Formal Methods Activities on the AFRL Certification Techniques for Advanced Flight Critical Systems Challenge Problem Integration (CerTA FCS CPI) Program

a briefing at the

High Confidence Software and Systems Tools Workshop

20 May 2009
Linthicum Heights, Maryland

Cleared for Public Release - 88ABW-2009-2085

Outline

- Background
- Approach
- SpecSafe
- CodeHawk
- Conclusion

The Challenge

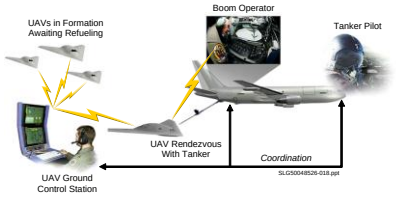
- DO-178B – the "gold standard" of SW verification
  - Based on defined processes and thorough testing
  - We can't "test our systems safe"!
- Systems growing exponentially in size and complexity
  - The test "space" is growing/exploding even faster
- The time has come for new approaches to assure system safety & security
  - "Software for Dependable Systems: Sufficient Evidence"
  - Also Internationally recognized

CerTA FCS Program

Boeing Research & Technology

Certification Techniques for Advanced Flight Critical Systems

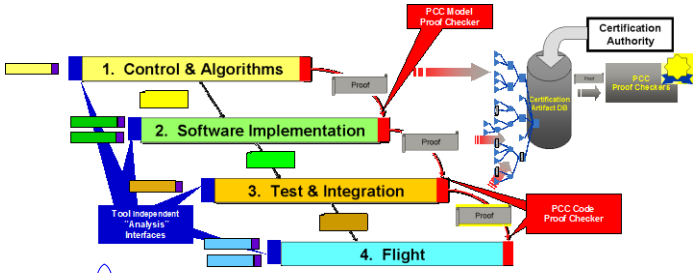
Addressing certifiability across the lifecycle



Multiple technology tasks, worked in parallel with a common vision

Gap Analysis and Technology Integration

Identify remaining certification challenges across the lifecycle
Identify existing and emerging technologies to close the gaps

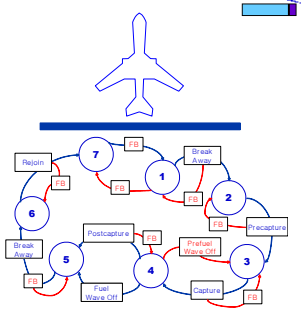


Verifiable Protocols for Decision Authority

Berkeley
UNIVERSITY OF CALIFORNIA

Develop certifiable protocols for decision authority to ensure safe and certifiable operation of UAVs

- Formal models of human-UAV interaction
- Formal specification of authority management
- Evaluate models to judge their effectiveness in demonstrating system safety

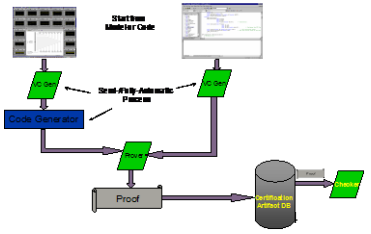


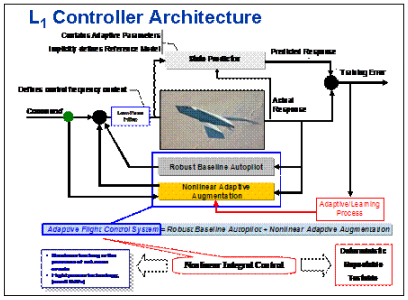
Proof-Carrying Code

Analysis techniques with easily checked evidence of results avoids tool qualification

