Will This Be Formal?

Dr. Steven P. Miller
July 15, 2008

Presentation Overview

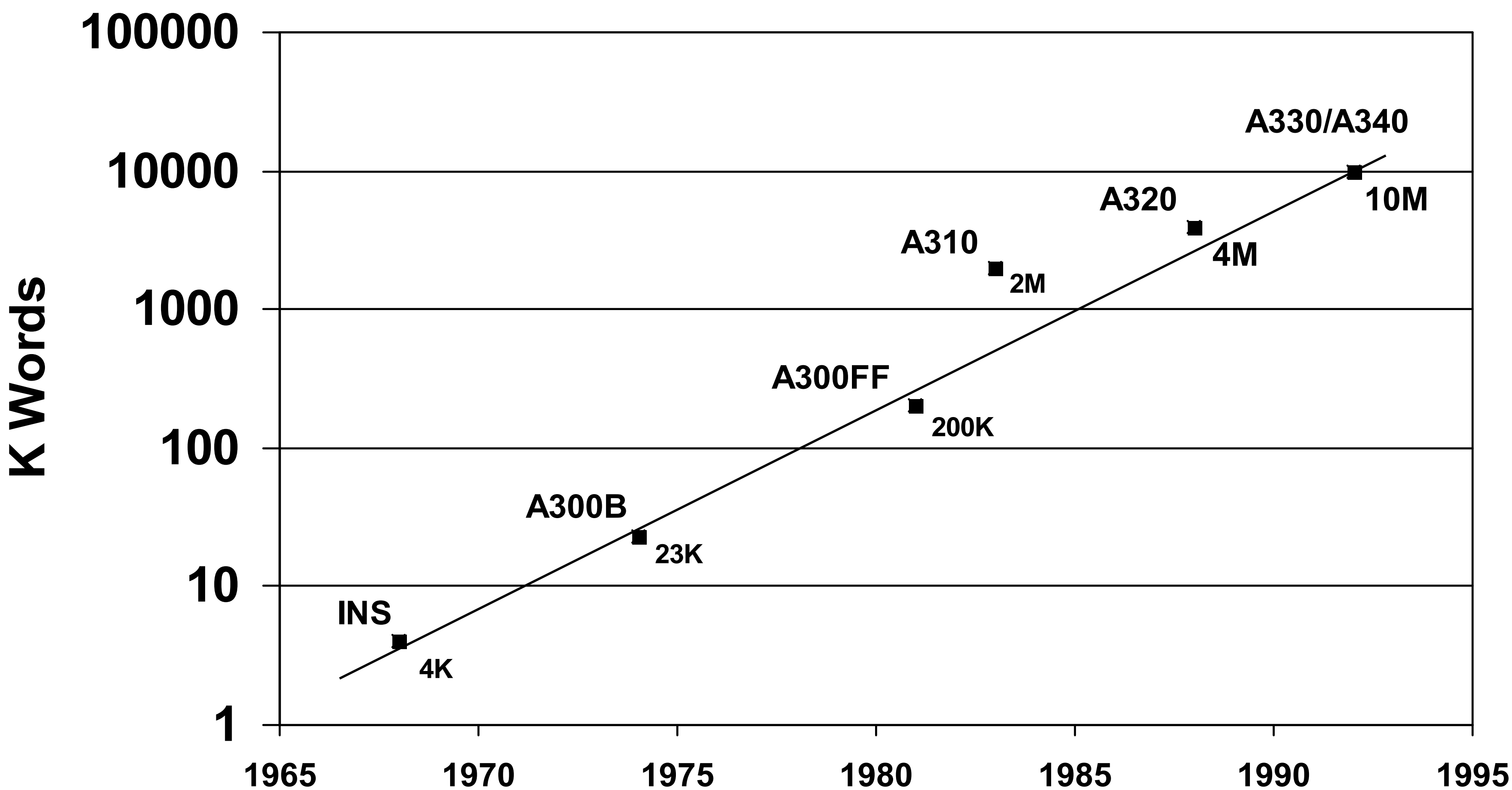
- What Problem are We Solving?
- Who Are We?
- What are Formal Methods?
- Examples of Using Formal Methods
- Challenges and Future Directions

What Problem Are We Solving?

- **Increasing Size and Complexity of Critical Systems**
  - Safety critical, security critical, and mission critical
  - Exponential growth in size and complexity

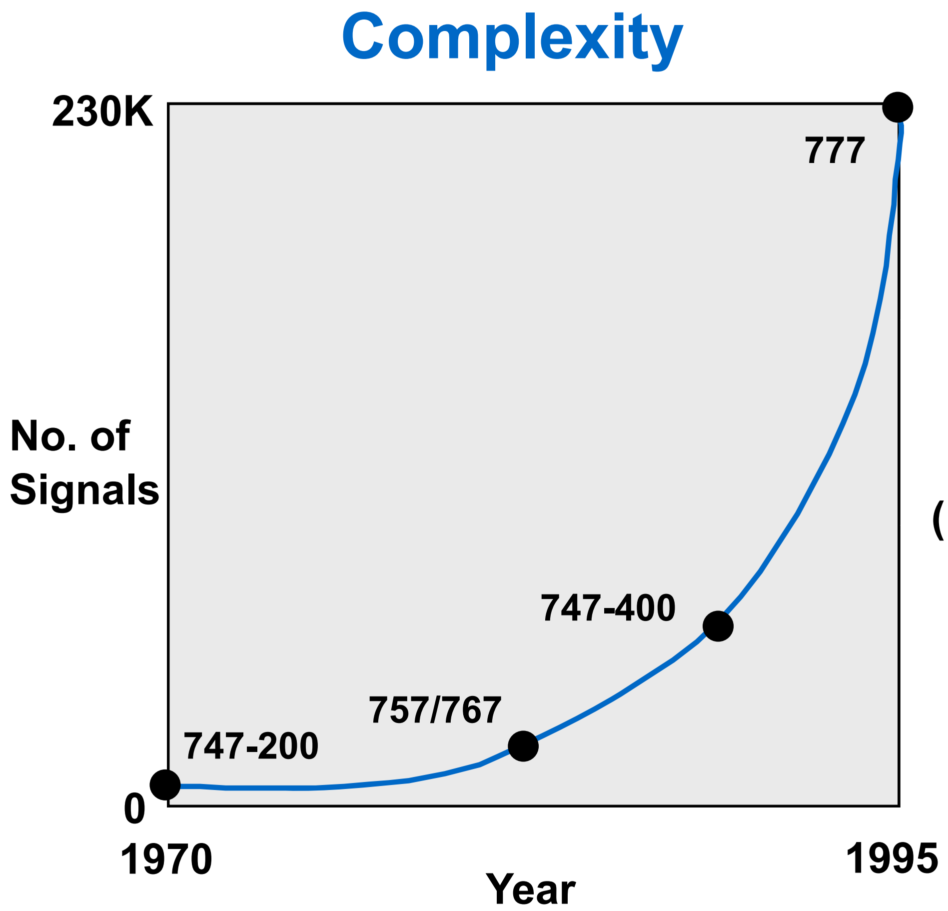
- **Rapidly Growing Cost of Verification**
  - Exponential growth in cost
  - Becoming the limiting factor in deployment

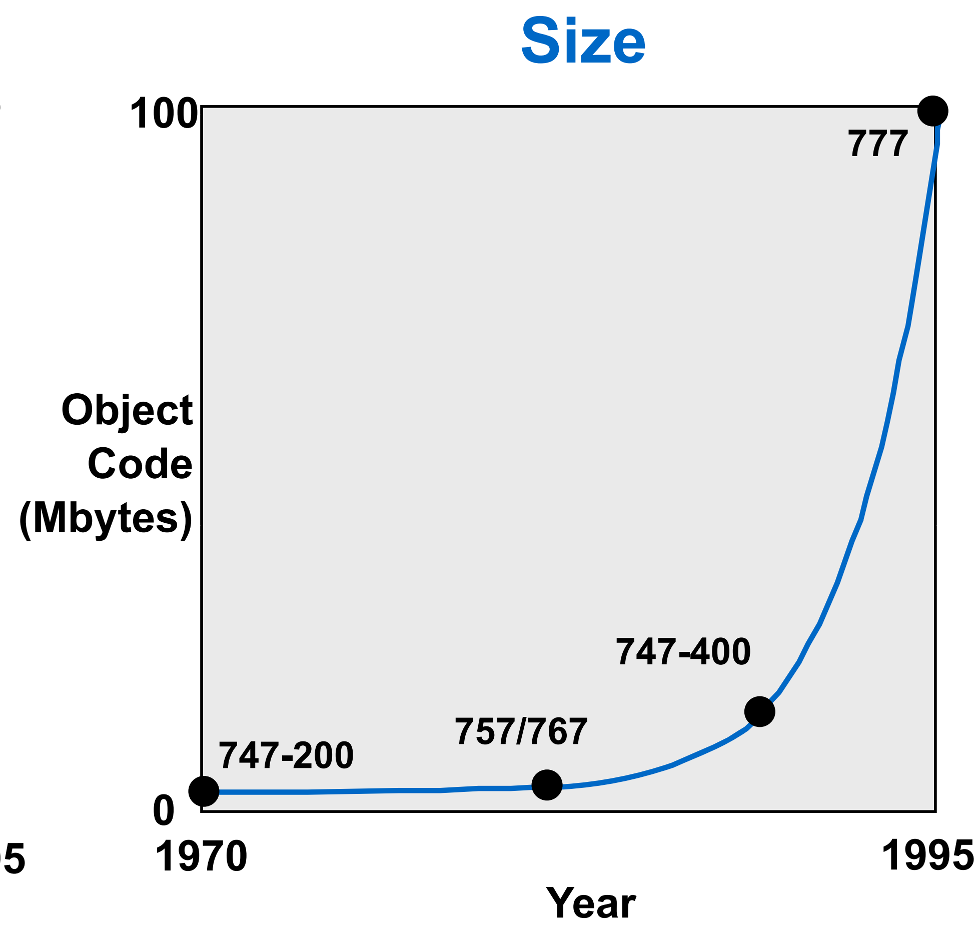
Airborne Software Doubles Every Two Years



J.P. Potocki De Montalk, Computer Software in Civil Aircraft, Sixth Annual Conference on Computer Assurance (COMPASS '91), Gaithersberg, MD, June 24-27, 1991.

