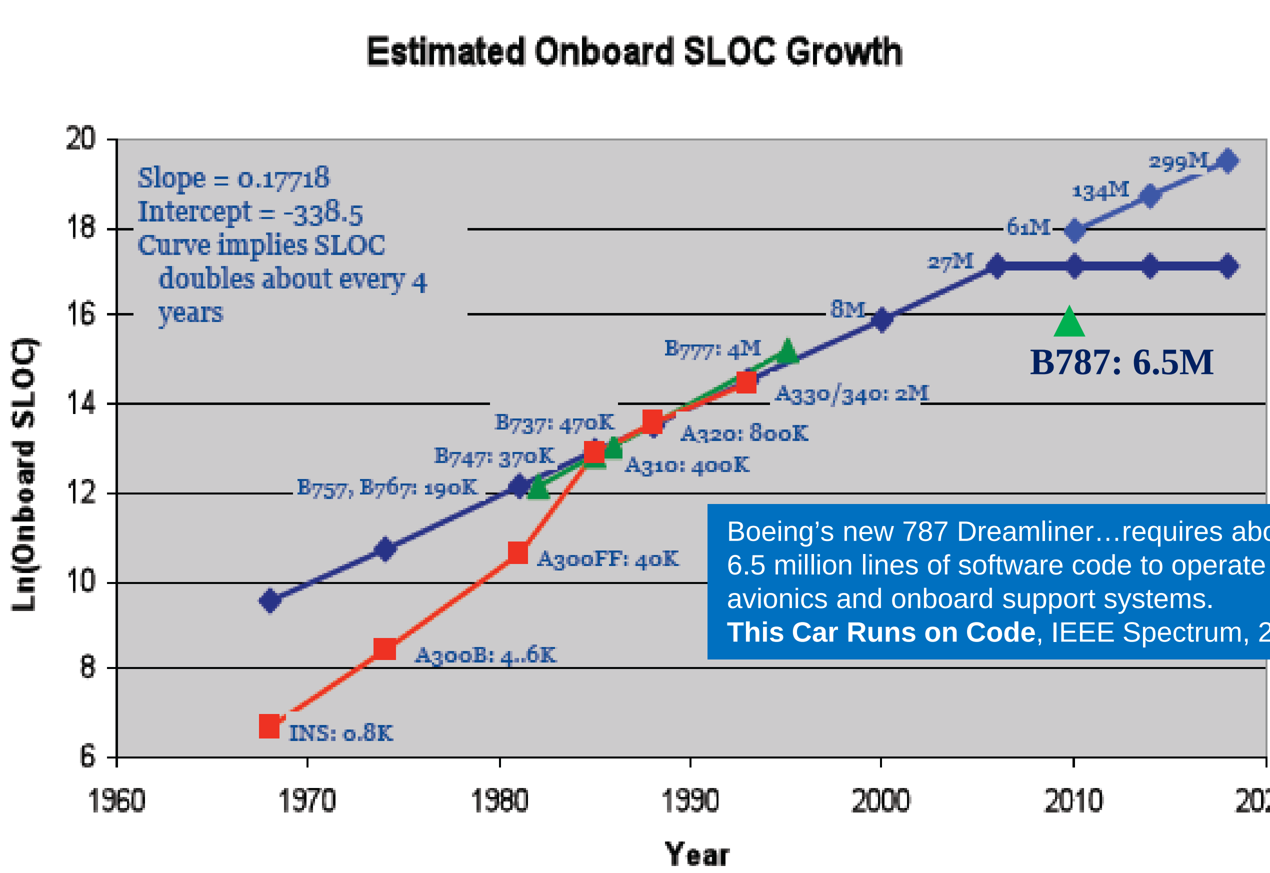
- **Embedded systems** with **safety** and **security** requirements that are critical to operation of vehicle and performance of the mission
- Commercial and military
- Manned and unmanned





Software in commercial aircraft

AVSI AEROSPACE VEHICLE SYSTEMS INSTITUTE



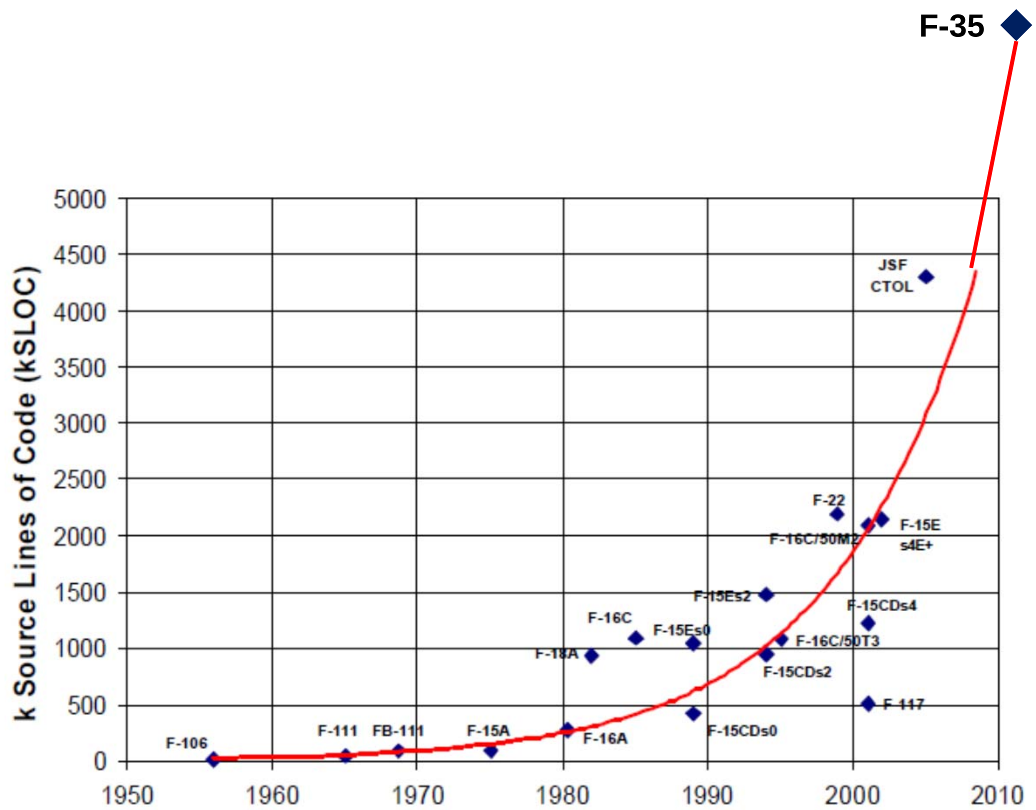
Boeing's new 787 Dreamliner…requires about 6.5 million lines of software code to operate its avionics and onboard support systems.
This Car Runs on Code, IEEE Spectrum, 2/09

The line fit is pegged at 27M SLOC because the projected SLOC sizes for 2010 through 2020 are unaffordable. The COCOMO II estimated costs to develop that much software are in excess of $10B.

Airbus data source: J.P. Potocki De Montalk, Computer Software in Civil Aircraft, Sixth Annual Conference on Computer Assurance (COMPASS '91), Gaithersberg, MD, June 24-27, 1991.
Boeing data source: John J. Chilenski. 2009. Private email.

Software in military aircraft

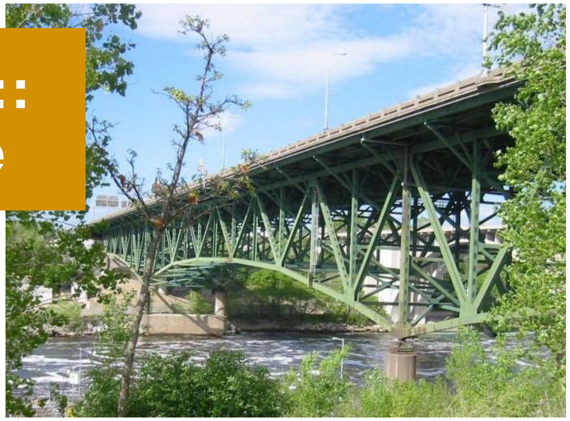
"Software providing essential JSF capability has grown in size and complexity, and is taking longer to complete than expected," the GAO warned. **Pentagon: Trillion-Dollar Jet on Brink of Budgetary Disaster,** Wired 3/21/12



Source: D. Gary Van Oss (USAF), "Avionics Acquisition, Production, and Sustainment: Lessons Learned – The Hard Way," NDIA Systems Engineering Conference, Oct 2002.

The use of **formal methods** is motivated by the expectation that, as in other **engineering** disciplines, performing appropriate mathematical analyses can contribute to establishing the correctness and robustness of a design.

FM : software ::
FEA : structure



Rockwell Collins

Why use formal methods with avionics SW? (A lesson in technology transition)

- Increase confidence?
  - Complete examination of complex software and requirements
  - “Our systems are already safe.”
- Satisfy certification objectives?
  - DO-178C allows certification credit for formal methods
  - Requirements/model verification is done by review (too cheap), and formal source/object code verification is difficult (too expensive)
- Reduce cost?
  - **YES!**
  - Early detection/elimination of defects
  - Automation of verification activities

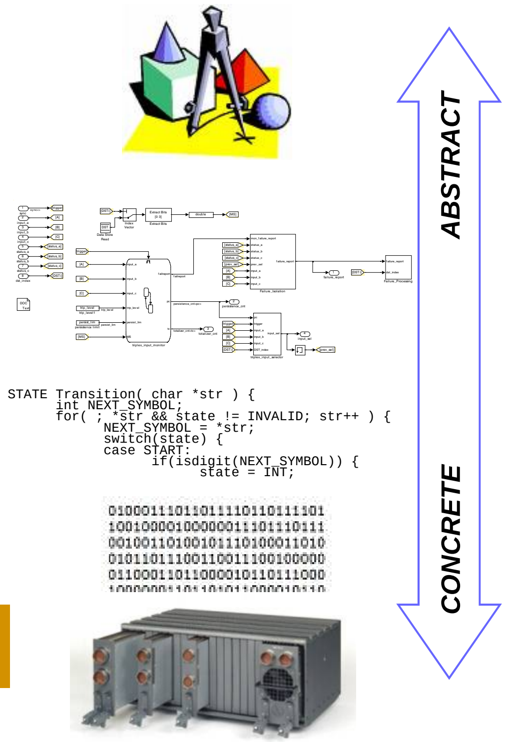
Follow the money.