Demonstrate using coverage analysis

Generalize to the complete development lifecycle





Stability / Robustness Margins for Adaptive Systems

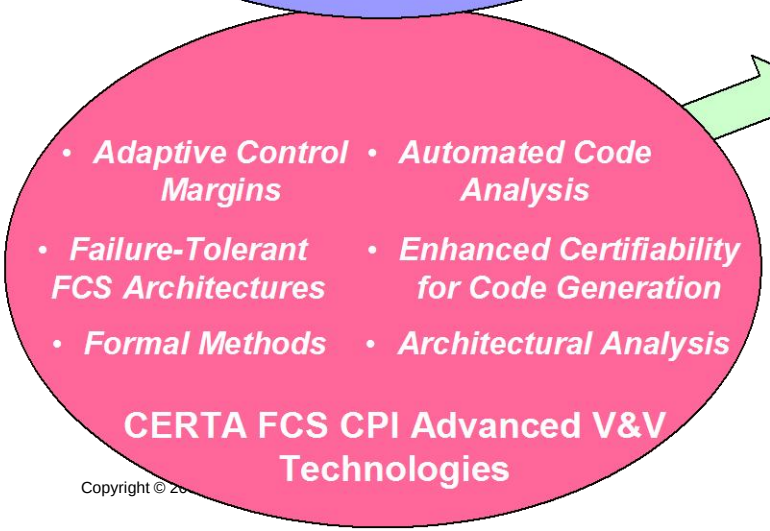
Identify stability / robustness margins for systems that contain adaptive elements

Develop theoretically justified and numerically efficient V&V methods applicable to adaptive flight controllers

Virginia Tech

CerTA FCS CPI Program Overview

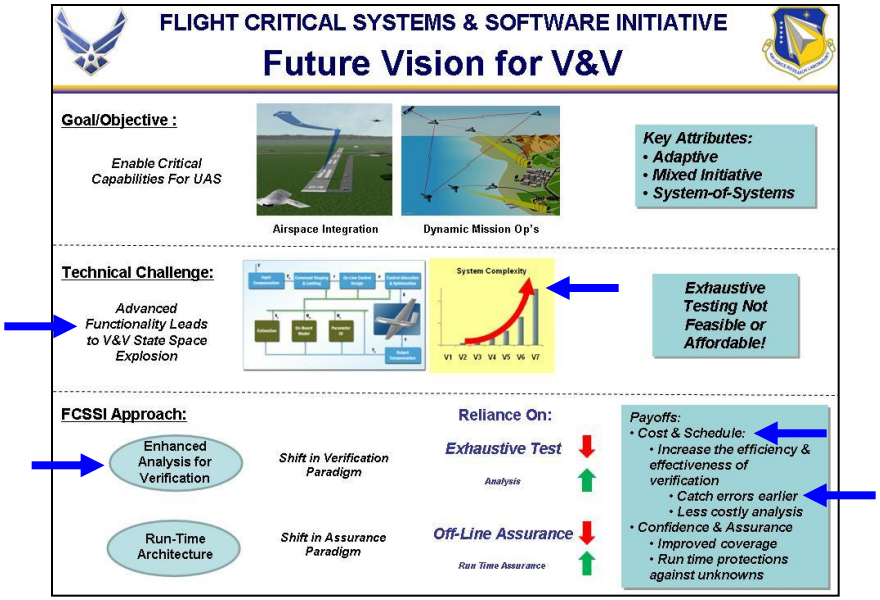
- **Apply advanced V&V technologies to a challenge problem to assess benefits of the technologies in a more realistic setting**
  - **Select a challenge problem**
  - **Select and integrate V&V technologies**
  - **Define Measures of Merit and Key Performance Parameters**
  - **Perform analytic assessment**

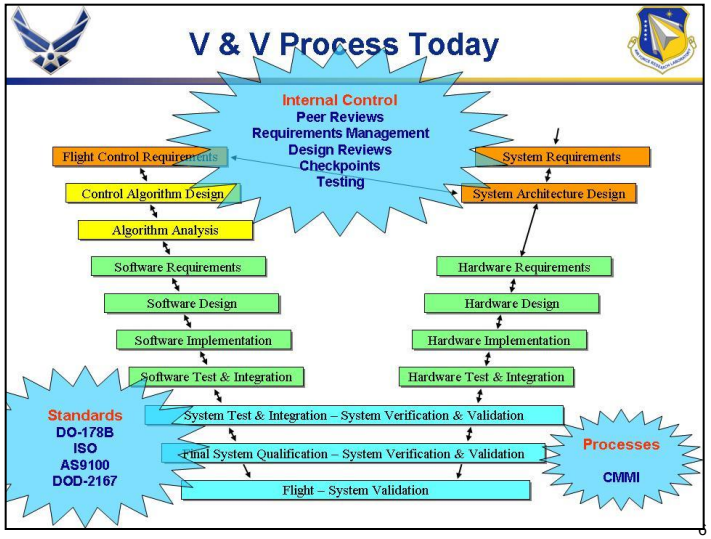


CERTA FCS CPI: Affordable V&V Technologies For Future Unmanned and Manned Aircraft and Weapons