Similar Growth Has Been Seen by Boeing

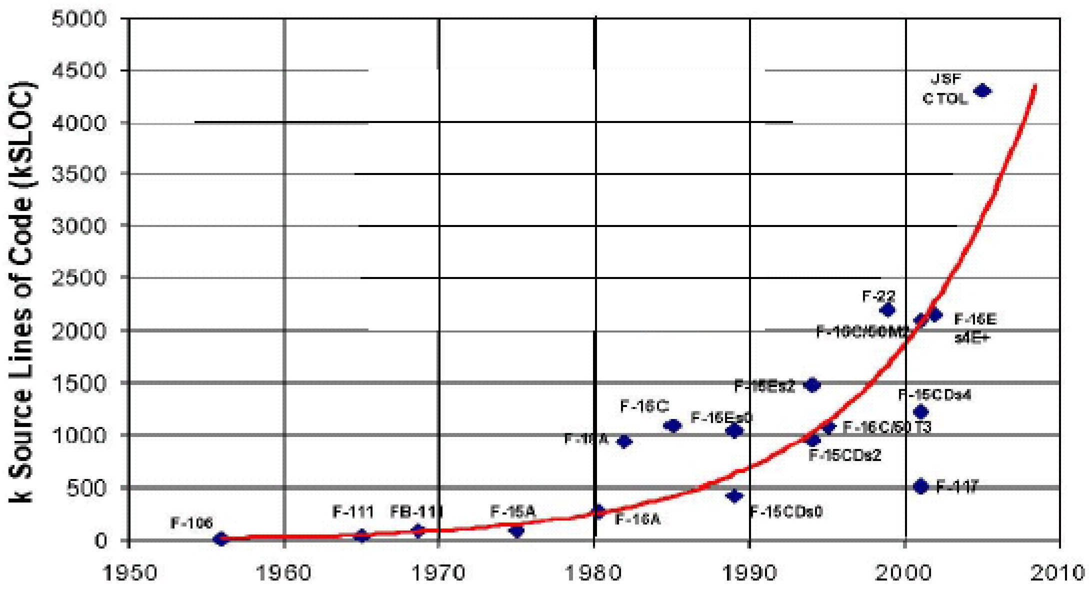




DoD software is growing in size and complexity

U.S. AIR FORCE

Total Onboard Computer Capacity (OFP)

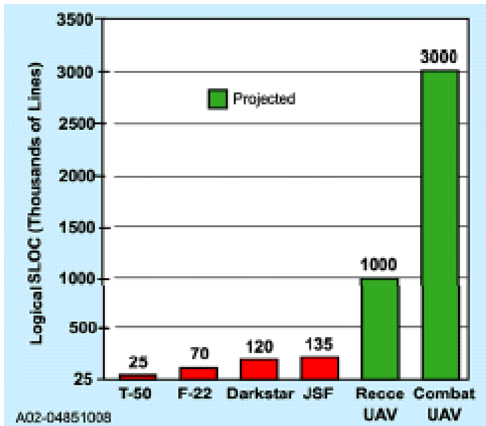


Source: "Avionics Acquisition, Production, and Sustainment: Lessons Learned -- The Hard Way", NDIA Systems Engineering Conference, Mr. D. Gary Van Oss, October 2002.

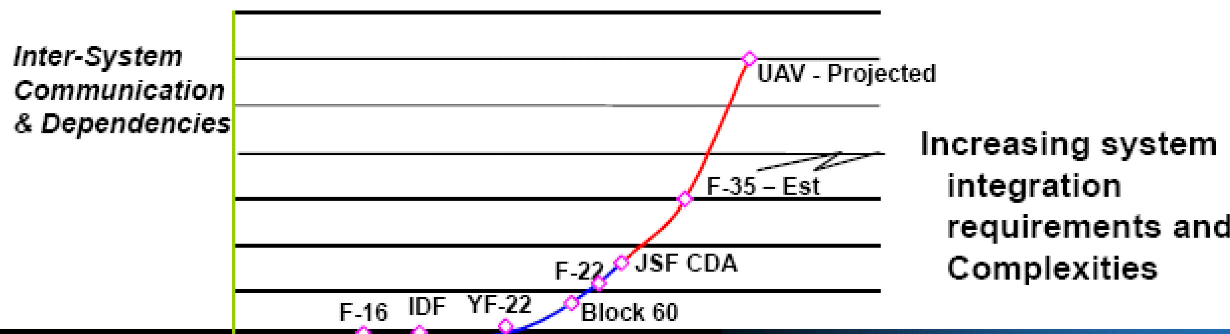
Robert Gold, OSD

Emerging Software Size and Complexity

U.S. AIR FORCE



- Advanced system attributes (on-board *intelligence* and *adaptive control laws*) will be required to accommodate emerging functional requirements.
- This will increase the size and complexity of control systems beyond the capability of current V&V practices.



Projected Exponential Increase in SW Size and Complexity

Criteria for Formal Verification

- Is the Problem Important?
- Are High Fidelity Models Available?
- Can the Properties of Interest be Formalized?
- Are the Right Analysis Tools Available?

Presentation Overview

What Problem are We Solving?

➡ Who Are We?

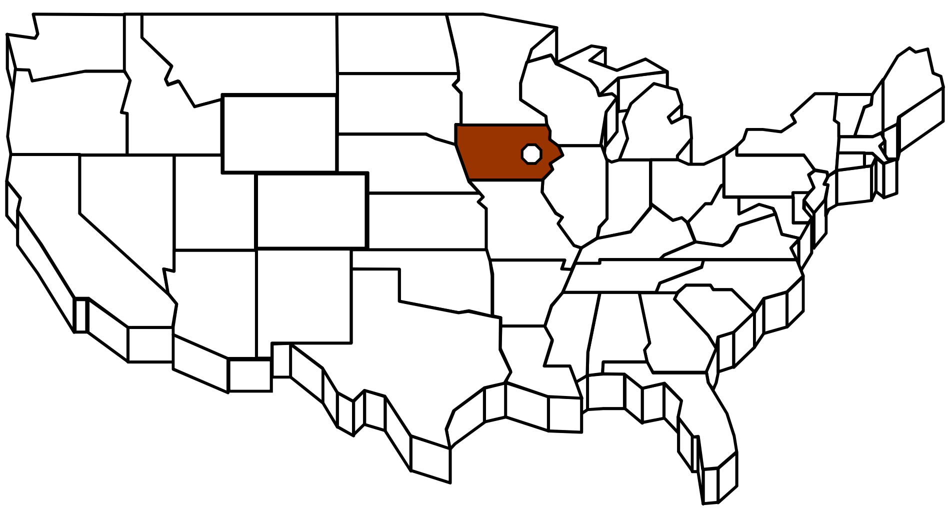
What are Formal Methods?