Formal Methods for Verification

Formal Methods for Certification

Model-based development

Domain-specific (often) graphical design environments for software development
  - Early simulation and debugging
  - Automated code generation
  - DSL promotes higher level of abstraction in design



MBD enhances the FM value proposition

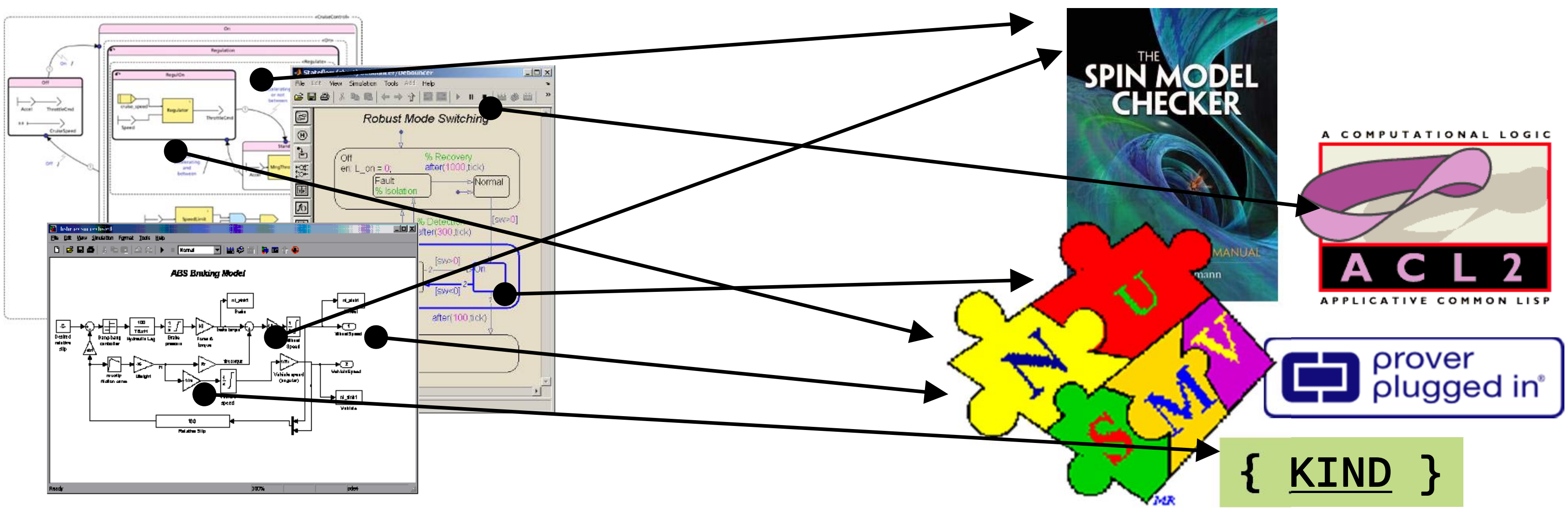
- Take advantage of
  - Industry adoption of Model-Based Development tools
  - Increasing power of formal methods analysis engines
  - Moore’s Law
- Use formal methods to fight cost and complexity with automation and rigor

Barriers to use of FM

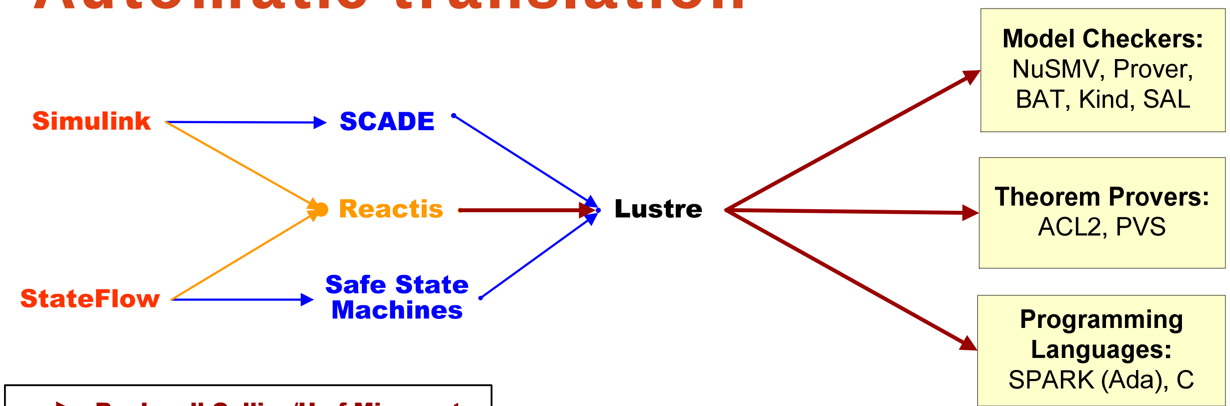
- “If formal methods are so great, why aren’t they more widely used?”
- The main barriers in the past have been:
  1. **Cost**: building/maintaining separate analysis models
  2. **Fidelity**: models don’t match real system
  3. **Usability**: unfamiliar notations/tools
  4. **Scale**: inadequacy of tools for industrial-sized problems
- ***MBD is eliminating the first three barriers***
  - Leverages existing modeling effort
  - Automated translations and analysis
  - Familiar notations for engineers (Simulink + Stateflow)
- ***Fourth barrier is also falling…***
  - Moore’s Law = more power available on desktop
  - Exploit rapid advances in model checking (e.g., SMT)

Problem: bridging the gap

- MBD captures design at sufficient detail and sufficient formality
- Powerful formal methods tools can analyze large models
- However…
  - there are still a variety of models used in MBD environments
  - and many good analysis tools with different strengths and weaknesses



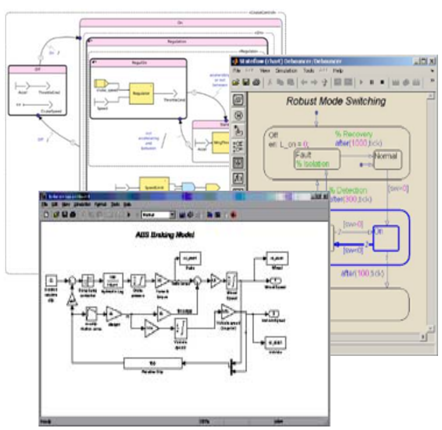
Automatic translation

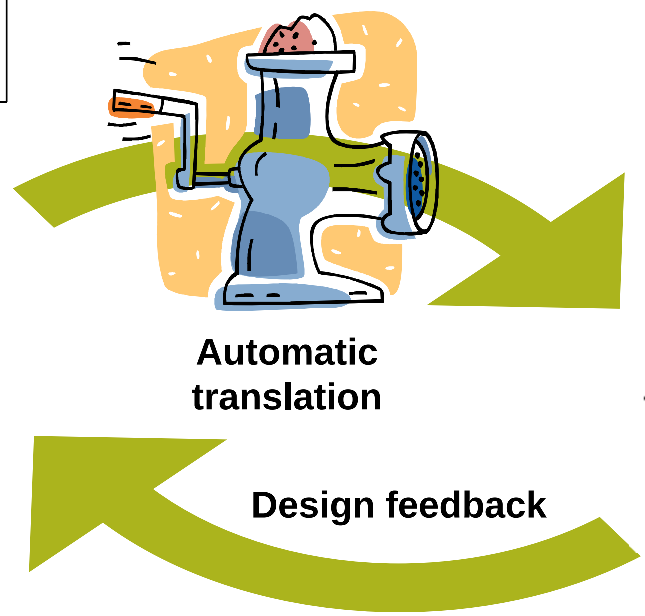


- Rockwell Collins/U of Minnesota
- Esterel Technologies
- MathWorks
- Reactive Systems