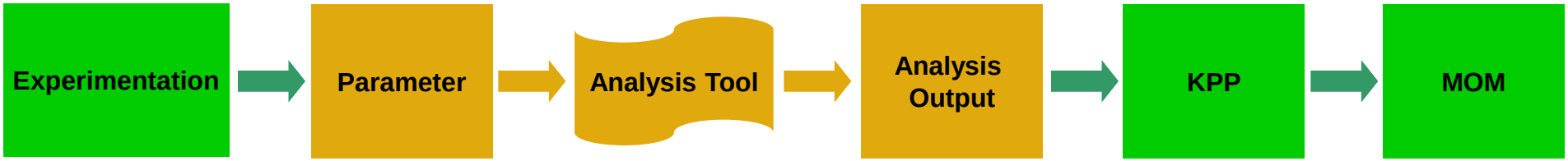
MOMs and KPPs

- **Measures of Merit (MOMs) represent the top level goals and objectives of a program, platform, or process improvement effort.**
  - **Time/Cost**
  - **Assurance**
  - **Expertise (to use/apply)**
  - **Enhanced Vehicle Capability (Increases in software complexity providing advanced functionality)**
- **Key Performance Parameters represent the benefits of individual features, aspects, steps or technologies**
  - **Benefit of investment in a development process step such as resulting in lower development cycle cost / lifecycle cost**