Examples of Using Formal Methods

Challenges and Future Directions

Rockwell Collins

- **Headquartered in Cedar Rapids, Iowa**
- **20,000 Employees Worldwide**
- **2007 Sales of $4.42 Billion**



Domestic

California
Carlsbad
Cypress
Irvine
Los Angeles
Pomona
Poway
San Francisco
San Jose
Tustin
Florida
Melbourne
Miami
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
Washington
Kirkland
Renton
Seattle
Washington, DC

International

Africa
Johannesburg, South Africa
Asia
Bangkok, Thailand
Beijing, China
Hong Kong
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

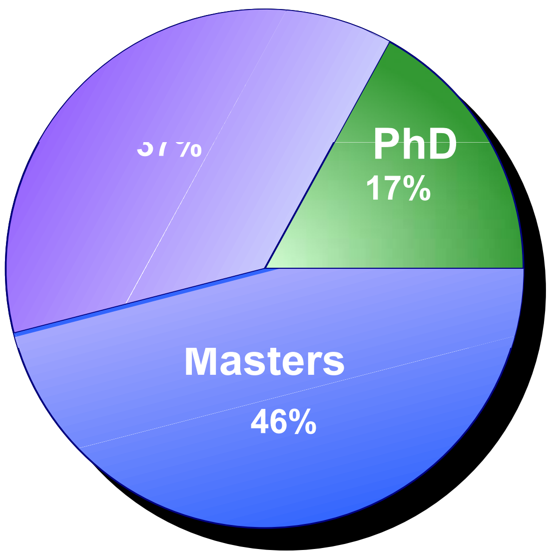
Rockwell Collins' core business is based on the delivery of *High Assurance* Systems

- **Commercial / Military Avionics Systems**
- **Communications**
- **Navigation & Landing Systems**
- **Flight Control**
- **Displays**

*"Working together creating **the most trusted source** of communication and aviation electronic solutions"*

Advanced Technology Center

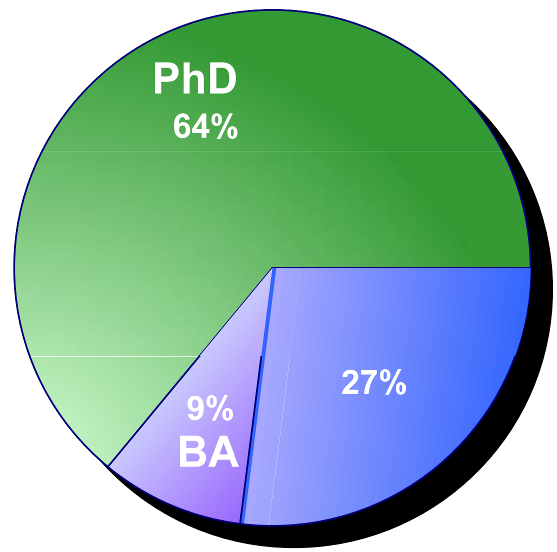
Identify, acquire, develop and transition value-driven technologies to support the continued growth of Rockwell Collins.



Technologists: 173
Administrators: 10
Technicians: 31

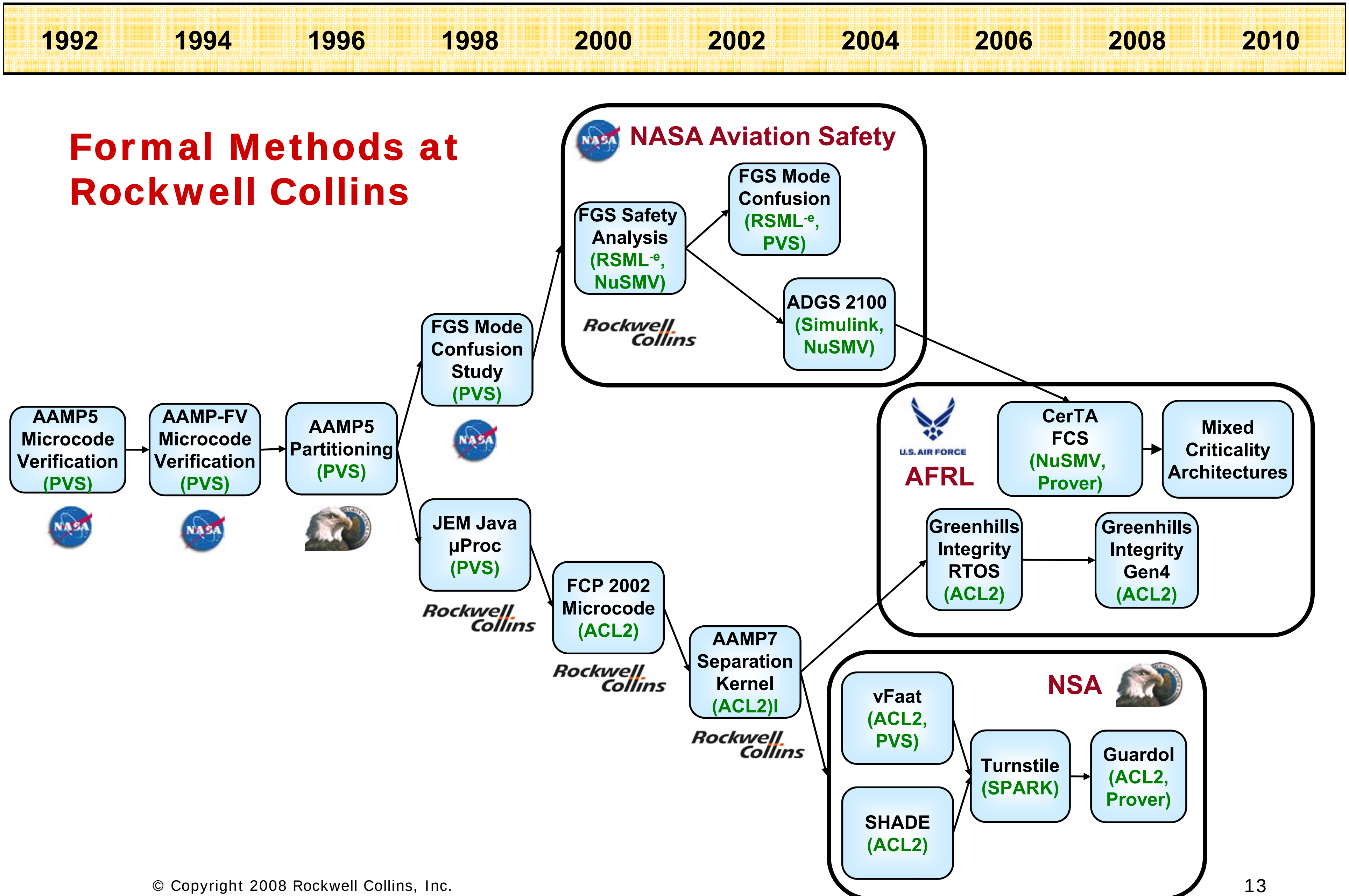
Automated Analysis Section

Technologists: 10
Administrators: 1



Applies mathematical tools and reasoning to the production of high assurance systems.

Formal Methods at Rockwell Collins



Presentation Overview

What Problem are We Solving?

Who Are We?

➡ What are Formal Methods?

Examples of Using Formal Methods

Challenges and Future Directions

What are Formal Methods?

> Mathematically-based techniques for the specification, development and verification of software and hardware systems.

Wikipedia, 8 April 2008

- **Specification**
  - **Textual notations (Z, B, VDM, CSP, ...)**
  - **Tabular notations (Parnas Tables, SCR, RSML, ...)**
  - **Graphical notations (SCADE, Simulink, Statecharts ...)**
- **Development**
  - **Stepwise refinement with proofs of correctness**
  - **Model-Based Development**
  - **Automated code generation**
- **Verification**
  - **Lightweight static analysis**
  - **Theorem proving (ACL2, PVS, HOL, ...)**
  - **Model-checking (SMV, SAL, Prover, ...)**

Specification

Textual (Z, VDM, PVS, Lustre, …)

```
node Thrust_Required(
   FG_Mode : FG_Mode_Type ;
   Airborne : bool ;
   In_Flare : bool ;
   Emergency_Descent : bool;
   Windshear_Warning : bool ;
   In_Eng_Accel_Zone : bool ;
   On_Ground : bool)
returns (IsTrue : bool) ;

let

IsTrue =
   (FG_Thrust_Mode(FG_Mode) and
    Airborne)
 or
   (Airborne and Emergency_Descent)
 or
   Windshear_Warning
 or
   ((FG_Mode = ThrottleRetard) and
     In_Flare)
 or
   (In_Eng_Accel_Zone and On_Ground) ;
tel ;
```

Tabular (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

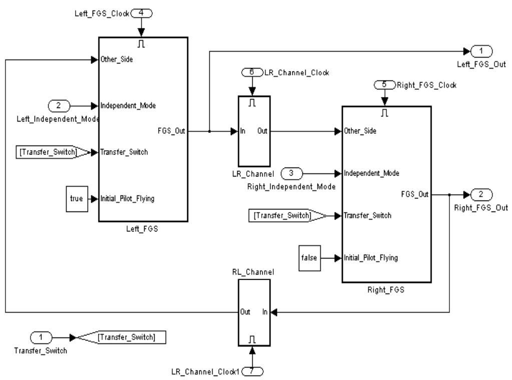
When_Turn_FD_On

Condition:

AND	OR						
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).

