TC3 Overview

Guillaume Brat and Terry Morris

Welcome!

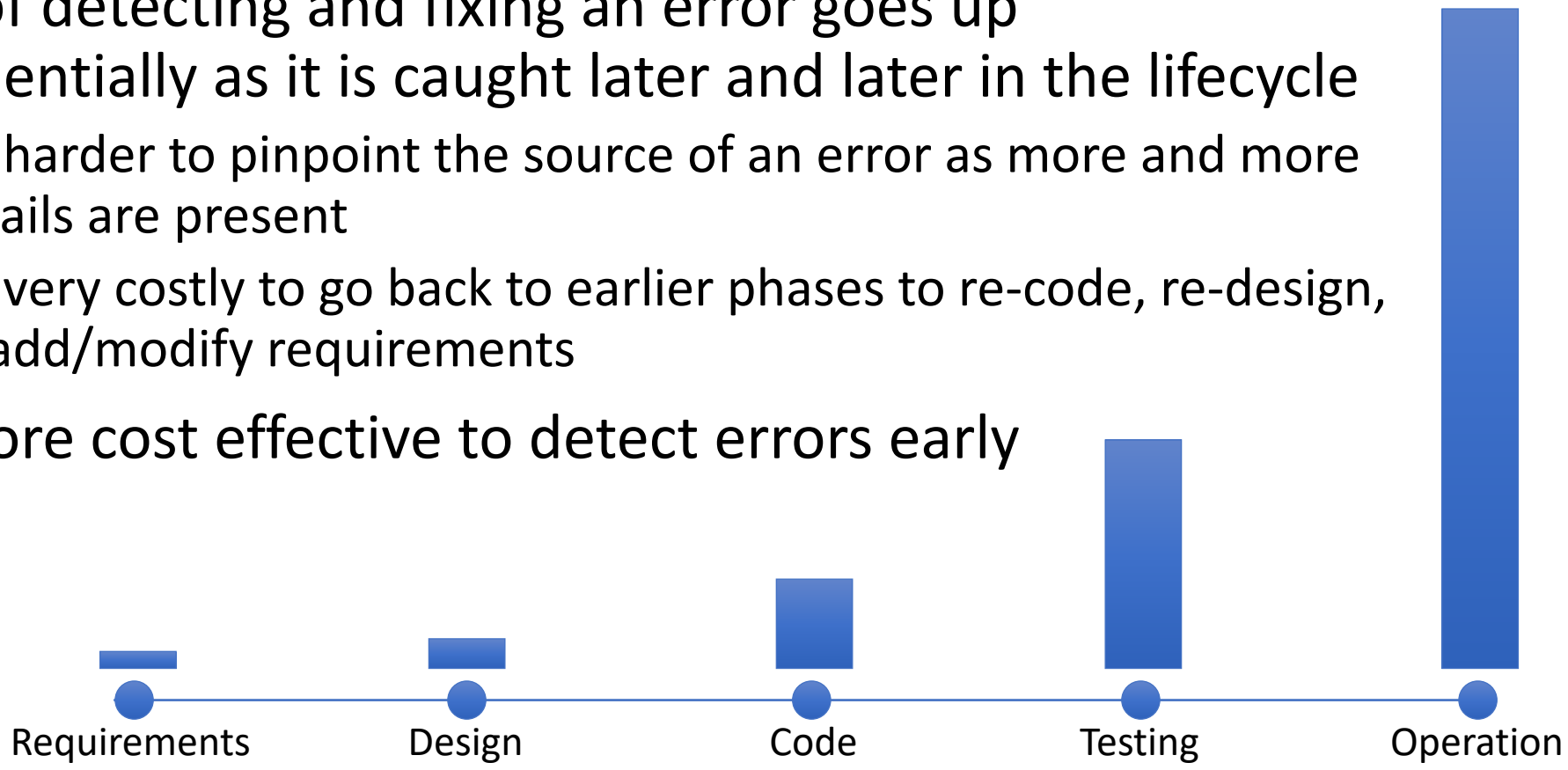
- This talk is summarizing the research done
  - under the Technical Challenge 3
  - in the System-Wide Safety project
  - in the Airspace Operations & Safety program
  - in the Aeronautics Research Mission Directorate