Analysis Approach

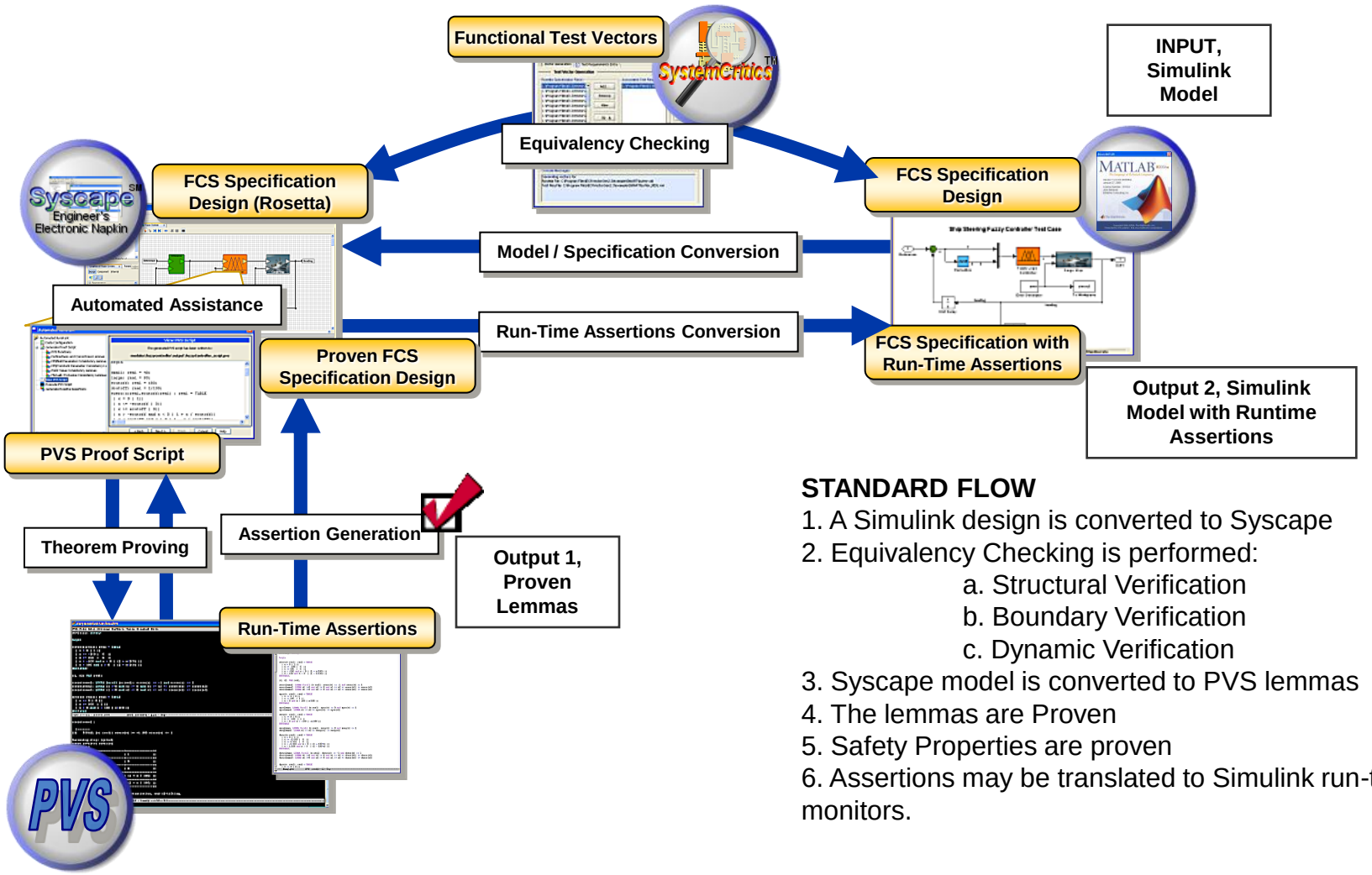


SpecSafe

- EDAptive Computing, Inc
- Theorem Proving applied to Control Models
  - Input: Simulink diagram
  - Output: Proven properties
- Approach
  - Translate to internal form (Rosetta/Syscape)
  - Automatically generate properties
  - Prove automatically (PVS)
- Additional capability: Generate run-time monitors

SpecSafe Tool Flow

Boeing Research & Technology



STANDARD FLOW

1. A Simulink design is converted to Syscape
2. Equivalency Checking is performed:
   a. Structural Verification
   b. Boundary Verification
   c. Dynamic Verification
3. Syscape model is converted to PVS lemmas
4. The lemmas are Proven
5. Safety Properties are proven
6. Assertions may be translated to Simulink run-time monitors.

Benefits to Embedded Flight Software

- **Formal Specification assures proven properties will hold over entire state space**
- **Reduces need for exhaustive testing**
- **Detection of potentially unsafe behavior during early design lifecycle processes provides cost saving benefits in later phases**
- **Reduces the need for Formal Methods experts through automated generation of lemmas**
- **Tested results in generation of stability properties**
  - **Example: Flight Control System command responsiveness (sensor inputs affect command outputs; command outputs respond correctly to changes).**
  - **Example: Range Checks (Proportional input / output changes; Directional similarities)**

Current tool and Planned Evolution

- **Assisted generation of complex safety properties (e.g., Phase Margin, Stability Margin, time-to-double/half amplitude)**
- **Support for more Simulink components**
- **Rosetta engine replacement for faster execution**
- **User Interface/Usability enhancements**
- **Integration of Model Checking capabilities**

MOMs and EDAptive SpecSafe

- **Time/Cost – without theoretically proving properties of the controller, additional human-intensive simulation-based validation of the model would be required, and additional errors may escape the controller design phase leading to rework**
- **Assurance – formal methods techniques provide more state space coverage than current methods**
- **Expertise (to use/apply) – want to bring this to the level of the normal Controls Engineer**
  - **CerTA FCS CPI is helping us explore this dimension**

KPPs and EDAptive SpecSafe

KPP	Measurands / Estimated Capabilities Informed by Experimentation
V&V Cost / Schedule Reduction through Earlier Defect Detection and Reduced Rework Costs	• Defect Detection Rates • Defect Escape Rates
Expertise (to use/apply)	• V&V with SpecSafe • V&V without SpecSafe

SpecSafe in CPI

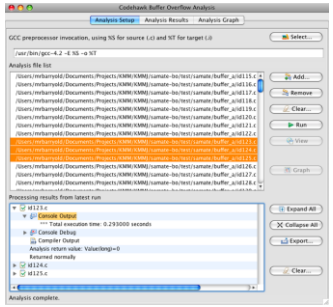
- Experimentally applying tool to challenge problem(s)
- Experimentation generated feedback which led to development of the following features:
  - Ability to enhance automatically-generated proof scripts with user-defined lemmas.
  - Ability for the tool to automatically generate components which have not been directly modeled in Syscape during the translation.
  - GUI Enhancements easing the use of the tool.
  - Enhancements in translation speed.
- Experiments have shown a need to support a wider range of lemmas, particularly at the system level.
- Experiments and data collection to feed final analysis still in progress

CodeHawk

- **Kestrel Technology**
- **Abstract Interpretation applied to source code**
  - **Input: C source code**
  - **Output: Errors, Warnings, Safe indications of each location in the source code relevant to the property**
- **Approach**
  - **Parse code**
  - **Produce conservative overapproximation of the reachable state space**
  - **Check property against state space**
- **CodeHawk is actually a checker generator**
  - **Define property**
  - **Generate checker**
  - **Automatically check property**