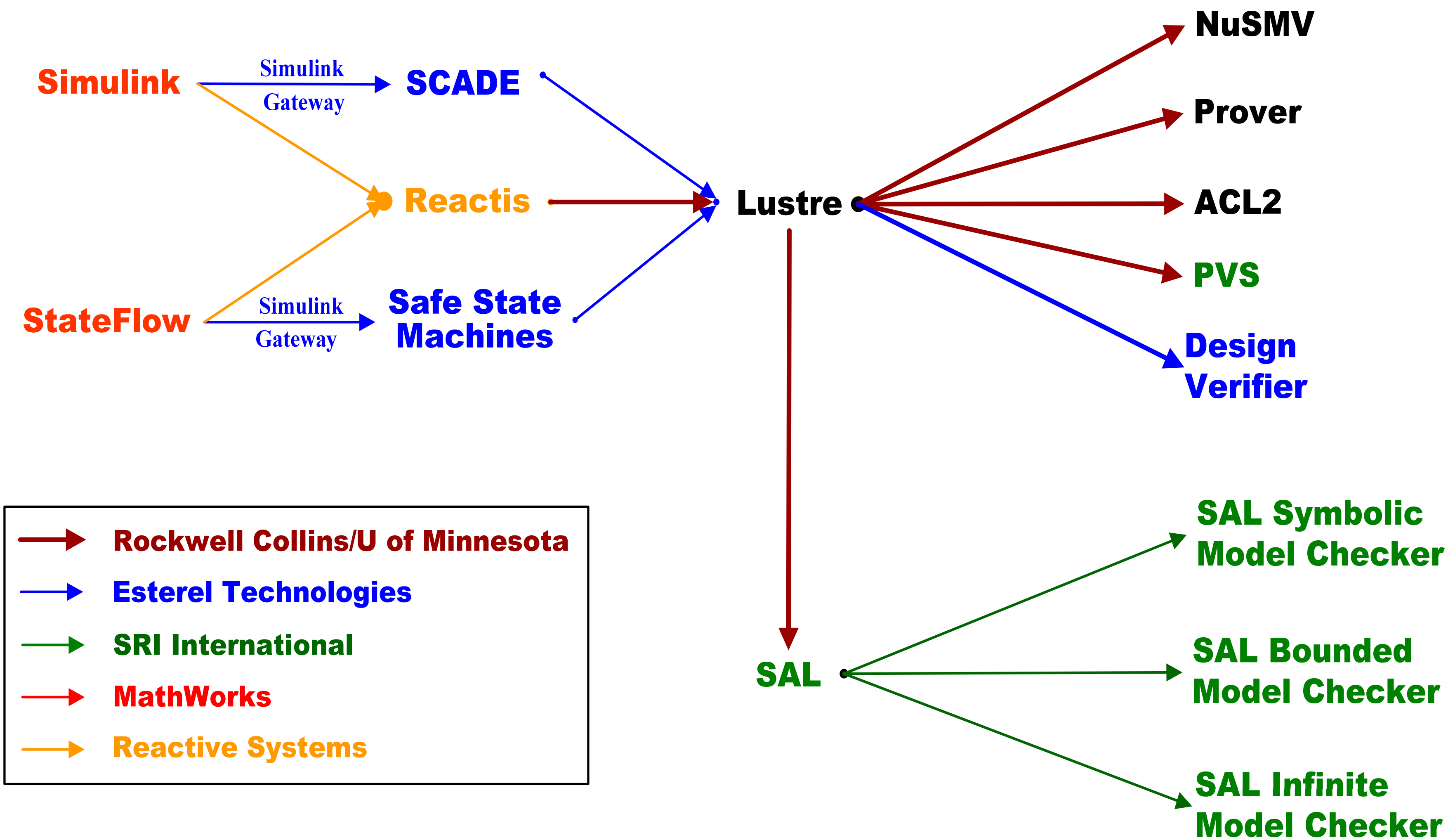
Graphical (SCADE, Simulink)



Model-Based Development

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

Verification - Rockwell Collins Translation Framework



Translators Optimize for Specific Analysis Tools

Model	CPU Time (For NuSMV to Compute Reachable States)		Improvement
	Before	After	
Mode1	> 2 hours	11 sec	> 650x
Mode2	> 6 hours	169 sec	> 125x
Mode3	> 2 hours	14 sec	> 500x
Mode4	8 minutes	< 1 sec	480x
Arch	34 sec	< 1 sec	34x
WBS	29+ hours	1 sec	105,240x

Presentation Overview

What Problem are We Solving?

Who Are We?

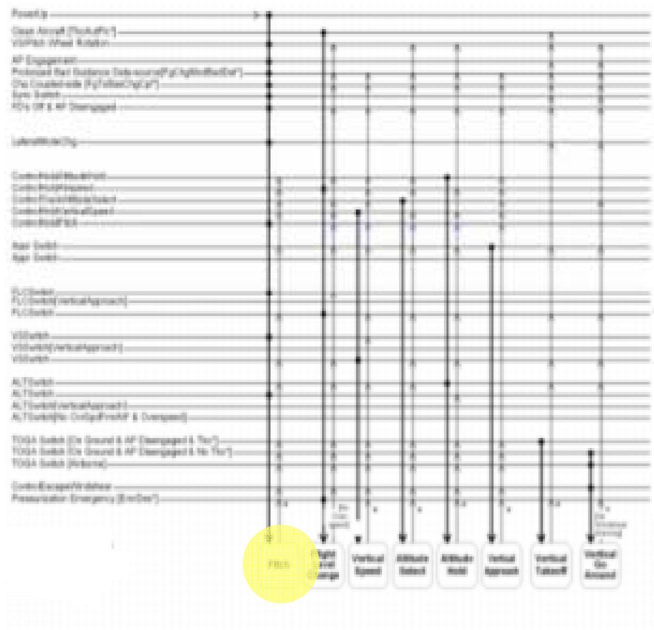
What are Formal Methods?

➡ **Examples of Using Formal Methods**

Challenges and Future Directions

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 in Early Requirements Models

Modeled in Simulink

Translated to NuSMV

6.8 x 10^{21} Reachable States

Mode Controller B

ADGS-2100 Adaptive Display & Guidance System

Modeled in Simulink

Translated to NuSMV

4,295 Subsystems

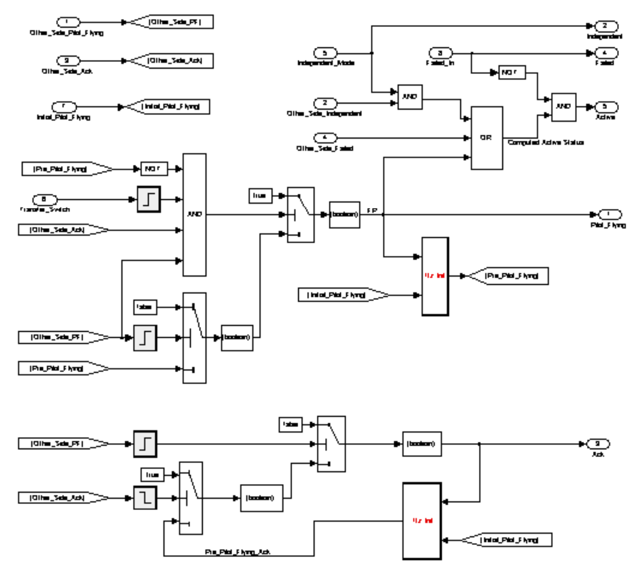
16,117 Simulink Blocks

Over 10^{37} Reachable States

Example Requirement:

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

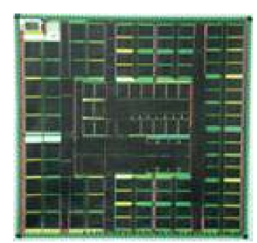
Counterexample Found in 5 Seconds



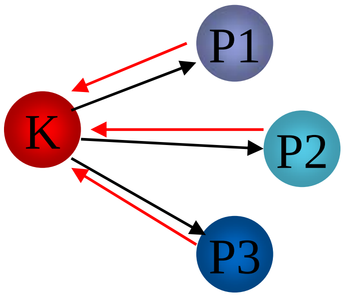
Checked 573 Properties - Found and Corrected 98 Errors in Early Design Models

AAMP7G Certified Microprocessor

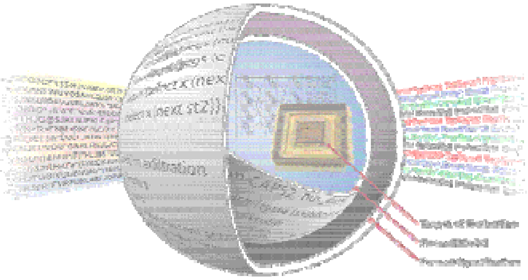
- Rockwell Collins proprietary microprocessor
- Formal proof of the MILS security partitioning implemented in the AAMP7G microprocessor
- Example of the industrial use of theorem proving using ACL2
- Developed formal description of separation for uniprocessor, multipartition system (GWV)
- Modeled trusted AAMP7G microcode in ACL2
- Constructed machine-checked proof of separation of the AAMP7G model using ACL2
- Model subject of intensive code-to-spec review with AAMP7G microcode
- Satisfied formal methods requirements for NSA AAMP7G certification awarded in May 2005
  - *"capable of simultaneously processing unclassified through Top Secret Codeword Information"*
  - *"verified using Formal Methods techniques as specified by the EAL-7 level of the Common Criteria"*



Greenhills Integrity-178B Real-Time OS Evaluation



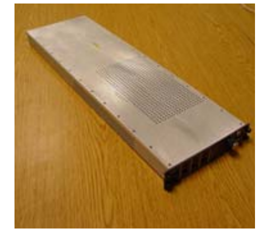
- **Formal proof of the MILS security partitioning implemented in the Integrity-178B Real-Time OS**
- **Example of the industrial use of theorem proving using ACL2**
- **Generalized the formal description of separation to describe the more dynamic scheduling managed by the OS (GWVr2)**
- **Modeled in ACL2 the target-independent C code implementing the Integrity-178B kernel.**
- **Constructed machine-checked proof of separation for the Integrity-178B kernel**
- **Model, analysis approach and proofs subject to intensive multi-national review**
- **Satisfied US Government SKPP (EAL6+), as well as Common Criteria v2.3 EAL7 ADV requirements**
  - ***Final certification pending NSA penetration testing***