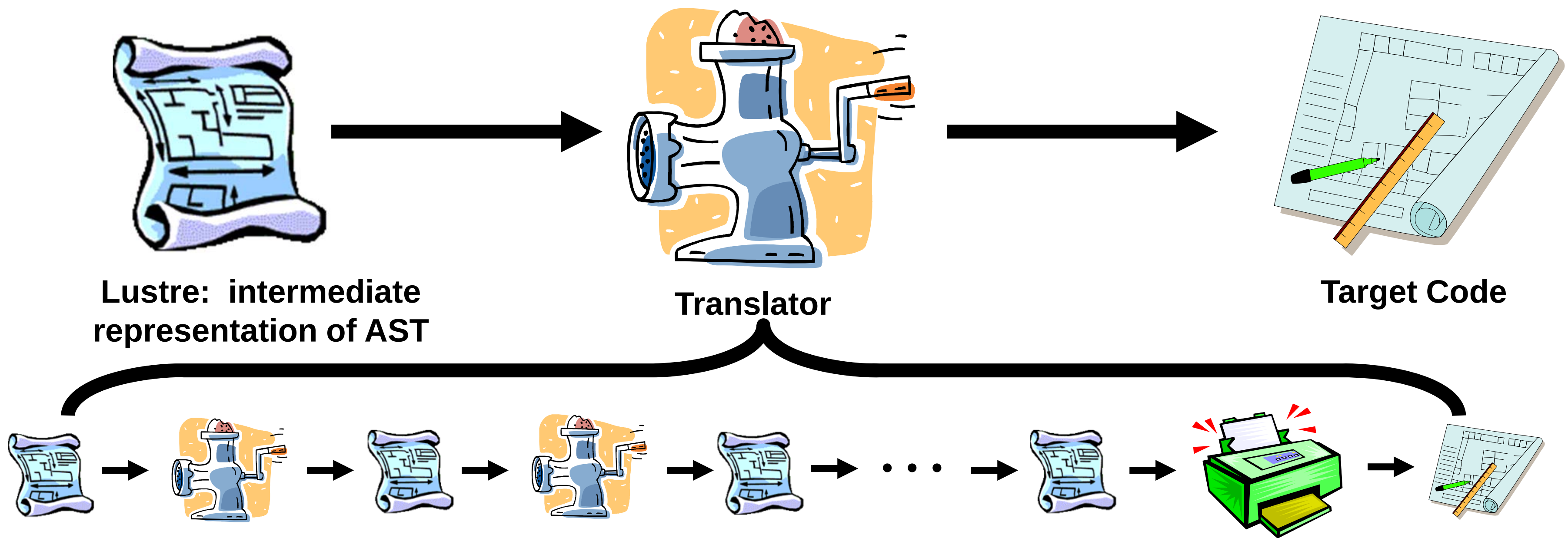
Gryphon translation framework

- Supports a wide variety of back end tools and languages
- Straightforward to add new tools (e.g. Prover support added in 4 days)
- Apply "the right tool for the job"



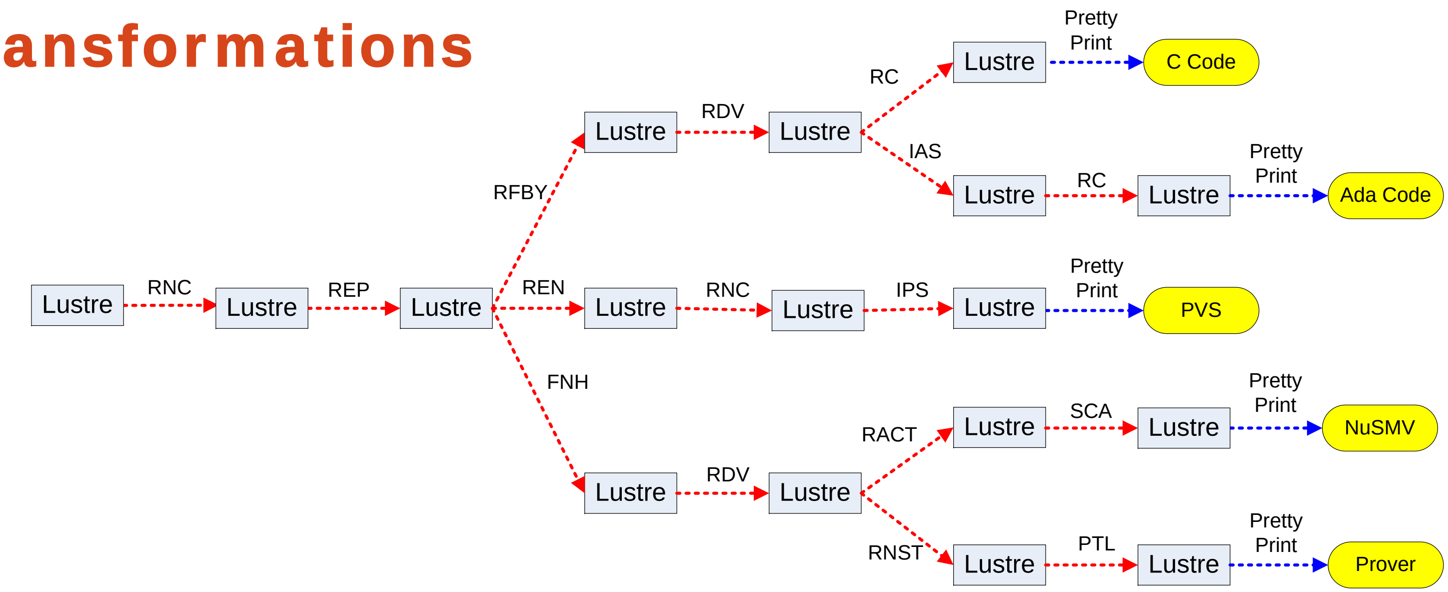


Translator Framework



- Mechanism: Small source-to-source transformations in Lustre
  - Deal with one language aspect at a time
  - Define pre/post-conditions that describe when transformation can be performed and its effect
  - Refine Lustre specification until it resembles target language
  - Create language-specific emitter to output target code

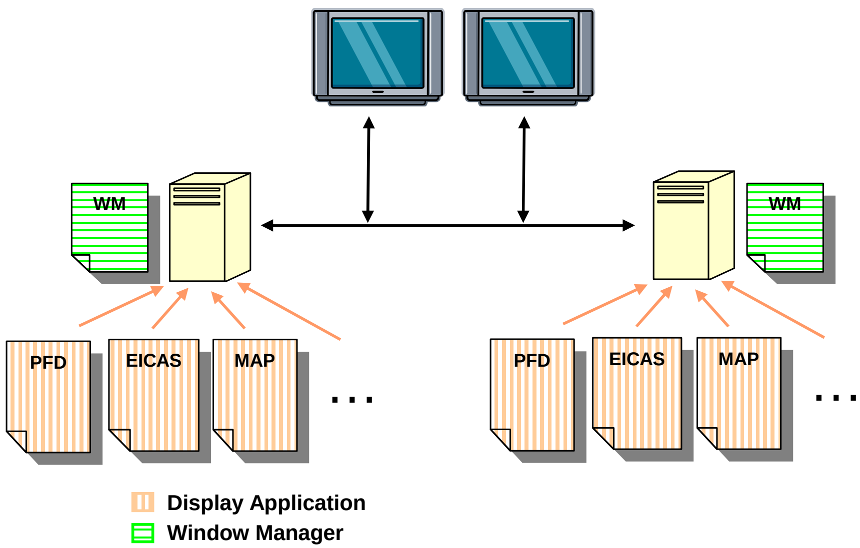
Transformations

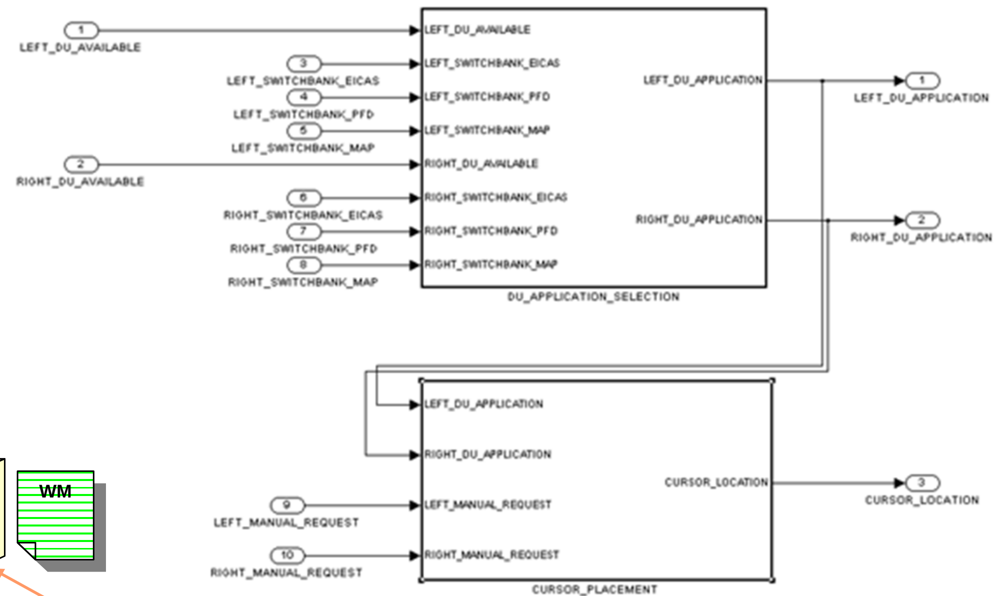


- Different target languages use different combinations of transformations
  - May be 50+ transformations for a given target
- Transformations **optimize** final output for target language
  - Strengths of selected analysis engine
  - Speed/size/readability of source code
  - Reduce analysis times from hours to seconds

Application: Eliminate errors

**ADGS-2100
Window Manager SW
(cockpit display)**





ADGS-2100 model checking results

Requirement	CTL Properties
R1: If a DU is available, then it shall display some application	`AG( LEFT_DU_AVAILABLE -> LEFT_DU_APPLICATION != BLANK )` `AG( RIGHT_DU_AVAILABLE -> RIGHT_DU_APPLICATION != BLANK )`
R2: If a DU is unavailable, then it shall not attempt to display any application	`AG( !LEFT_DU_AVAILABLE -> LEFT_DU_APPLICATION = BLANK )` `AG( !RIGHT_DU_AVAILABLE -> RIGHT_DU_APPLICATION = BLANK )`
R3: The cursor will not be displayed on a DU that is unavailable	`AG( !LEFT_DU_AVAILABLE -> CURSOR_LOCATION != LEFT_DU )` `AG( !RIGHT_DU_AVAILABLE -> CURSOR_LOCATION != RIGHT_DU )`
R4: The cursor shall not be displayed on a DU whose application is not MAP	`AG( LEFT_DU_APPLICATION != MAP -> CURSOR_LOCATION != LEFT_DU))` `AG( RIGHT_DU_APPLICATION != MAP -> CURSOR_LOCATION != RIGHT_DU)`