CodeHawk static analyzer

Boeing Research & Technology



C program

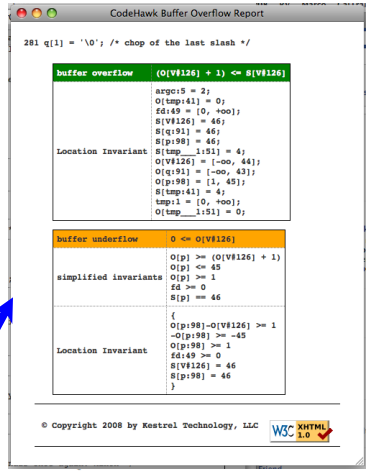
source text

no annotations
no modeling
no abstraction

CodeHawk analyzer

sound - static
semantic analyzer
for a given property

fully automatic




```
??
✓✓   278        do {
     279          --q;
     280        } while (q > resolved && *q == '/');
?  ✓ 281        q[1] = '\0'; /* chop of the last slash */
     282        q = resolved; /* q = /home/misha */
     283        }
     284
     285        printf("now resolved = %s\n", resolved);
✓✓   286        errno = 0;
     287
     288        resultcode = chdir(q); /* cd to /home/misha */
     289
     290        printf("result of chdir(%s) = %d\n", q, resultcode);
✓✓✓✓ 291        if (EACCES == errno) {
     292        uid_t userid = geteuid();
```

Underlying mathematical theory: **abstract interpretation**
(developed in France in 1970's)

Abstract Interpretation

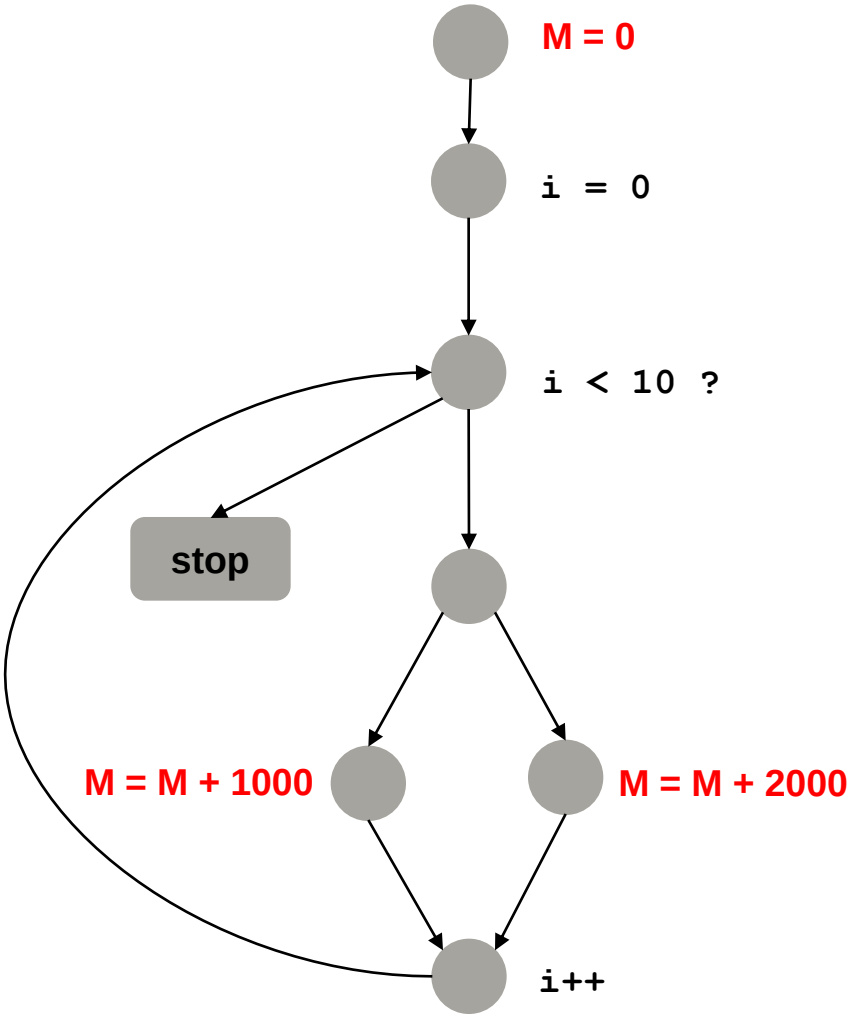
Conservative approximation of **all** program behaviors