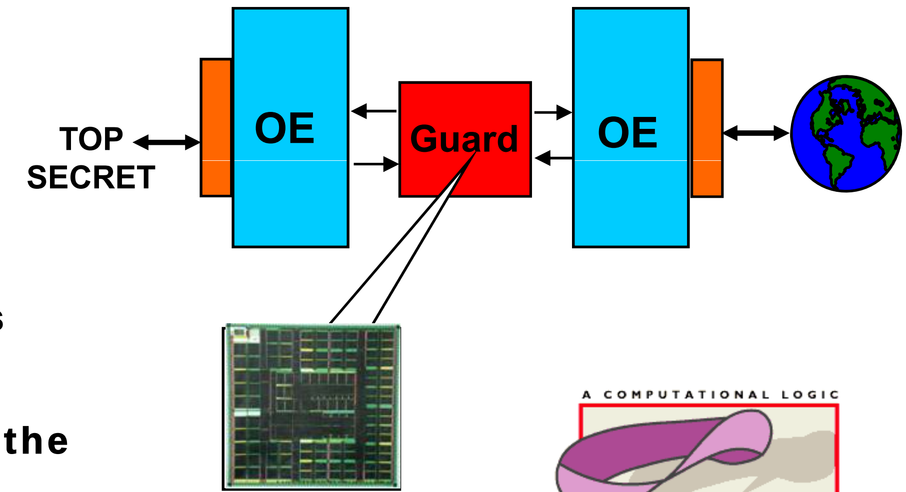
A COMPUTATIONAL LOGIC
ACL2
APPLICATIVE COMMON LISP

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 to DCID 6/3 PL-5.



AAMP7G

- **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**

CerTA FCS Phase I

- **Sponsored by the Air Force Research Labs**
  - Air Vehicles (RB) Directorate - Wright Patterson
- **Investigate Roles of Testing and Formal Verification**
  - Can formal verification complement or replace some testing?
- **Example Model – Lockheed Martin Adaptive UAV Flight Control System**
  - Redundancy Management Logic in the Operational Flight Program (OFP)
  - Well suited for verification using the NuSMV model-checker

Lockheed Martin Aero

- Based on Testing
- Enhanced During CerTA FCS
  - Graphical Viewer of Test Cases
  - Support for XML/XSLT Test Cases
  - Added C++ Oracle Framework
- Developed Tests from Requirements
- Executed Tests Cases on Test Rig

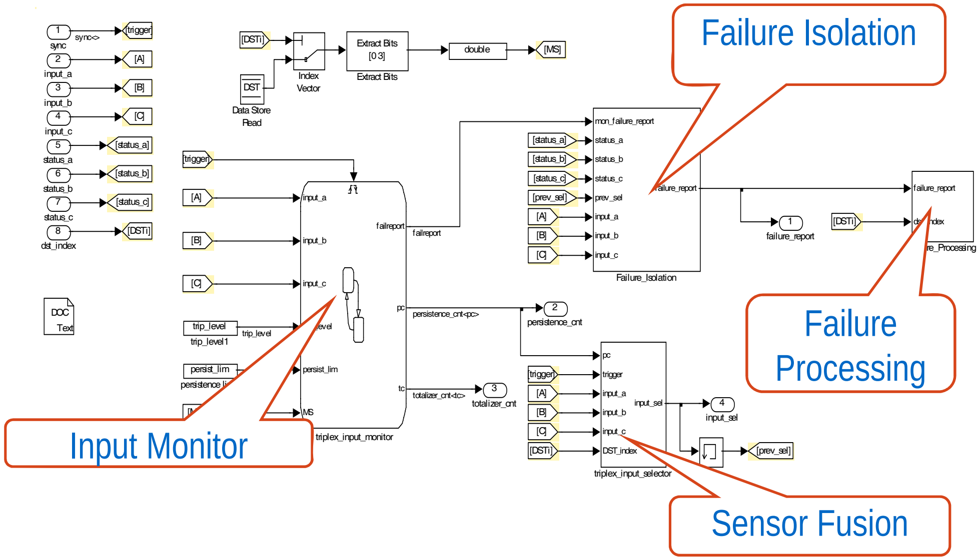
Rockwell Collins

- Based on Model-Checking
- Enhanced During CerTA FCS
  - Support for Simulink blocks
  - Support for Stateflow
  - Support for Prover model-checker
- Developed Properties from Requirements
- Proved Properties using Model-Checking

CerTA FCS Phase I - OFP Redundancy Management Logic

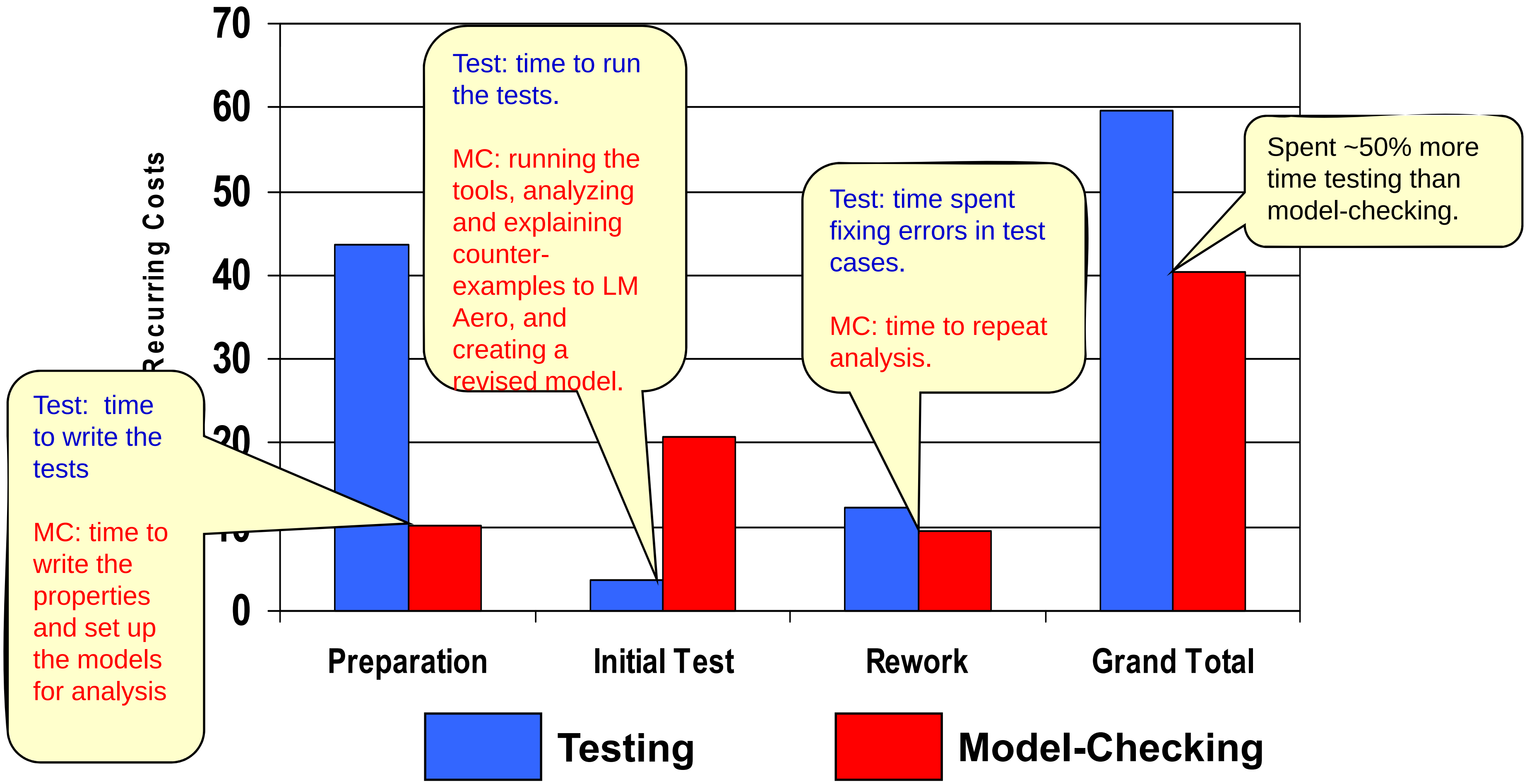
For Each of Ten Control Surfaces

- **Triplex Voter**
  - **Input monitor, sensor fusion, and failure isolation**
- **Failure Processing**
  - **Logs failures into a data store**
- **Reset Manager**
  - **Reset logic for sensors and control surfaces (not shown)**