Motivation

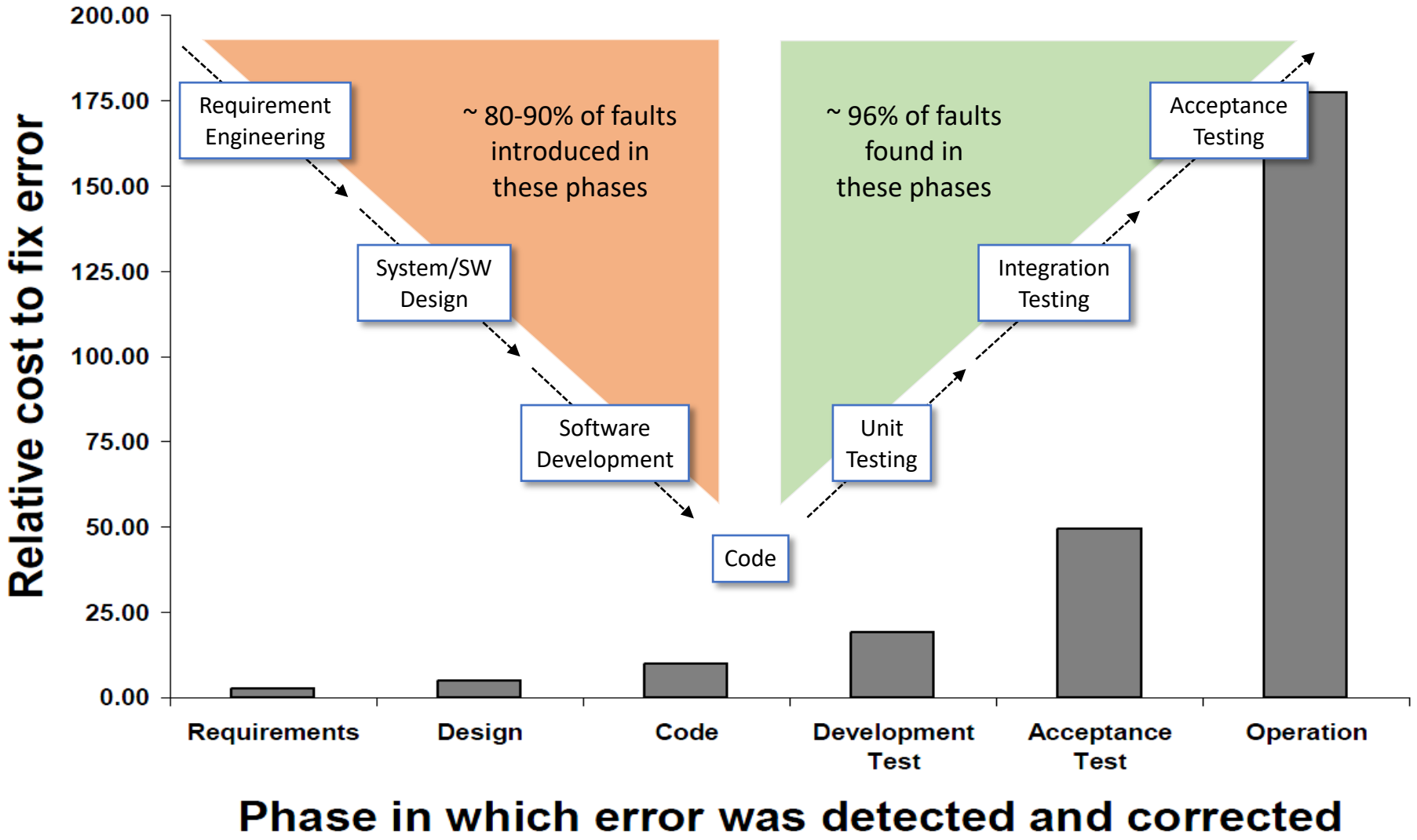
- What drive the V&V research in the first place?
  - Industry saying that the cost of V&V for certification is prohibitive
- Where is that cost coming from?
- How did this drive the NASA research strategy?

Cost of finding and fixing errors

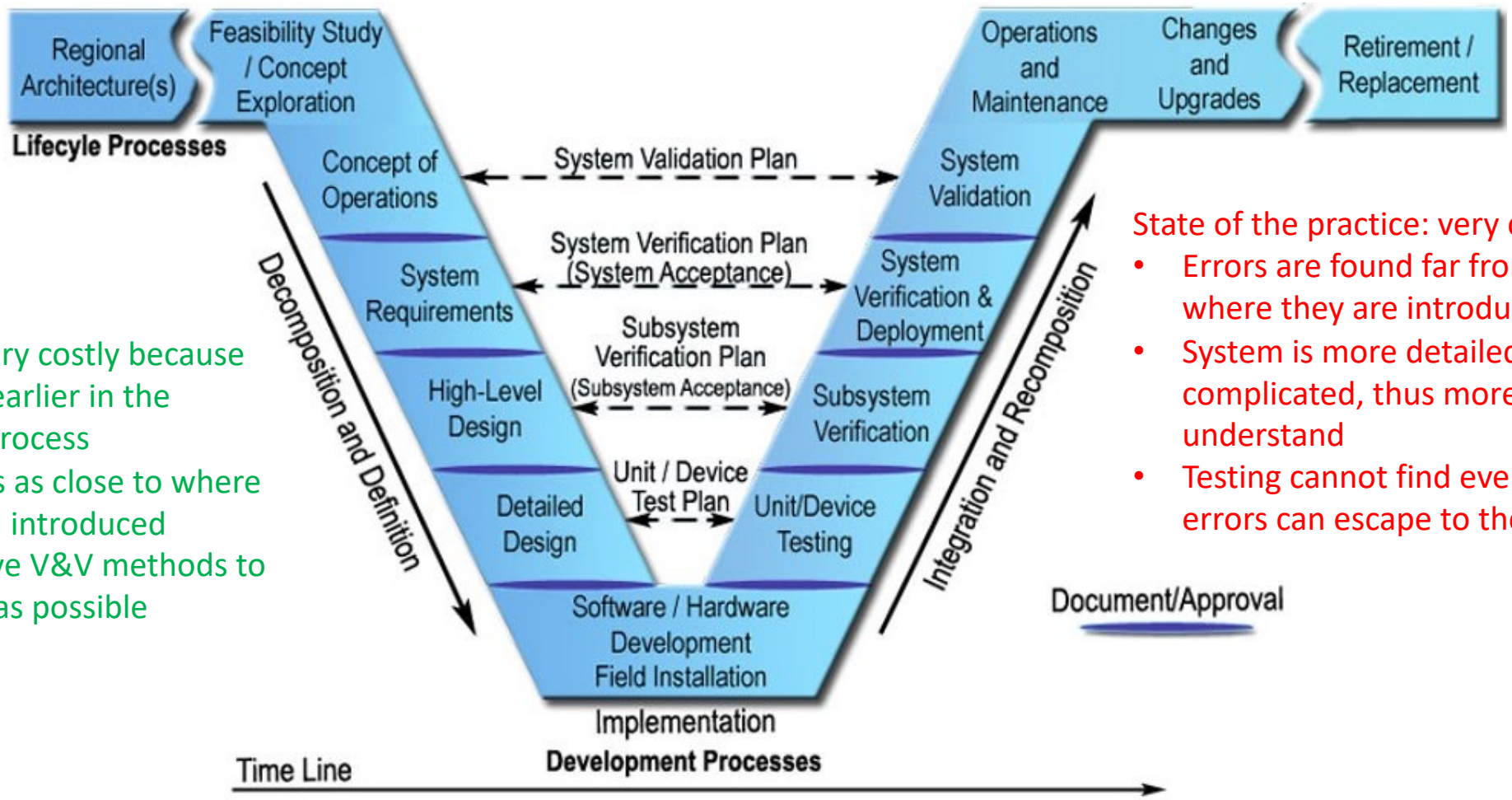
- Cost of detecting and fixing an error goes up exponentially as it is caught later and later in the lifecycle
  - it's harder to pinpoint the source of an error as more and more details are present
  - it's very costly to go back to earlier phases to re-code, re-design, or add/modify requirements
- It's more cost effective to detect errors early