Subsystem	Simulink Diagrams	Simulink Blocks	State Space	Properties	Errors found
GG	2,831	10,669	9.8×10^{9}	43	56
PS	144	398	4.6×10^{23}	152	10
CM	139	1,009	1.2×10^{17}	169	10
DUF	879	2941	1.5×10^{37}	115	8
MFD	302	1,100	6.8×10^{31}	84	14
Totals	4295	16,117	n/a	563	98

Application: Eliminate errors and save money

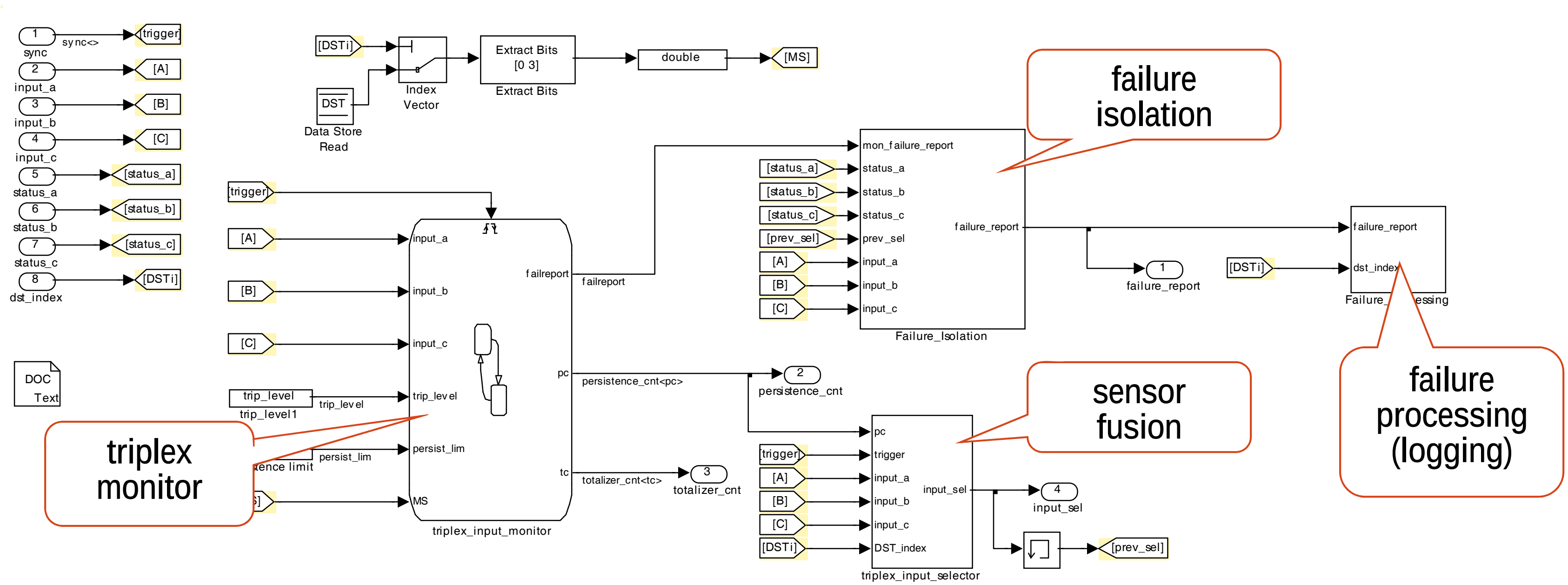
- AFRL CerTA FCS program
  - Team: Lockheed Martin + Rockwell Collins
- Problem
  - The cost of software V&V for UAVs has been identified as the primary obstacle to their future development
  - These costs are expected to grow rapidly as sophisticated adaptive control systems are introduced
- Measure cost and quality improvements using model checking for verification of UAV software
  - Use RC model-checking tools to verify LM Aero advanced flight control models
  - **Quantify the cost and quality** achieved by formal verification vs. test-based verification

It's a contest!

Redundancy Manager software for UAV

- Sensor fusion, failure detection, and reset management for sets of triply redundant sensors
- Mostly discrete logic: ideal problem for model checking

Subsystem	Subsystems / Blocks	Charts / Transitions / TT Cells	Reachable State Space	Properties	Confirmed Errors
Triplex voter no FHT	10 / 96	3 / 35 / 198	$6.0 * 10^{13}$	48	5
failure processing	7 / 42	0 / 0 / 0	$2.1 * 10^{4}$	6	3
reset manager	6 / 31	2 / 26 / 0	$1.32 * 10^{11}$	8	4
Totals	23 / 169	5 / 61 / 198	N/A	62	12



Redundancy Manager counterexample

Requirement: A sensor that does not miscompare shall not be declared failed in the next frame.

Property:

```
SPEC AG((!b_miscompare & !dst_b_failed) ->
        AX (failure_report != b_failed));
```

Counterexample:

INPUTS						
input_a	0	3	3	3	3	**3**
input_b	0	1	1	1	1	**5**
input_c	0	3	3	3	3	**5**
status_a	0	0	0	0	0	0
status_b	0	0	0	0	0	0
status_c	0	0	0	0	0	0
OUTPUTS						
failure_report	0	0	0	0	0	**4**
persistence_cnt	0	1	2	3	4	0
totalizer_cnt	0	1	2	3	4	4

Sensor **b** miscompares for 4 steps.

Sensor **a** miscompares on next step.

Sensor **a** is declared failed instead of sensor b! (failure report "4")

Problem: Only one miscompare persistence counter

Testing vs. Model Checking

- LM and RC teams start with same set of requirements and software models
- Both teams spent comparable effort to add enhancements to their verification framework (support for new blocks, graphical test case viewer, XML test case generation)
- Measure effort to perform verification and diagnose results
- [FMICS 2007]

Redundancy Manager
(verification effort)

	Effort	Errors found
LM: Test	X	0
RC: MC	0.7X	12

RC effort includes fixing the errors found!

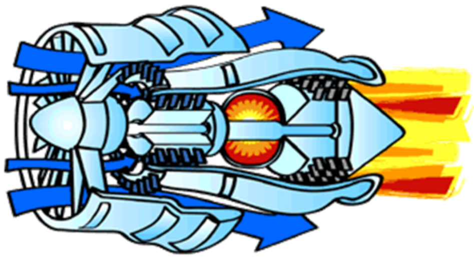
Certification

- Legal recognition by the regulatory authority that a product, service, organization or person complies with the requirements
  - Type certification: design complies with standards to demonstrate adequate safety, security, etc.
  - Product conforms to certified type design
  - Certificate issued to document conformance
- Examples of certification evidence
  - We used verification tool X to accomplish these objectives.
  - These are the reasons why we think the tool is acceptable.
  - We ran 1000 tests using the tool, and this is why we think these 1000 tests are sufficient.
  - And (almost incidentally) here are the test results.

Convincing the relevant Certification Authority that all required steps have been taken to ensure the safety/reliability/integrity of the system

Certification and civil aviation

- Software is not actually certified



- But certification of an aircraft *does* include “software considerations”

DO-178B:
"Software Considerations in Airborne Systems and Equipment Certification"

- Published in 1992
- Developed jointly by industry and governments from North America and Europe
  - Published by RTCA in U.S.
  - Published by EUROCAE in Europe as ED-12B
- Certification authorities agree that an applicant can use guidance contained in DO-178B as a means of compliance (but not the only means) with federal regulations governing aircraft certification

- What about military aircraft?

Overview of DO-178B

- Primarily a **quality** document, not safety
  - Demonstrate that software implements requirements
  - and nothing else (no surprises)
- Requires auditable evidence of specific processes
  - Software Planning
  - Software Development
  - Software Verification
  - Software Configuration Management
  - Software Quality Assurance
  - Certification Liaison

Overview of DO-178B

- Five Software Levels (DAL in other contexts)
  - A: Catastrophic (everyone dies)
  - B: Hazardous/Severe (serious injuries)
  - C: Major (significant reduction in safety margins)
  - D: Minor (annoyance to crew)
  - E: No Effect (OK to use Windows)
- Objective based
  - Specifies what is to be achieved, not how
- Different objectives and requirements for each SW level
  - Higher level $\Rightarrow$ more objectives to be satisfied