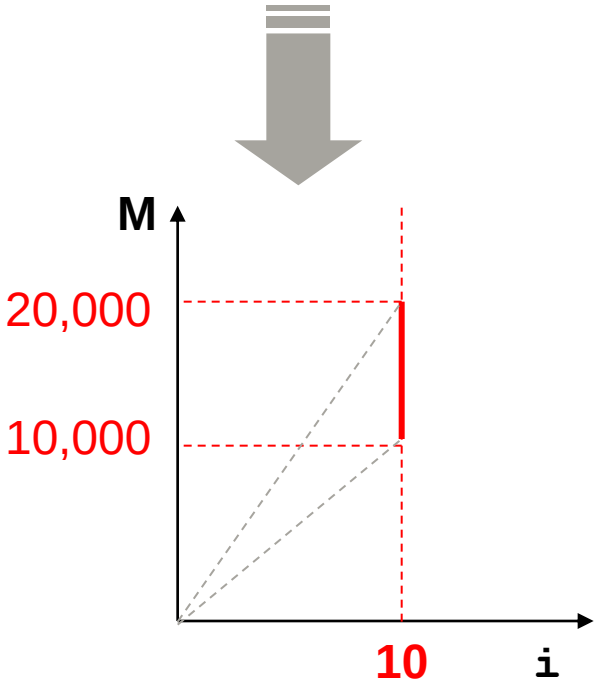
in a **domain** representation
in which property checking is **decidable**

Mathematical theory of abstract interpretation guides the design of domains and justifies the abstract semantics of program statements to guarantee that the approximation is conservative.