	Subsystems / Blocks	Charts / Transitions	Truth Table Cells	Reachable State Space	Properties
Triplex voter	10 / 96	3 / 35	198	$6.0 * 10^{13}$	48
Failure processing	7 / 42	0 / 0	0	$2.1 * 10^{4}$	6
Reset manager	6 / 31	2 / 26	0	$1.32 * 10^{11}$	8
Total	23 / 169	5 / 61	198	N/A	62

CerTA FCS Phase I - Testing and Model Checking Recurring Costs



CerTA FCS Phase I – Errors Found

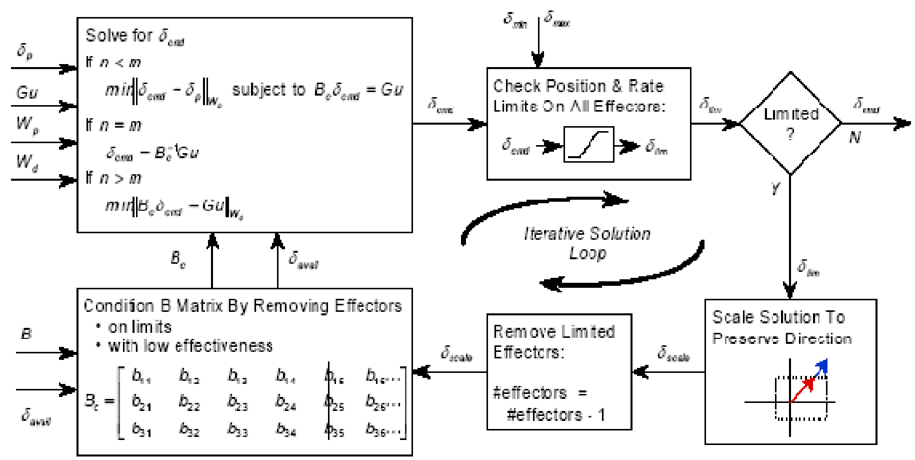
	Model Checking	Testing
Triplex Voter	5	0
Failure Processing	3	0
Reset Manager	4	0
Total	12	0

- **Model-Checking Found 12 Errors that Testing Missed**
- **Spent More Time on Testing than Model-Checking**
  - 60% of total on testing vs. 40% on model-checking

Model-checking was more cost effective than testing at finding design errors.

CerTA FCS Phase II

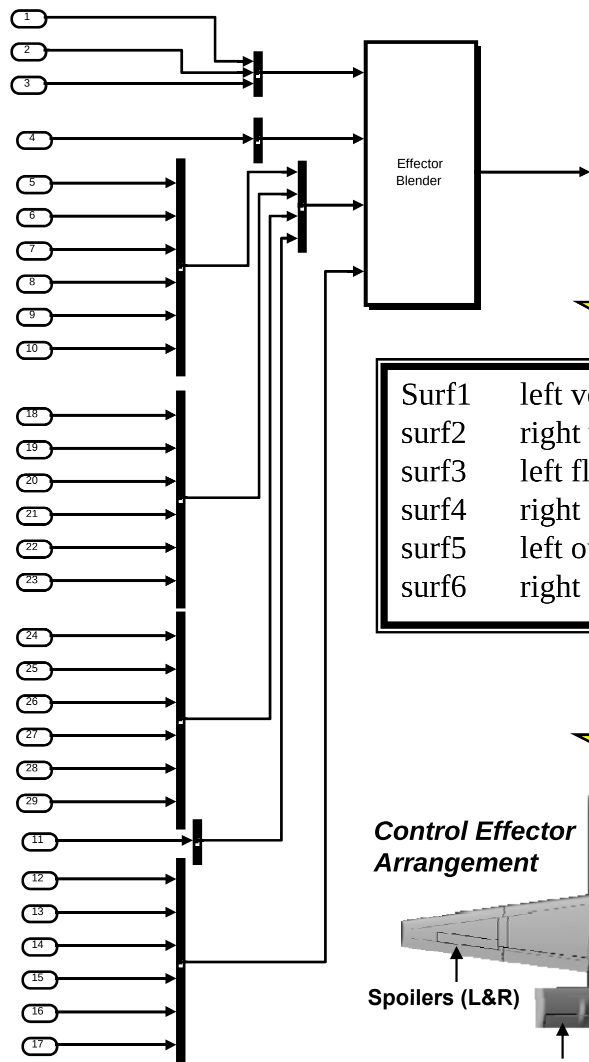
- **Sponsored by the Air Force Research Labs**
  - Air Vehicles (RB) Directorate - Wright Patterson
- **Can Model-Checking be Used on Infinite State Systems?**
  - Large, numerically intensive, non-linear systems
- **Example Model**
  - Lockheed Martin Adaptive UAV Flight Control System
  - Effector Blender (EB)
  - Generates actuator commands for aircraft control surfaces
  - Matrix arithmetic of floating point numbers



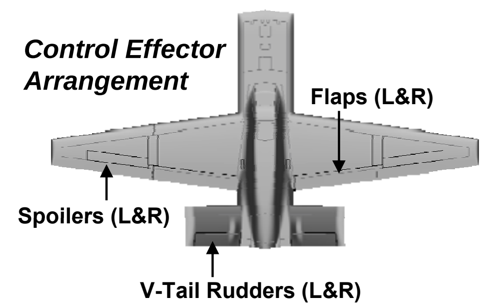
- **Challenges**
  - Identifying the right properties to verify
  - Verification of floating point numbers
  - Verification of Stateflow *flowcharts* with cyclic transition paths
  - Compositional verification to scale to entire Effector Blender

CerTA FCS Phase II – Effector Blender

- **Generates Actuator Commands**
  - Six control surfaces
  - Adapts its behavior as aircraft state changes
  - Iterative algorithm that repeatedly manipulates a 3 x 6 matrix of floating point numbers
- **Large Complex Model**
  - Inputs
    - 32 floating point inputs
    - 3 x 6 matrix of floating point values
  - Outputs
    - 1 x 6 vector of floating point values
  - 166 Simulink subsystems
  - 2000+ basic Simulink blocks
  - Huge reachable state space
- **Completely Functional**
  - No internal state



Surf1	left vertical tail
surf2	right vertical tail
surf3	left flap
surf4	right flap
surf5	left outboard spoiler
surf6	right outboard spoiler



CerTA FCS Phase II – What to Verify?

- **No Explicit Requirements for the Effector Blender Model**
  - Requirements defined for Effector Blender + aircraft model
  - Addition of aircraft model pushes verification beyond current tools
- **Avoid Properties Verifiable by Other Means**
  - Control theory – stability, tracking performance, feedback design …
  - Simulation – design validation
  - Implementation – code generation/compilation, scheduling, …
- **Focus on the Consistency of the Effector Blender Model**
  - Relationships the model should always maintain
  - Partial requirements specification
- **Preservation of Control Surface Limits**
  - EB computes upper and lower limits for each control surface command
  - Function of aircraft design, aircraft state, and max extension per cycle
  - Commanded extension should always be between these limits

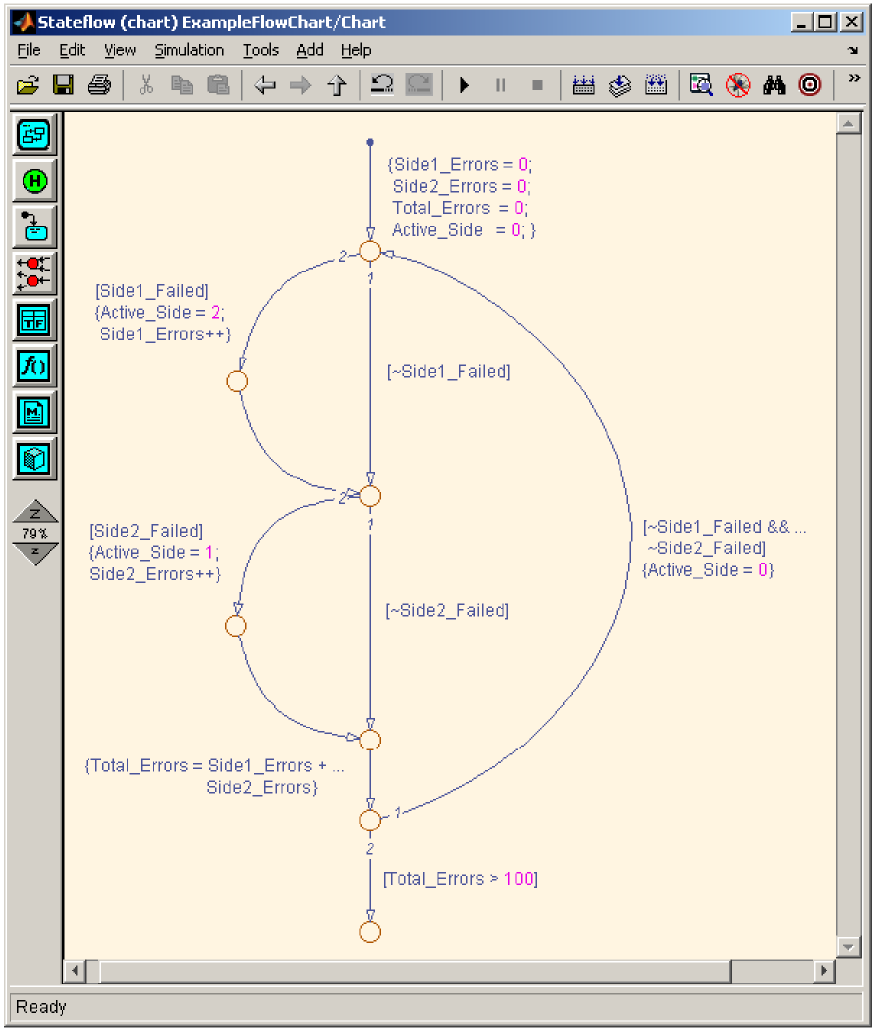
CerTA FCS Phase II – Verification of Floating Point Numbers