Verification in DO-178B

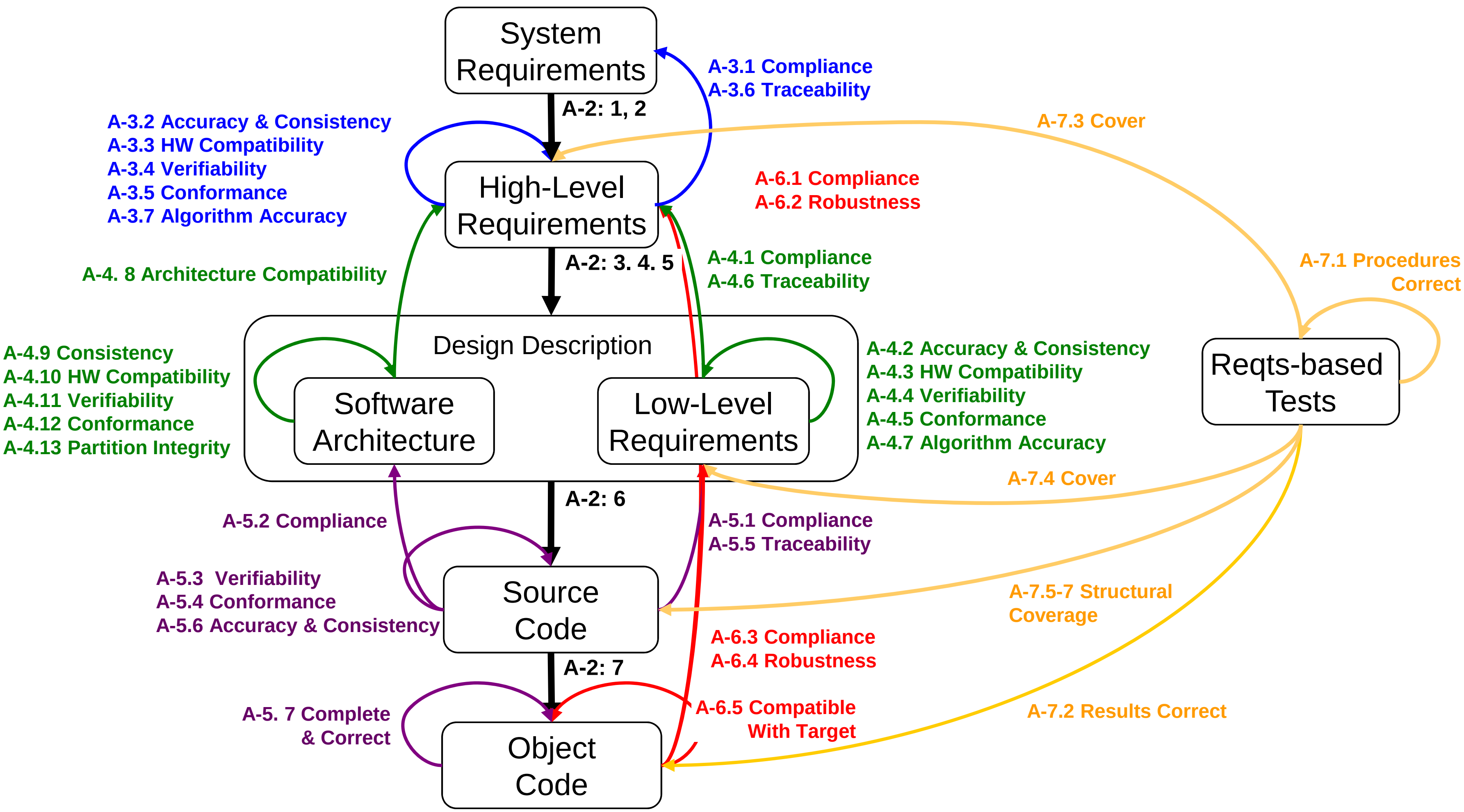
- Verification = review + analysis + test
- Requirements-based testing
- Traceability among
  - Requirements
  - Test cases
  - Code
- How do we know if we have done enough testing?
  - Coverage metrics to determine adequacy of testing/requirements
- Two complementary objectives
  - Demonstrate that the software satisfies its requirements.
  - Demonstrate with a high degree of confidence that errors which could lead to unacceptable failure conditions, as determined by the system safety assessment process, have been removed.

Coverage metrics

- Defines structural coverage metrics
  - Statement coverage (A, B, C)
    - Every statement in the program has been invoked at least once
  - Decision coverage (A, B)
    - and every point of entry and exit in the program has been invoked at least once, and every decision (branch) in the program has taken on all possible outcomes at least once
  - Modified condition / decision coverage (A)
    - and every condition in a decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision's outcome.
- Coverage shortcomings could indicate
  - Missing requirements
  - Inadequacy of test cases
  - Dead or deactivated code

Problem: discrete nature of software
Goal: provide **complete** evaluation of software behavior

DO-178B Verification Objectives (Level A)



That was 1992…

- Any changes in software technology since then?

- New SW development technologies
  - Object-oriented programming languages
  - Model-based development (MBD)
- New verification technologies
  - Formal methods (FM)
- More software!!

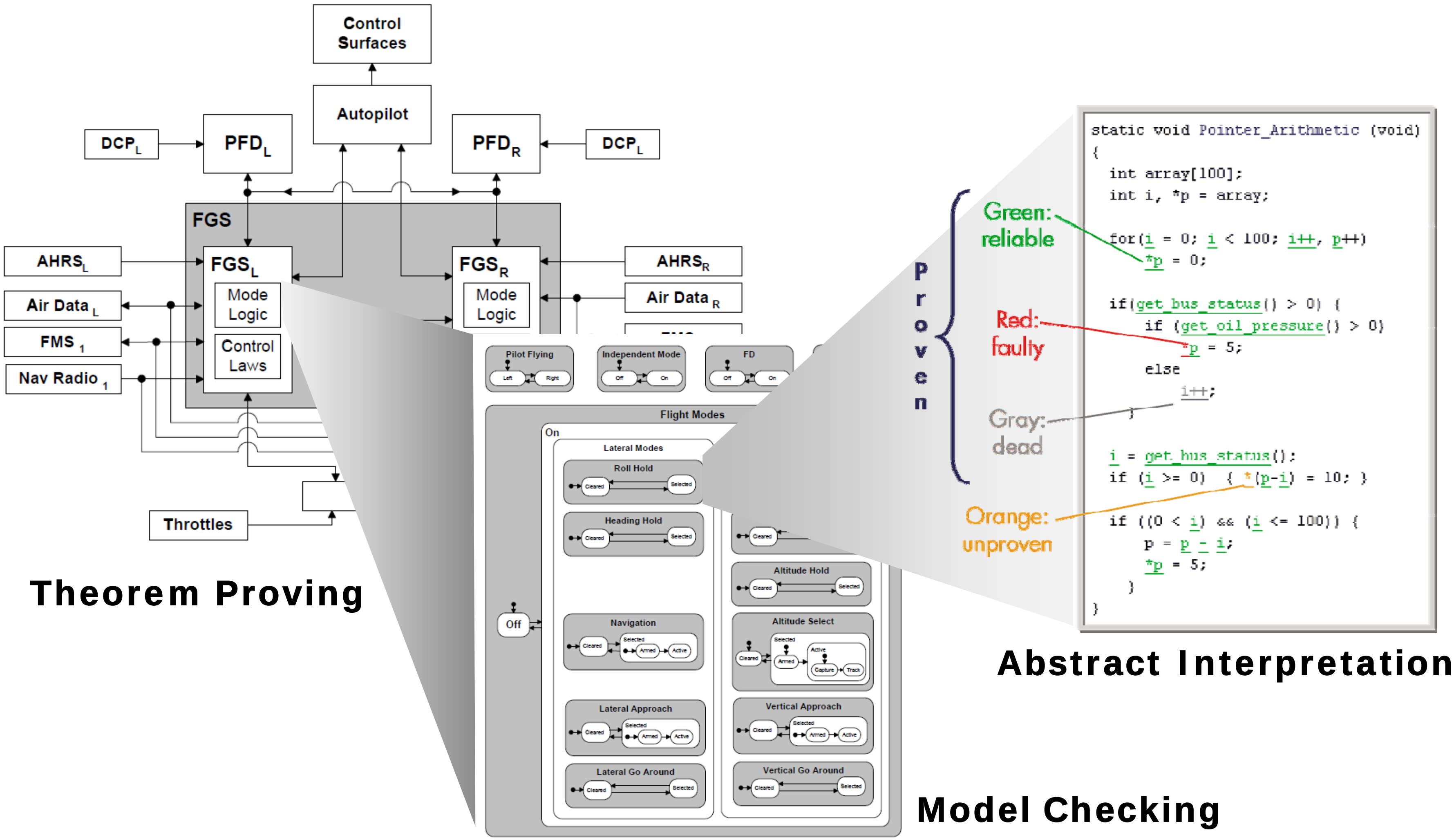
DO-178C

- In late 2004, RTCA & EUROCAE agree to create joint committee to update DO-178B and develop DO-178C
  - Start: 2005
  - Finish: ~~2008~~ ~~2010~~ 2011
- Terms of Reference governing update
  - Minimize changes to core document, yet…
  - Update to accommodate 15+ years of SW experience
- Strategy: Address new technologies in “supplements”
  - OO, MBD, FM
  - Also tool qualification
- Other issues
  - Air/ground synergy (DO-278)
  - Rationale, consolidation, issues, errata (DO-248)

DO-333: Formal Methods Supplement

- Objectives
  - No longer an "alternate method" (as in DO-178B)
  - Provide basis for communication between applicants & certification authorities
  - Focus on verification (DO-178 section 6)
  - Partial use is OK
  - What should formal methods evidence look like?
  - Define new objectives/activities/documentation (abstractions, assumptions)
  - Avoid common errors (check false hypotheses)
- Key issues
  - Capturing assumptions used in analysis (constraints, assertions, environment…)
  - If analysis replaces unit testing, what constitutes "completeness" of analysis? (analog of MC/DC coverage metric)
  - How should formal analysis tools be <u>qualified</u>?
- Keep the bar high enough
  - Applicants with sufficient expertise

NASA DO-333 Case Study project

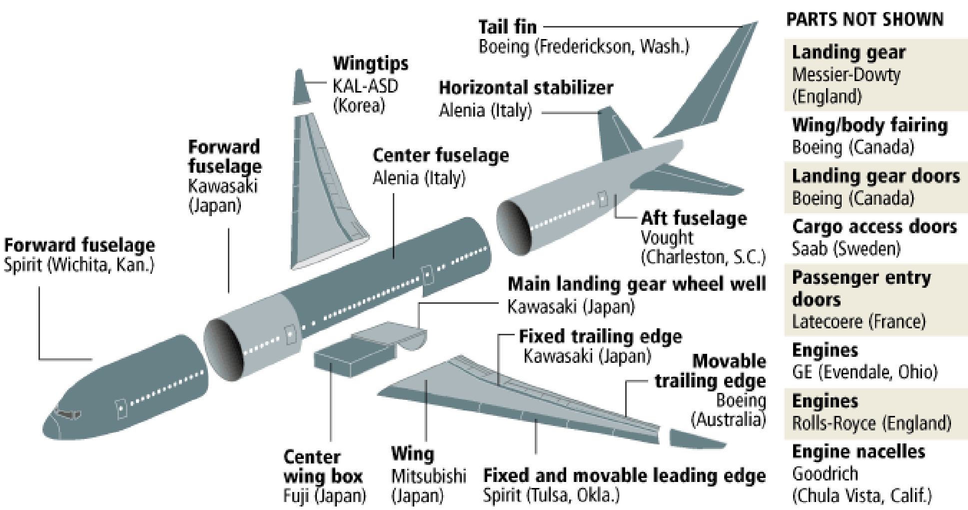


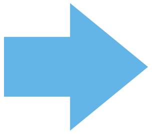
What's next?