Challenge: scalability

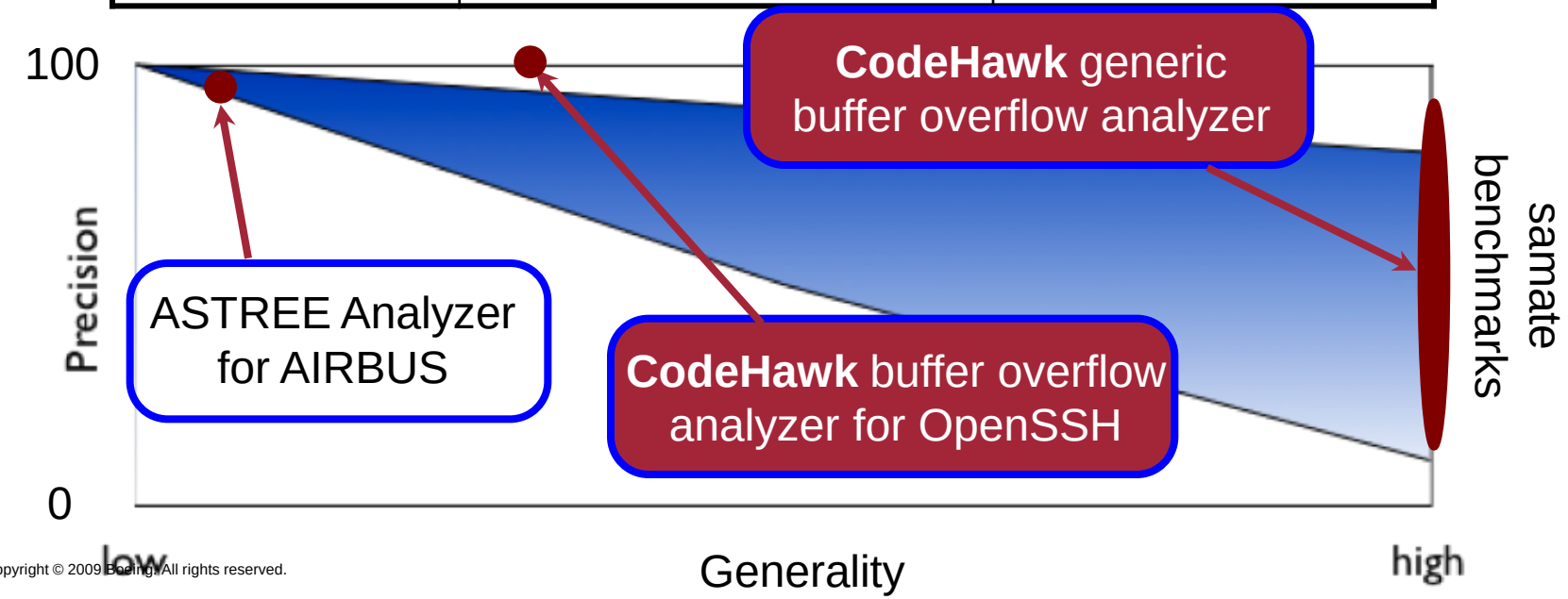
Illustration of Abstract Interpretation





CodeHawk Static Analyzer in Context

characteristic	Bug-finding tool	CodeHawk
soundness		✓
scalability	✓	(✓)
precision		**trade-off**
generality	✓	**trade-off**



Evaluating CodeHawk Analysis Output

Precision: (for a sound tool)

$$1 - \frac{\#\text{ false positives}}{\#\text{ warnings} + \#\text{ proven errors}}$$

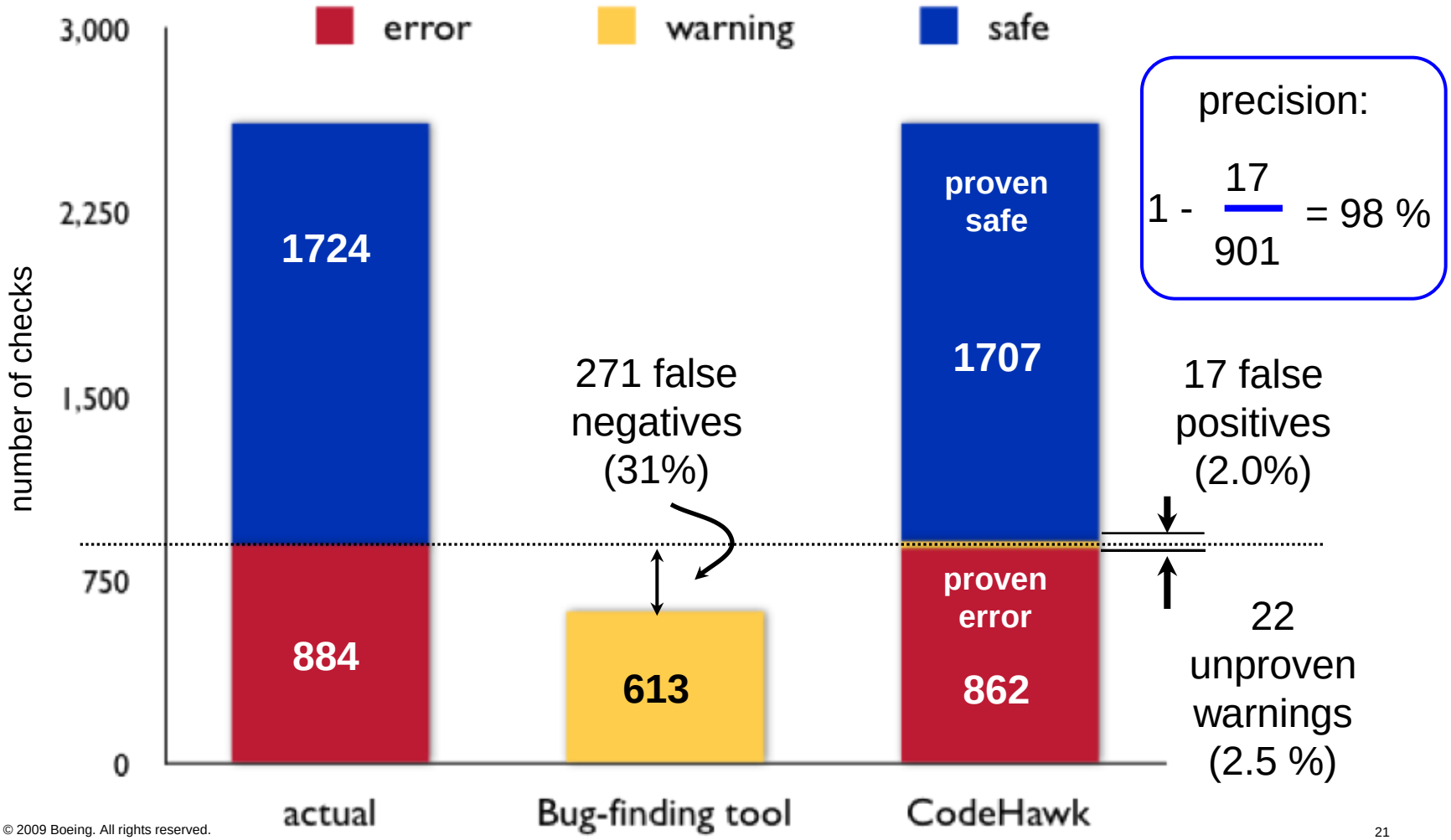
warnings: buffer accesses that cannot be proven safe and that cannot be proven to be an error