Cost analysis



Motivation summary



TC3 philosophy: very costly because

- Push the V&V earlier in the development process
- Catch problems as close to where they have been introduced
- Favor exhaustive V&V methods to catch as much as possible

State of the practice: very costly because

- Errors are found far from the place where they are introduced
- System is more detailed, thus more complicated, thus more difficult to understand
- Testing cannot find every thing and errors can escape to the field

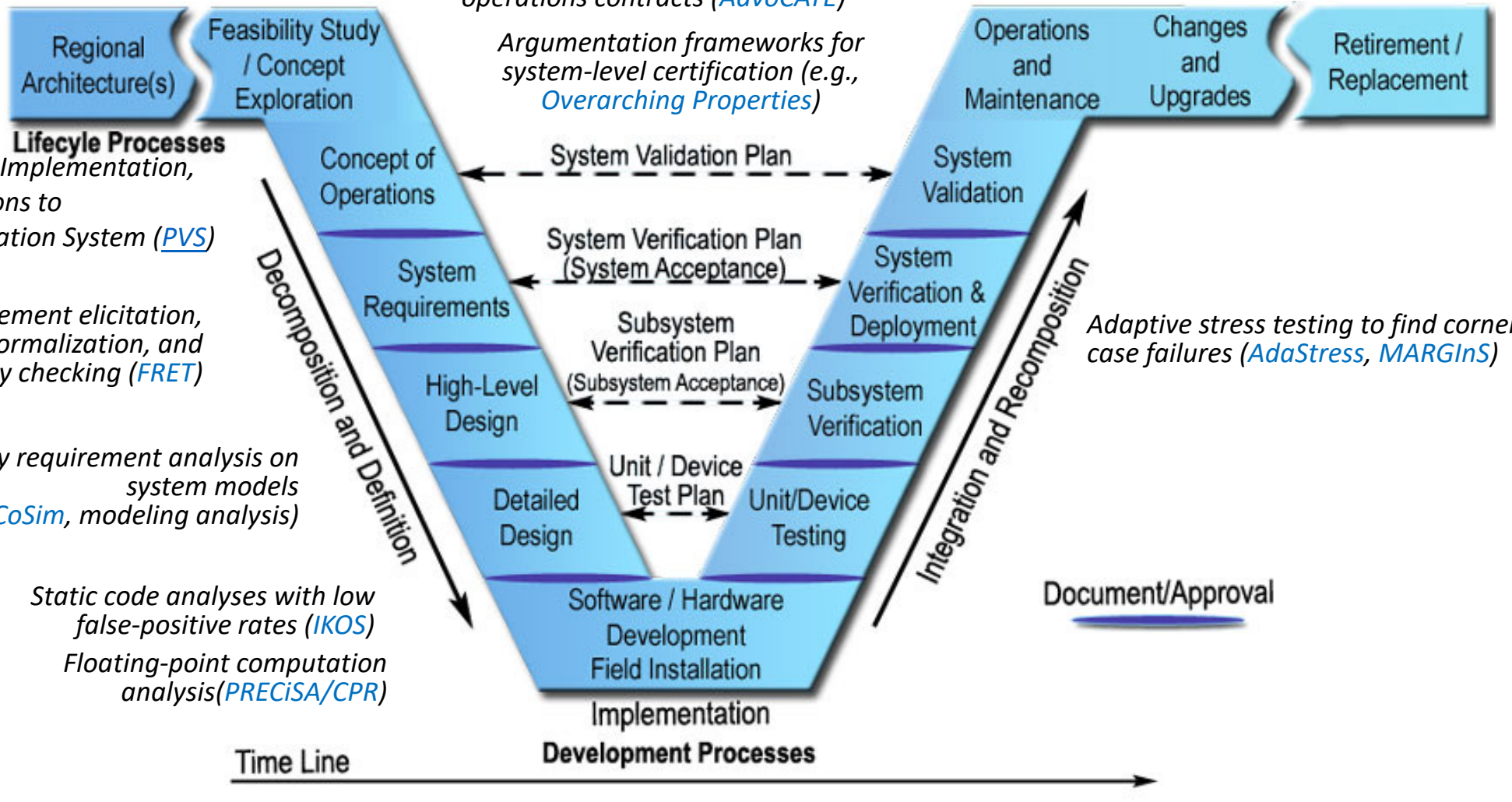
Overall strategy

- Formal methods and other advanced V&V techniques
  - Model complex systems as rigorous, mathematical entities
  - Make it possible to verify a system's properties in a more thorough fashion than empirical testing
- Focus on methods that reduce cost and reliance on testing
  - Applied earlier in the process on design models,
  - Avoid the high cost of detecting and fixing errors caught at testing
- Focus on methods that address complexity
  - Methods that work on models of the system where details are abstracted - complexity is reduced
  - Scalability challenges can be addressed such as with compositional V&V techniques
- Facilitate change at FAA to accommodate these techniques
  - Shift emphasis from testing for verification to testing for validation (of models used to verify system properties)
  - Case studies and other guidance for Industry and FAA
  - Industry has stepped up efforts to develop methods now that they see a path forward with the FAA

Research results

- What tools, techniques and processes were created?
- Have they been used outside NASA?