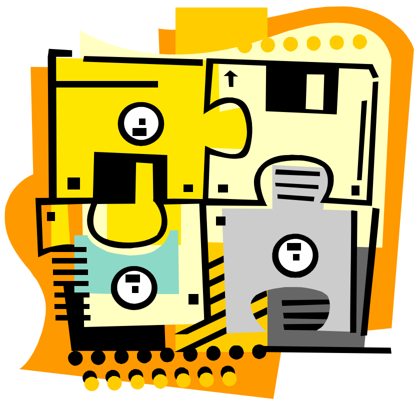
Compositional Reasoning

Vision: "Integrate, then Build"

- Build on success of formal verification of software components
- Extend to system level via software architecture models
- Goals: Early detection/elimination of bugs
  - Cheaper to fix in design vs. integration
  - High-assurance
- Hardware analogy…



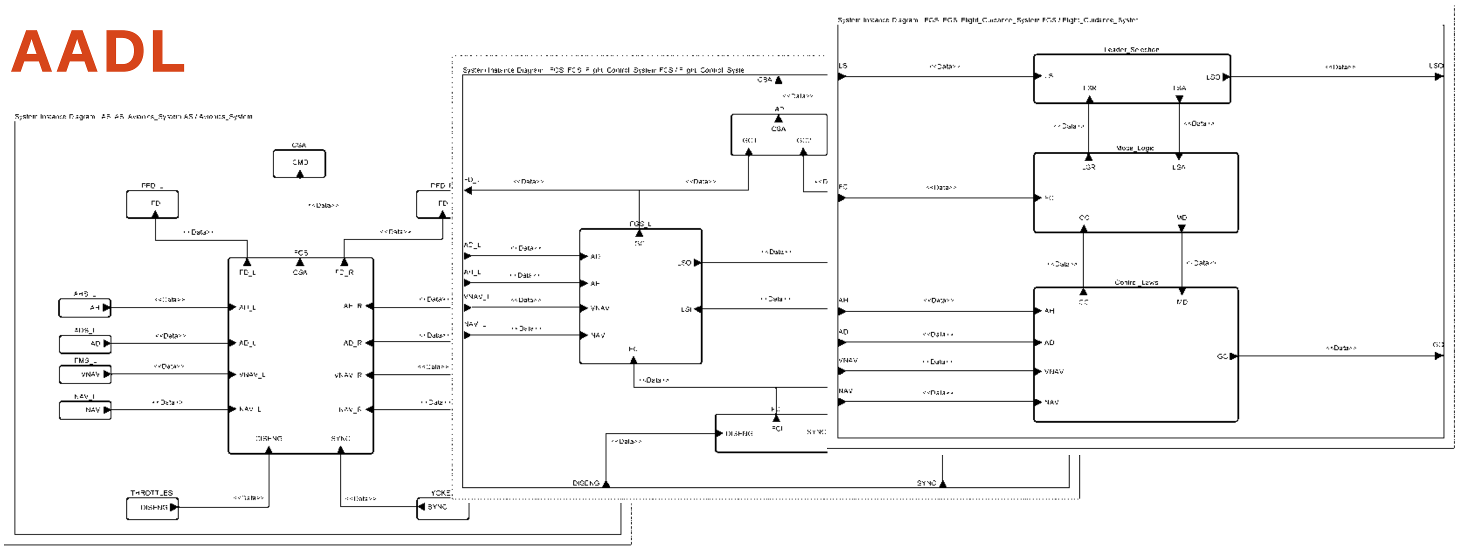




Scale and Composition

- Architectural model should not capture implementation details
  - Component descriptions, interfaces, interconnections
  - Link to implementations
- **Assume-guarantee contracts** provide the information needed from other modeling domains to reason about system-level properties
  - Guarantees correspond to the component requirements
  - Assumptions correspond to the environmental constraints that were used in proving the component requirements
  - Contract specifies precisely the information that is needed to reason about the component's interaction with other parts of the system
  - Supports hierarchical decomposition of verification process
- Add contracts to AADL model

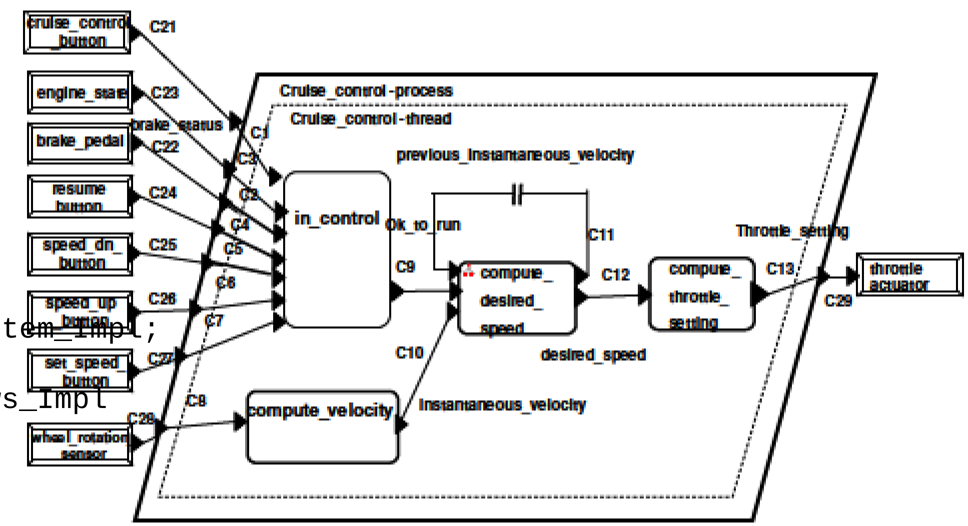
AADL



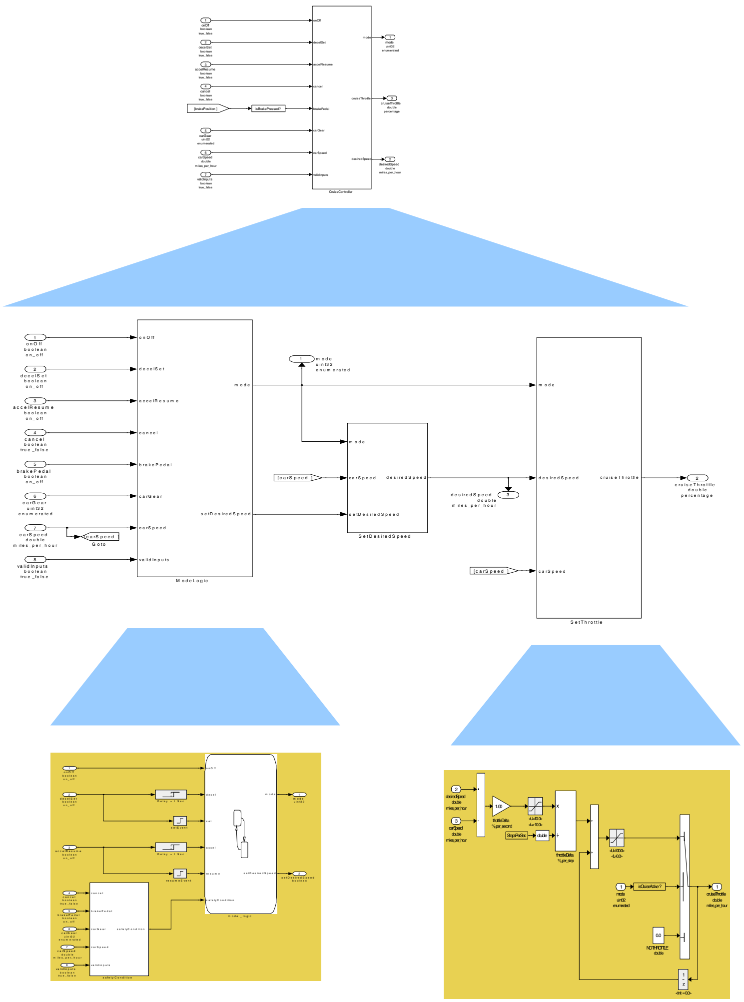
```
system implementation Flight_Guidance_System.Flight_Guidance_System_Impl
  subcomponents
    FGP: process Flight_Guidance_Process.Flight_Guidance_Process_Impl;

  connections
    VNAVtoFGP: port VNAV -> FGP.VNAV;
    ADtoFGP: port AD -> FGP.AD;
    AHtoFGP: port AH -> FGP.AH;
    NAVtoFGP: port NAV -> FGP.NAV;
    FCItoFGP: port FCI -> FGP.FCI;
    LSItoFGP: port LSI -> FGP.LSI;
    FGPtoLSO: port FGP.LSO -> LSO;
    FGPtoGC: port FGP.GC -> GC;
 end Flight_Guidance_System.Flight_Guidance_System_Impl;

 thread implementation Control_Laws.Control_Laws_Impl
 end Control_Laws.Control_Laws_Impl;
```



Compositional reasoning follows architecture



Typical Model-Based Design

- Models are organized in a hierarchy several (many) levels deep
- Much of the complexity is in the leaf models
- Leaf models can often be verified through model checking