- **Floating Point Numbers**
  - Fixed number of bits with a movable decimal (radix) point
  - No decision procedures for floating point numbers available
- **Real Numbers**
  - Real numbers have unbounded size and precision
  - Would hide errors caused by limitations of floating point arithmetic
  - Control theory problems are inherently non-linear
  - Decision procedures for non-linear real numbers have exponential cost
- **Solution - Translate Floating Point Numbers into Fixed Point**
  - Extended translation framework to automate this translation
  - Convert floating point to fixed point (scaling provided by user)
  - Convert fixed point into integers (use bit shifting to preserve magnitude)
  - Shift from NuSMV (BDD-based) to Prover (SMT-solver) model checker
- **Advantages & Issues**
  - Use bit-level integer decision procedures for model checking
  - Results unsound due to loss of precision
  - Highly likely to find errors – very valuable tool for debugging

CerTA FCS Phase II – Verification of Stateflow Flowcharts

- **Stateflow Flowcharts**
  - No explicit states
  - Stateflow junctions
  - Cyclic paths
  - Transitions modify local state variables
  - Imperative programming

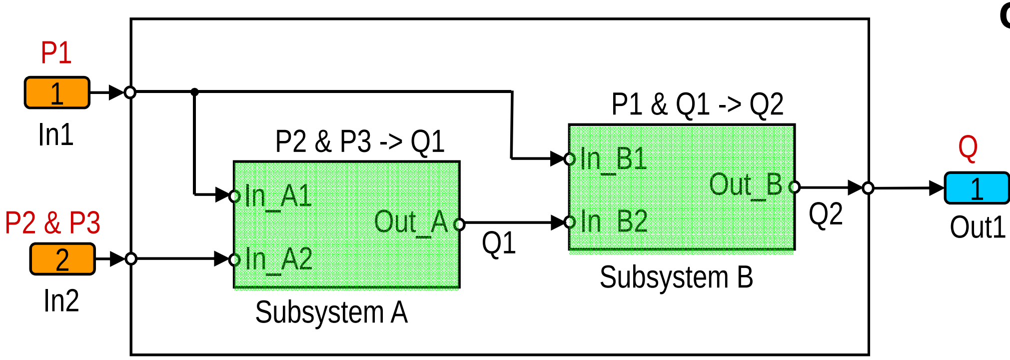
- **Solution**
  - Extend translator to support flowcharts
  - Require a parameter that specifies the maximum times any cycle will be executed



CerTA FCS Phase II – Compositional Verification

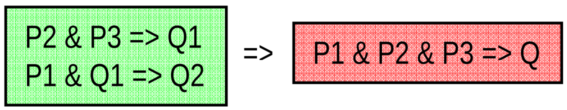
Typical Specification

- Models are typically organized in a hierarchy of subsystems
- Subsystems are often nested several levels deep
- Most of the complexity is in the leaf subsystems
- Leaf subsystems can often be verified through model checking



Composition of Subsystems

- Tends to be simple
- Lends itself well to theorem proving

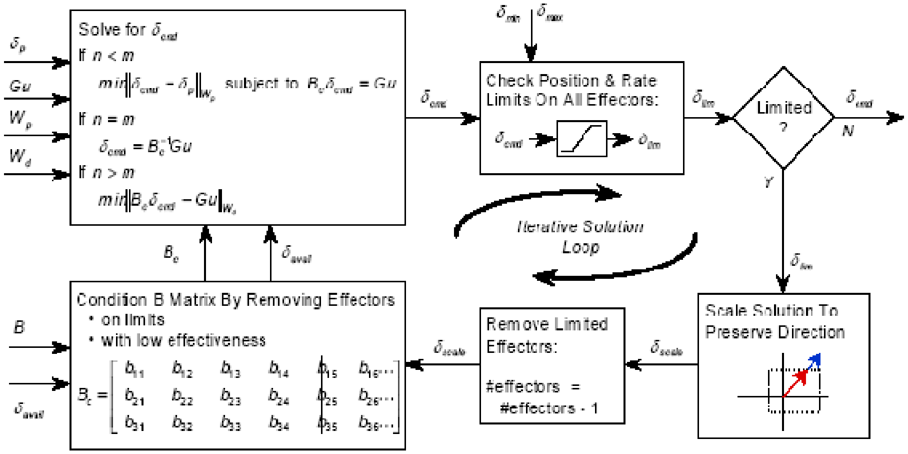


Issues

- Need to avoid circular reasoning to ensure soundness
- Can be ensured by eliminating cyclic dependencies between <u>atomic</u> subsystems
- Identifying the right leaf level invariants to support composition
- Complexity of the proof obligations for the intermediate levels
- Lack of a unified automated verification system

CerTA FCS Phase II - Results

- **Can Model-Checking be Used on Infinite State Systems?**
  - **Large, numerically intensive, non-linear systems**
- **Effector Blender**
  - **Inputs**
    - 32 floating point inputs
    - 3 x 6 matrix of floating point values
  - **Outputs**
    - 1 x 6 vector of floating point values
  - **166 Simulink subsystems**
  - **2000+ basic Simulink blocks**



- **Errors Found**
  - **Five previously unknown errors that would drive actuators past their limits**
  - **Several implementation errors were being masked by defensive programming**
- **Areas for Future Research**
  - **Decision procedures for floating point arithmetic**
  - **Interval arithmetic**
  - **Automation for compositional verification**

Rockwell Collins

Presentation Overview

What Problem are We Solving?

Who Are We?

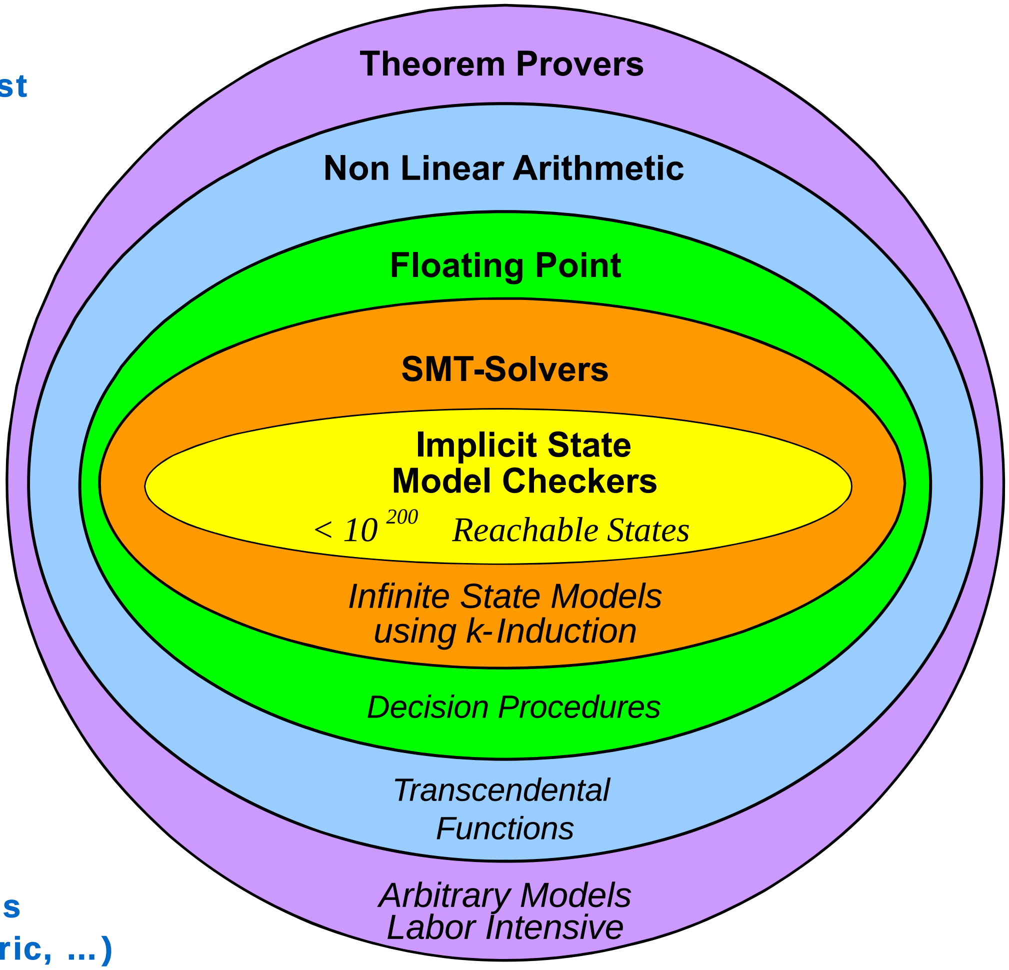
What are Formal Methods?