- What impact did they have?
- What is happening to them now?

Overview



Automated creation of and visualization of safety cases that carry over into operations contracts (AdvoCATE)

Operational run-time analysis (RACE, R2U2, Copilot, MESA)

Argumentation frameworks for system-level certification (e.g., Overarching Properties)

Requirements to Implementation, NASA Contributions to Prototype Verification System (PVS)

Requirement elicitation, formalization, and consistency checking (FRET)

Adaptive stress testing to find corner case failures (AdaStress, MARGInS)

Safety requirement analysis on system models (CoCoSim, modeling analysis)

Static code analyses with low false-positive rates (IKOS)

Floating-point computation analysis(PRECiSA/CPR)

Theorem Proving

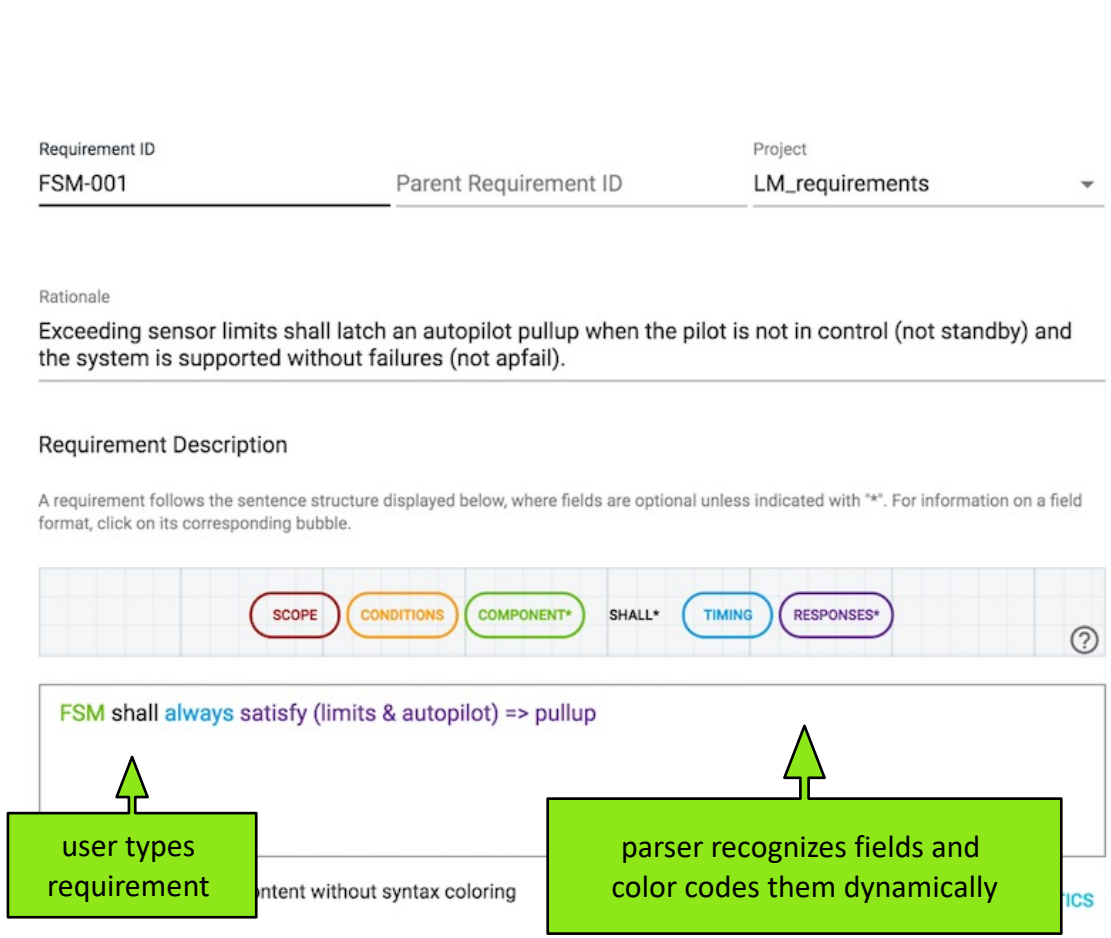
- Formalization of semi-decision and decision procedures for non-linear arithmetic in the Prototype Verification System (PVS):
  - Branch and Bound: Solve global optimization problems using sound numerical enclosures.
  - Sturm Theorem: Satisfiability and validity of quantified univariate polynomial inequalities.
  - Tarski Theorem: Satisfiability and validity of systems of univariate polynomials inequalities.
- Implementation of automated strategies for proving properties involving non-linear arithmetic, including square root and transcendental functions, and for performing rigorous computations:
  - Interval arithmetic.
  - Affine arithmetic.
  - Bernstein polynomial basis.
  - Exact arithmetic.
- Included in the publicly available NASA PVS Library:
  - http://github.com/nasa/pvslib