false positives: warnings that are, in fact, safe

SAMATE Benchmark Results

Boeing Research & Technology

Comparison with major bug-finding tool on NIST SAMATE benchmarks



Benefit: analysis results can be independently verified

		8	9		3			
4				2				7
	3	6				2		
	6	3			8	9	4	
8								5
	9	7	1			6	2	
		1				7	6	
6				4				3
			6		2	8		

Solving is hard →

7	2	8	9	1	3	4	5	6
4	5	9	8	2	6	1	3	7
1	3	6	4	5	7	2	8	9
2	6	3	5	7	8	9	4	1
8	1	4	2	6	9	3	7	5
5	9	7	1	3	4	6	2	8
9	4	1	3	8	5	7	6	2
6	8	2	7	4	1	5	9	3
3	7	5	6	9	2	8	1	4

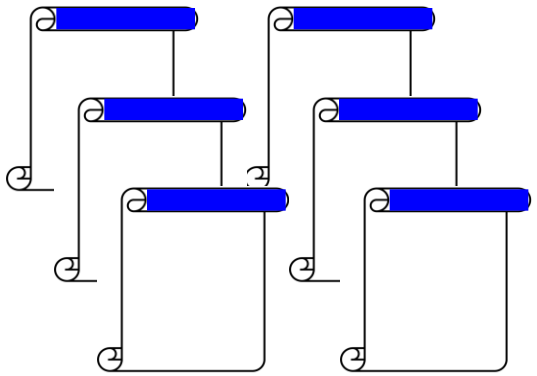
Checking that the solution is correct is easy

C Program source text

Generating proofs of safety and errors is hard →

requires sophisticated tool

CodeHawk



certification artifacts

Checking that the proofs are correct is easy

can be performed independently by certifying agency

Near-term planned enhancements

- **Redesign of the abstract interpretation to increase scalability**
- **Backward analysis to determine safe input constraints**
- **Desktop analyzer for exploratory and differential analysis**
- **Integration in Eclipse platform**
- **Support for other programming languages**

MOMs and Kestrel Technology CodeHawk

- **Time/Cost – without theoretically proving properties of the source code, additional testing and peer review would be required, and additional errors may escape the software coding phase leading to rework**
- **Assurance – formal methods techniques provide more state space coverage than current methods**
- **Expertise (to use/apply) – want to bring this to the level of the normal Software Engineer**
  - **CerTA FCS CPI is helping us explore this dimension**

KPPs and Kestrel Technology CodeHawk

Boeing Research & Technology

KPP	Measurands / Estimated Capabilities Informed by Experimentation
V&V Cost / Schedule Reduction through Earlier Defect Detection and Reduced Rework Costs	• Defect Detection Rates • Defect Escape Rates • Warning Rates
Expertise (to use/apply)	• V&V with CodeHawk • V&V without CodeHawk

CodeHawk in CPI

- **Experimentally applying tool to challenge problem(s)**
- **Experimental results**
  - **Detected buffer overflow error in controller code**
  - **Generated analyzers are easy to use**
  - **Interpreting inconclusive results is difficult**
  - **Application-directed customization can actually be done fairly easily (by experts) and can make a significant difference in the precision of the analysis**
  - **Generating new analyzers requires direct participation by Kestrel**
- **Experiments and data collection to feed final analysis still in progress**

Summary

- All the tools/technologies we are working with are showing promise of benefits to the development process for flight critical software subject to airworthiness certification
- SpecSafe and CodeHawk are also both meeting the goal of formal methods tools that “normal engineers” can use without
  - Extensive formal methods training
  - Creating tool specific representations
- In the process of incorporating our experimentation results into impact analysis results
- Areas for further maturation of both SpecSafe and CodeHawk have been identified that would further enhance their value