Examples of Using Formal Methods

➡ **Challenges and Future Directions**

Extending the Verification Domain

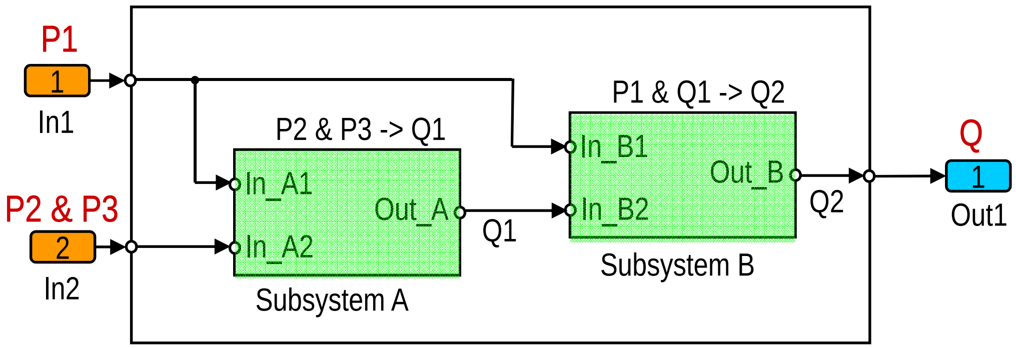
- **Theorem Provers**
  - Deal with arbitrary models
  - Concerns are ease of use and labor cost
- **Large Finite Systems (<10^{200} States)**
  - Implicit state (BDD) model checkers
  - Easy to use and very effective
- **Very Large or Infinite State Systems**
  - SMT-Solvers
  - Large integers and reals
  - Limited to linear arithmetic
  - Ease of use is a concern
- **Floating Point Arithmetic**
  - Most modeling languages use floating point (not real) numbers
  - Decision procedures
- **Non-Linear Arithmetic**
  - Multiplication / division of real variables
  - Transcendental functions (trigonometric, …)
  - Essential to navigation systems



Compositional Verification

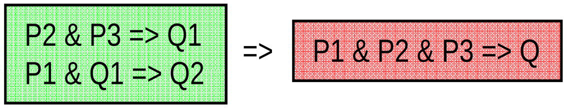
Typical Model-Based Specification

– Models are organized in a hierarchy of subsystems several levels deep
– Most of the complexity is in the leaf models
– Leaf models can often be verified through model checking



Composition of Subsystems

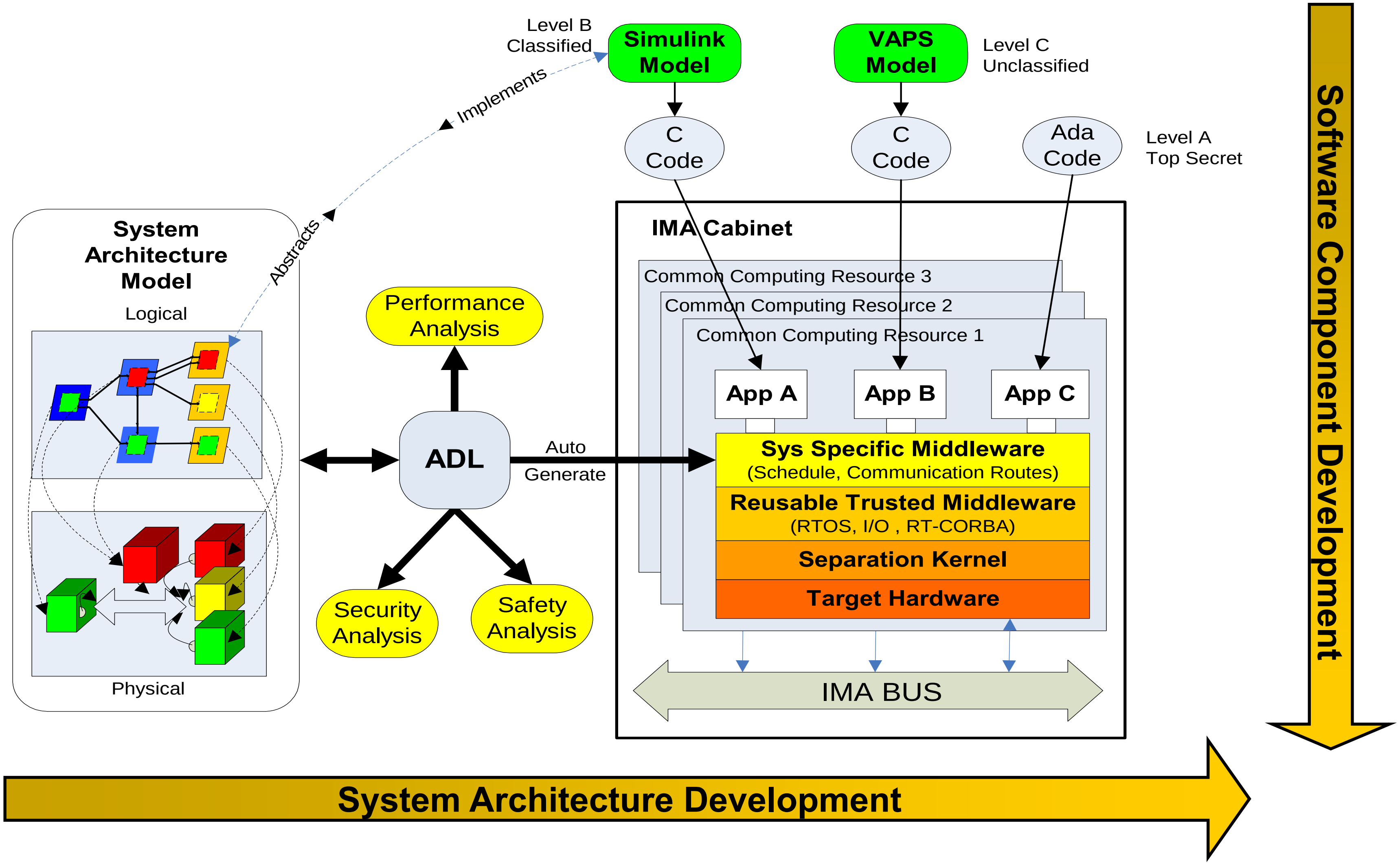
– Tends to be simple
– Well suited for theorem proving



Issues

– Lack of a unified automated verification system
  • Use model-checking to verify leaf models and theorem proving for composition
– Avoid circular reasoning to ensure soundness
  • Can be ensured by eliminating cyclic dependencies between atomic subsystems
– Identifying the right leaf level invariants to support composition
– Complexity of the proof obligations for the intermediate levels

System Architectural Modeling & Analysis



Conclusions

- **Formal Methods *Are* Practical and *Are* Being Widely Used**
  - Model Based Development is the industrial face of formal methods
  - The engineers get to pick the modeling tools!
  - Semantics of some of the commercial tools could be improved

- **Formal Verification Tools Are Being Used in Industry**
  - Key is to verify the models the engineers are already building
  - Large portions of existing systems can be verified with model checkers
  - Model checkers are only going to get better
  - Theorem proving can be used on stable industrial systems

- **Directions for the Future Work**
  - Making verification tools more powerful and easier to use
  - Addressing scalability through compositional verification
  - Integration of theorem proving and model checking
  - Modeling and analysis of system architectural models

For More Information

http://shemesh.larc.nasa.gov/fm/fm-collins-intro.html

- Whalen, M., Cofer, D., Miller, S., Krogh, B., Storm, W.: Integration of Formal Analysis into a Model-Based Software Development Process. In 12th International Workshop on Formal Methods for Industrial Critical Systems (FMICS2007), Berlin, Germany (2007).
- Whalen, M., Innis, J., Miller, S., Wagner, L.: ADGS-2100 Adaptive Display & Guidance System Window Manager Analysis, CR-2006-213952, NASA (2006).
- Miller, S., Tribble, A., Whalen, M., Heimdahl, M., Proving the Shalls, International Journal on Software Tools for Technology Transfer (STTT), Feb 2006.
- Miller, S., Anderson, E., Wagner, L., Whalen, M., Heimdahl, M.: Formal Verification of Flight Critical Software. In AIAA Guidance, Navigation and Control Conference and Exhibit, AIAA-2005-6431, American Institute of Aeronautics and Astronautics (2005).
- Greve, D., Wilding, M., Vanfleet, W. M.: A Separation Kernel Formal Security Policy. In Fourth International Workshop on the ACL2 Prover and Its Applications (ACL2-2003) (2003).
- Greve, D., Richards, R., Wilding, M.: A Summary of Intrinsic Partitioning Verification. In Fifth International Workshop on the ACL2 Prover and Its Applications (ACL2-2004) (2004).
- Greve, D., Wilding, M., Richards, R., Vanfleet, W. M.: Formalizing Security Policies for Dynamic and Distributed Systems. In Systems and Software Technology Conference (SSTC 2005), Utah State University, (2005).