PVSioChecker: Validation By Model Animation

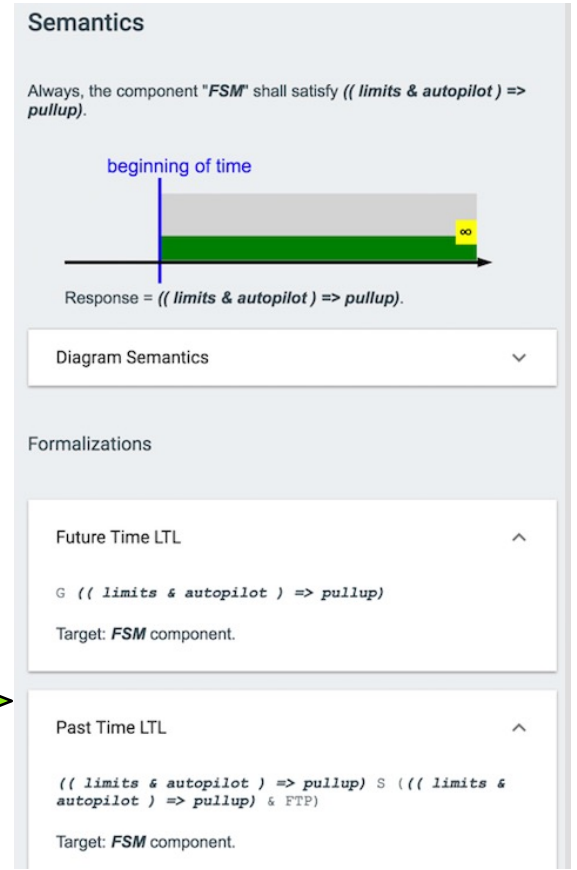
- PVSioChecker is a PVS tool for validating numerical software against their functional specification using the following approach:
  - Formally verified functional models of algorithms, which use real arithmetic, are evaluated on a set of test vectors.
  - Implementation of these algorithms, which use floating-point computations, are evaluated on the same set of test vectors.
  - Outputs are compared using a user-specified precision.
- Released as part of NASA PVS Library.

FRET

Create tool (FRET) to create requirements (as close to English as possible) and formalize them for analysis



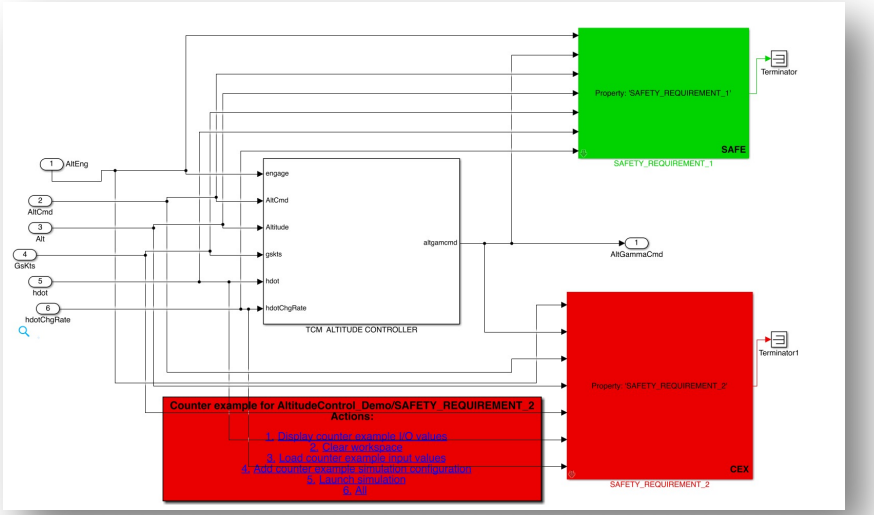
FRET generates formal semantics



CoCoSim

CoCoSim is applied directly to models; it can be used to verify safety requirements *before* the code is produced

- Integrated verification framework for Simulink models
- Fully automated analysis framework
- Shown effective in avionics applications



Limitations and future work

- Currently supports a subset of Simulink models described here https://github.com/coco-team/cocoSim
- Uses Z3 constraint solver – plan to integrate other solvers to extend the constraints that can be handled, for example trigonometric constraints
- Does support Stateflow but not embedded matlab

Kodiak: A Library For Solving Global Optimization Problems

- C++ Implementation of a formally verified generic branch and bound algorithm.
- Features:
  - Enclosure methods: Bernstein polynomial basis and interval arithmetic.
  - Parametric branching and bounding heuristics.
  - Support for symbolic computations.
- Tools:
  - Min/Max solver of non-linear functions.
  - Paving algorithm for computing under and over approximations of regions defined by system of non-linear inequalities.
  - Bifurcation analysis of systems of partial differential equations.
  - Boolean solver.
- Released under NASA's Open Source Agreement:
  - http://github.com/nasa/kodiak

IKOS

IKOS performs compile-time static analysis of software code and looks for errors that can happen at run-time