Composition of Subsystems

- Combines heterogeneous evidence
- Assume/guarantee reasoning
- Well suited for theorem proving

Compositional reasoning

- Given
  - Assumptions for system
  - Assumptions/Guarantees for components (A, P)
- Prove
  - System guarantees (requirements)
- Assume-Guarantee Reasoning Environment (AGREE)
  - Automatic translation of model structure, contracts, and verification conditions
  - Verify via k-induction model checker (KIND/U. Iowa, Yices/SRI)

Contract compliance:
$G(H(A) \Rightarrow P)$

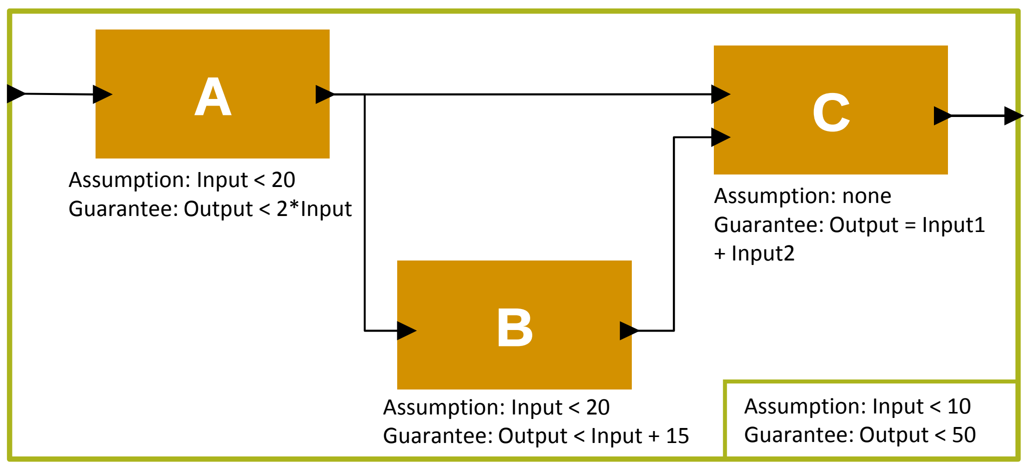
Example (to prove)

$A_S \rightarrow A_A$

$A_S \wedge P_A \rightarrow A_B$

$A_S \wedge P_A \wedge P_B \rightarrow A_C$

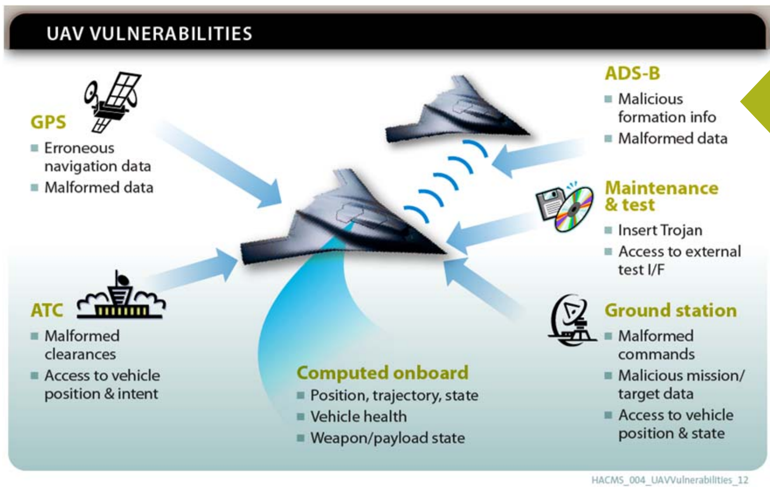
$A_S \wedge P_A \wedge P_B \wedge P_C \rightarrow P_S$



HACMS motivation…

Rockwell Collins

High Assurance Cyber-Military Systems



Network-enabled UAVs are vulnerable to cyber-attack

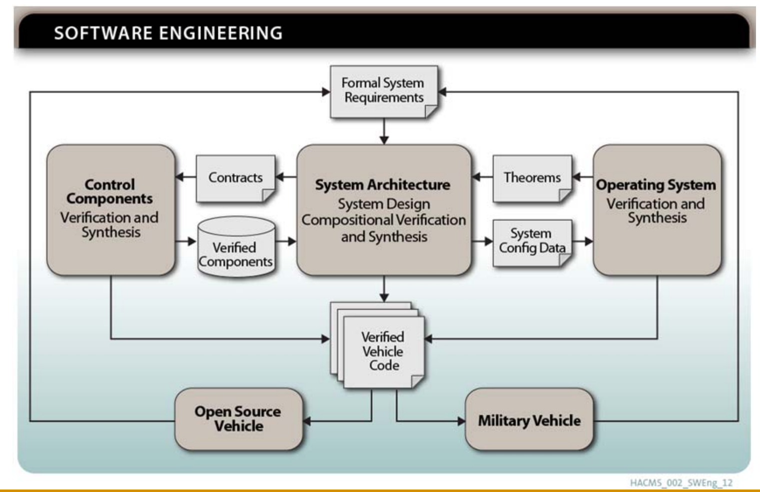
SMACCM

SECURE MATHEMATICALLY-ASSURED COMPOSITION OF CONTROL MODELS

SMACCM: $18M/4.5 year project funded by DARPA Information Innovation Office

Objective: Produce a clean-slate, formal methods–based approach to the development of network-enabled military vehicles to build systems that provide the highest levels of dependability and are resistant to emerging cyber threats

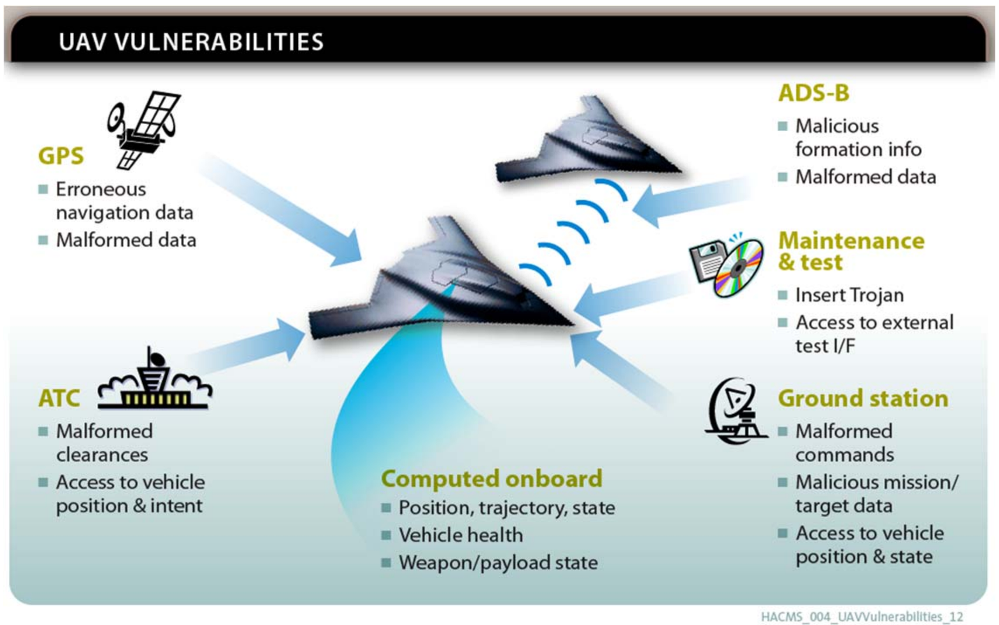
- Develop a complete, formally proven architecture for UAVs (and other embedded systems) that provides robustness against cyber attack
- Develop compositional verification tools for combining formal evidence from multiple sources, components, and subsystems
- Prototype these technologies on an open research platform and transfer them to a military platform to demonstrate their practicality and effectiveness
- Team includes Boeing, NICTA, Galois, Univ. of MN



ROCKWELL COLLINS HAS ASSEMBLED A TEAM OF THE WORLD'S LEADING AUTHORITIES ON SOFTWARE VERIFICATION AND HIGH ASSURANCE DEVELOPMENT TO CREATE A REVOLUTIONARY NEW WAY OF BUILDING ROBUST AND SECURE MILITARY VEHICLES