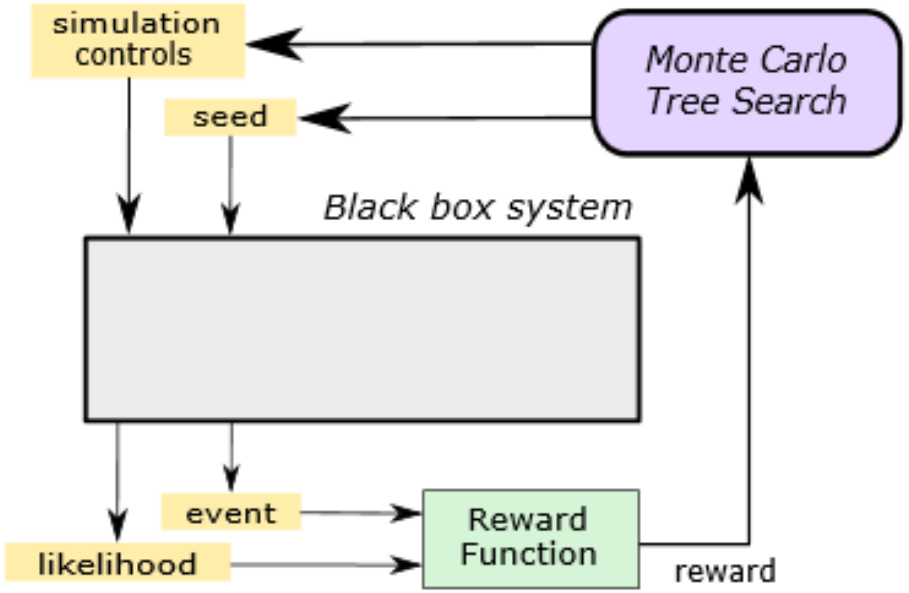
- Analyzes C code and C++ code
  - using the LLVM frontend
  - limited support for C++
- The code does not need to execute:
  - Libraries are not required, but knowledge on the environment and libraries is needed on a per case basis
- Available analyses:
  - Buffer overflow errors, Division by Zero, null pointer de-references, use of uninitialized variables
- No size restriction, but so far applied to 500 KLOC max
- Available open source at

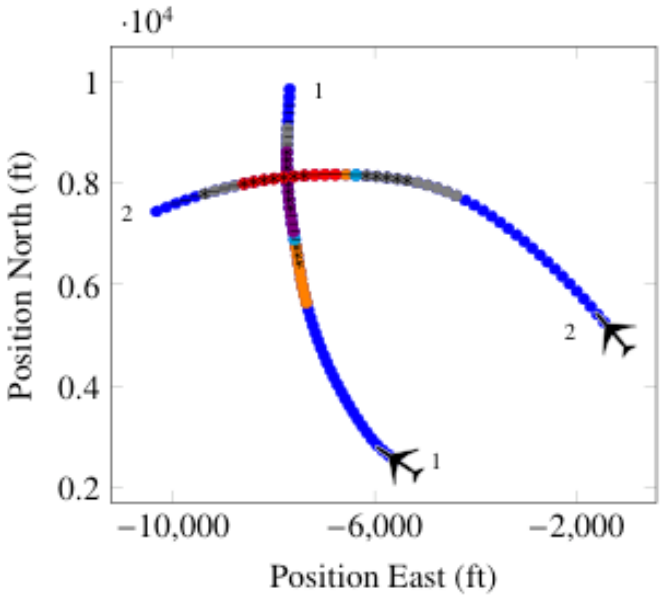
PRECIsA: Floating-Point V&V

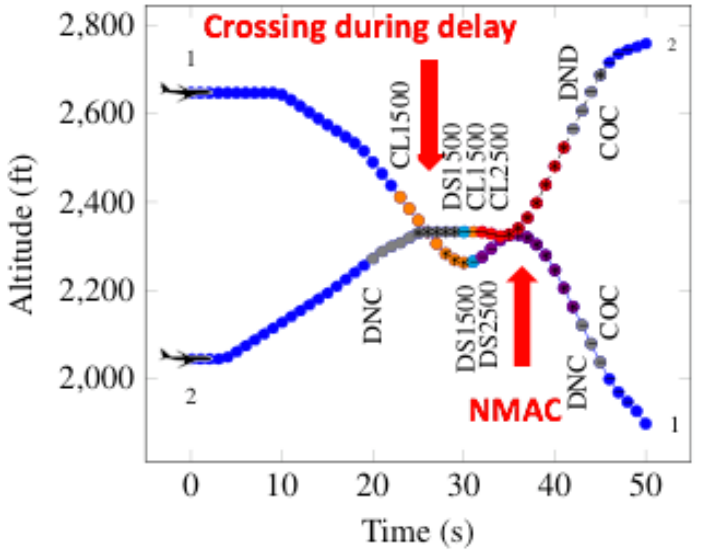
- Development of static analysis tool for floating-point error analysis:
  - Input:
    - A set of functions involving floating-point arithmetic.
    - Range of input variables.
  - Outputs:
    - Bounds on round off errors of floating-point computations.
    - PVS proofs that those bounds are correct.
- Integration with Frama-C (under development)

AdaStress: ML for stress testing

- AdaStress is a software package for an accelerated simulation-based stress testing method for finding the most likely path to a failure event







Integrated by GE in their Continuous Testing framework for aviation products

MARGInS

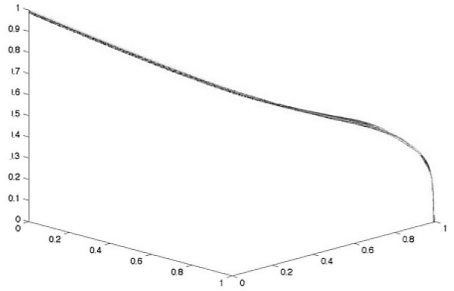
MARGInS is a library of machine learning and statistical tools for system testing.
MARGInS creates visualizations for aiding analyst understanding.

- Is an advanced test generation and test analysis toolset
- Implemented in MATLAB, C, and R – also runs in Octave
- Used for large and small scale space applications (Pad Abort, EFT-1, LADEE)

Pad Abort example

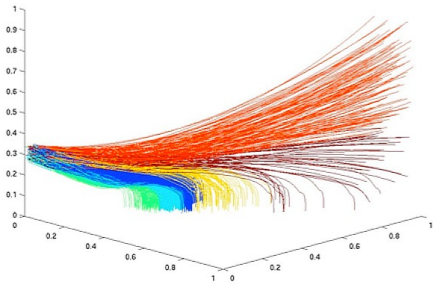
Limitations and future work

- Like any testing tool, MARGInS can increase confidence in the system but cannot guarantee the absence of errors
- Many of the results are visual, and require interpretation from domain experts
- Framework can be extended with additional components such as different machine learning techniques or test case generation schemes.



Standard testing
No automated analysis

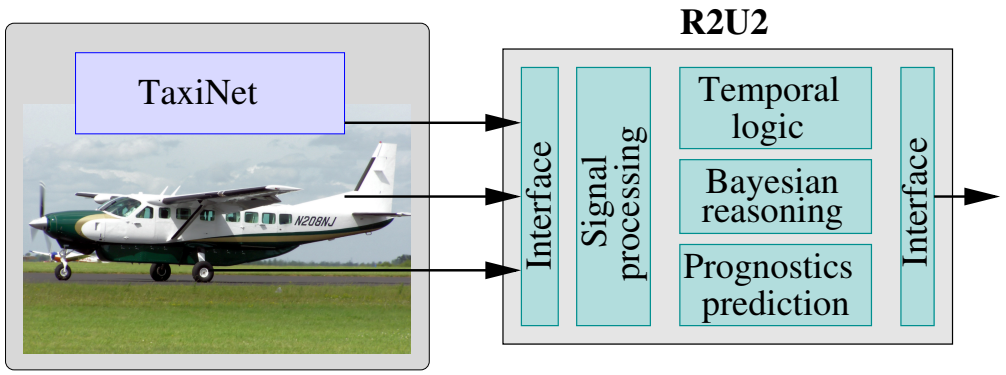
Vs.



Testing using MARGInS
Color indicates "risk classes"

R2U2: Run-Time monitoring

- R2U2 (Realizable, Responsive, Unobtrusive Unit), a hardware-supported tool and framework for the continuous monitoring of safety-critical and embedded cyber-physical systems.
- The R2U2 monitoring engine can be instantiated as a hardware solution, running on an FPGA, or as a software component.



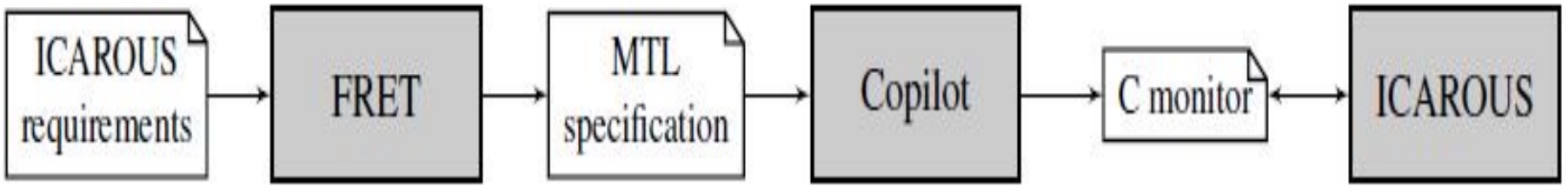
R2U2 is a run-time monitoring and V&V tool that combines *Metric Temporal Logic* observers, *Bayesian Network* reasoners, and *model-based prognostics*.

Co-Pilot

Goal: Develop run-time verification capabilities:

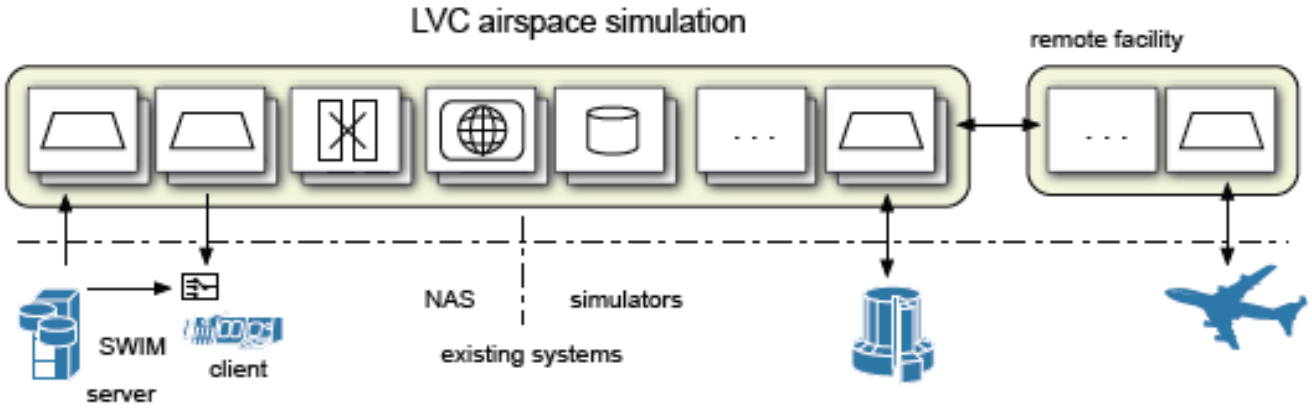
- ✓ Ultra-critical systems require high-level assurance
- ✓ They cannot always be verified during compile time
- ✓ Runtime verification enables monitoring of these systems during runtime
- ✓ RV Detects property violations early
- ✓ Limits the potential consequences of violations
- ✓ Facilitates writing RV monitors in high-level languages