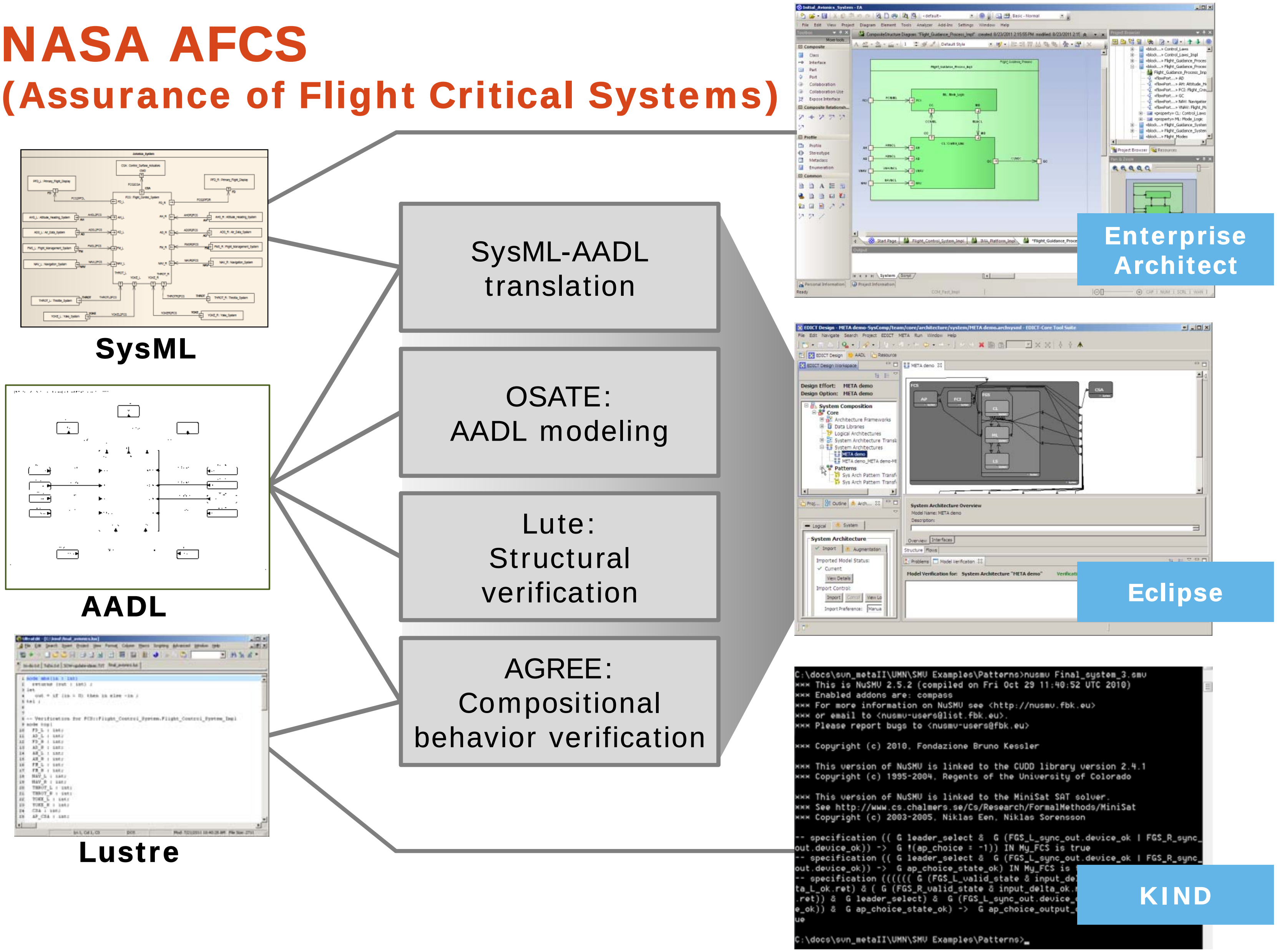
BOEING NICTA galois UNIVERSITY OF MINNESOTA

Target: Unmanned Air Vehicles



Security vulnerabilities that can lead to safety hazards

NASA AFCS
(Assurance of Flight Critical Systems)



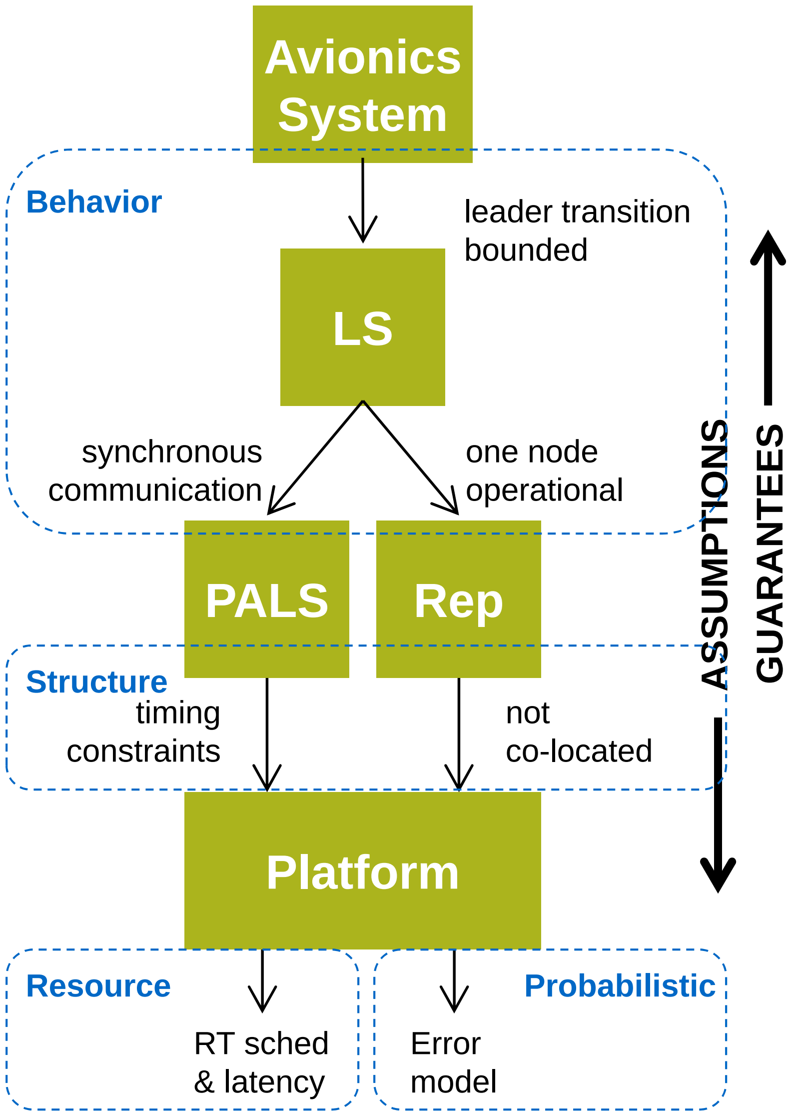
Composition of heterogeneous evidence

- Avionics system requirement

Under single-fault assumption, GC output transient response is bounded in time and magnitude

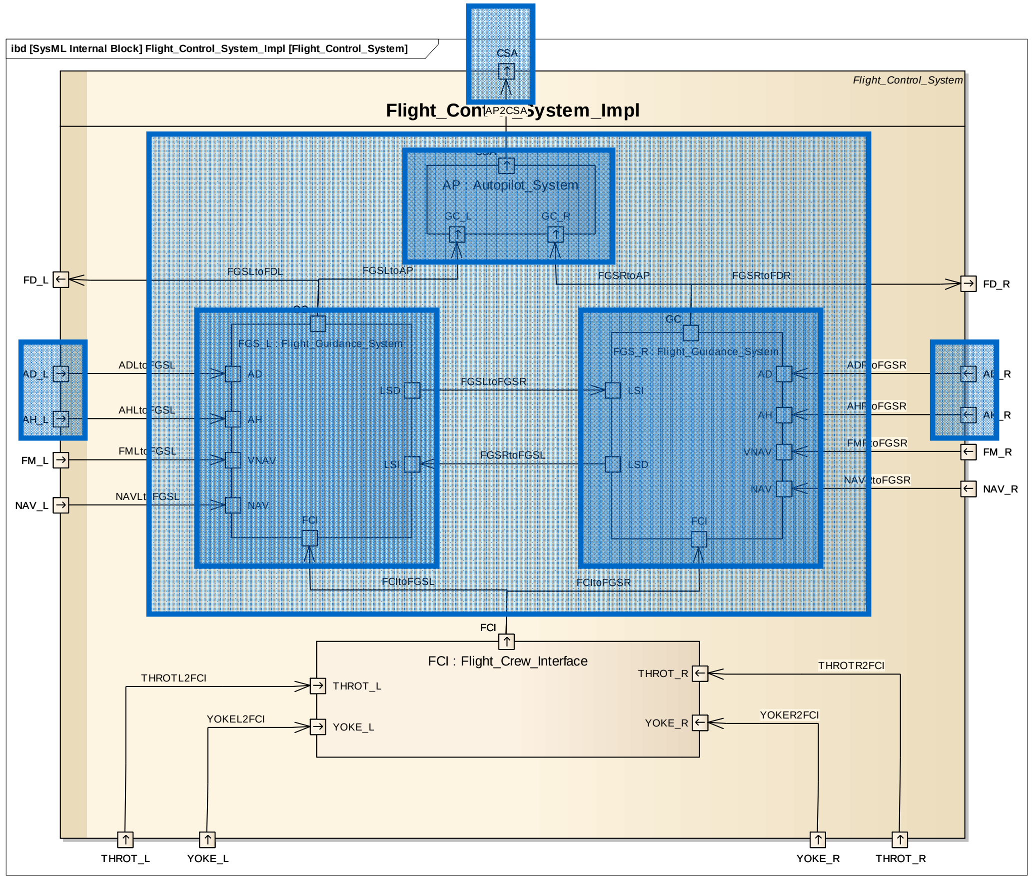
- Relies upon
  - Guarantees provided by components & design patterns
  - Structural properties of model
  - Resource allocation feasibility
  - Probabilistic system-level failure characteristics

Principled mechanism for "passing the buck"



Compositional reasoning for FCS (example)

- Want to prove a **transient response** property
  - The autopilot will not cause a sharp change in pitch of aircraft.
  - Even when one FGS fails and the other assumes control
- Given **assumptions** about the environment
  - The sensed aircraft pitch from the air data system is within some absolute bound and doesn’t change too quickly
  - The discrepancy in sensed pitch between left and right side sensors is bounded.
- And **guarantees** provided by components
  - When a FGS is active, it will generate an acceptable pitch rate
- As well as facts provided by **architecture**
  - Leader selection: at least one FGS will always be active (modulo one “failover” step)




```
transient_response_1 : assert true ->
  abs(CSA.CSA_Pitch_Delta) < CSA_MAX_PITCH_DELTA ;
transient_response_2 : assert true ->
  abs(CSA.CSA_Pitch_Delta - prev(CSA.CSA_Pitch_Delta, 0.0))
     < CSA_MAX_PITCH_DELTA_STEP ;
```

Conclusions

- Model-based development has been key to our adoption of formal methods
- Current work is expanding the size and scope of systems/models that can be analyzed
- There are many good reasons to use formal methods for verification and certification…
- But follow the money