- Co-Pilot 3.1 (Open Source)
  - ✓ Used in DARPA Assured Autonomy
  - ✓ Performs the translation to C
  - ✓ Stream-based runtime verification language and framework capable of generating portable, hard-real time C99 code
  - ✓ Suitable for embedded systems and architectures
- X-Plane
  - ✓ Co-Pilot has versatility to monitor flight sims in X-Plane



RACE

- started as a distributed LVC simulation framework in 2015

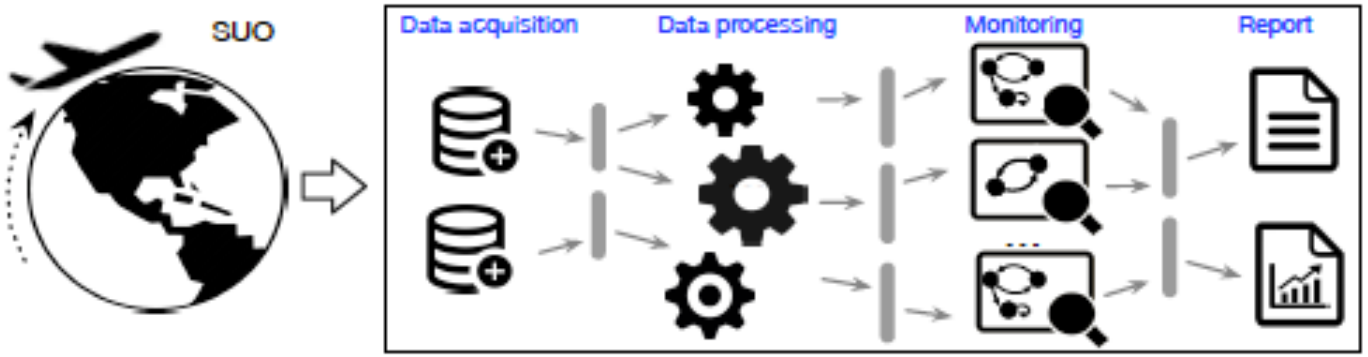


- evolved into a general event driven application framework:
  - can import/export from/to external systems - **connectivity**
  - can process high event rate and data volume - **scalability**
  - supports distributed and massively concurrent operation
  - has batteries included (except Java runtime, SBT build system)

Application: automatic detection of
(1) temporal inconsistencies in SWIM messages
(2) unsafe parallel approach trajectories
(3) trajectory deviations between different reporting systems
(4) loss of separation between live and simulated aircraft

MESA

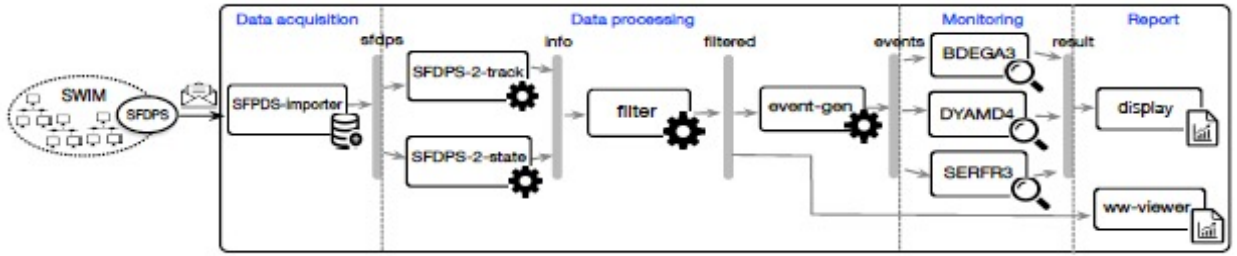
MESA is a framework for building actor-based monitoring systems.



MESA (MEssage-based System Analysis) enables using concurrent (runtime) monitors to check for properties specified in data parameterized temporal logic and state machines.

Application: Using MESA, we discovered violations at SFO of the following property:

PRSA : a flight shall cross inside a 1.0 NM radius around each waypoint in the assigned RNAV STAR route, in order.



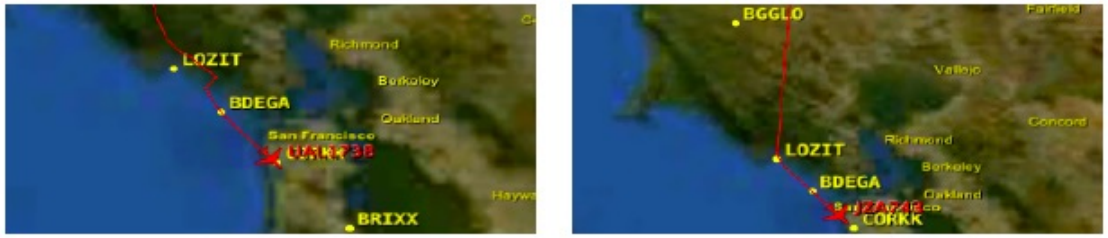
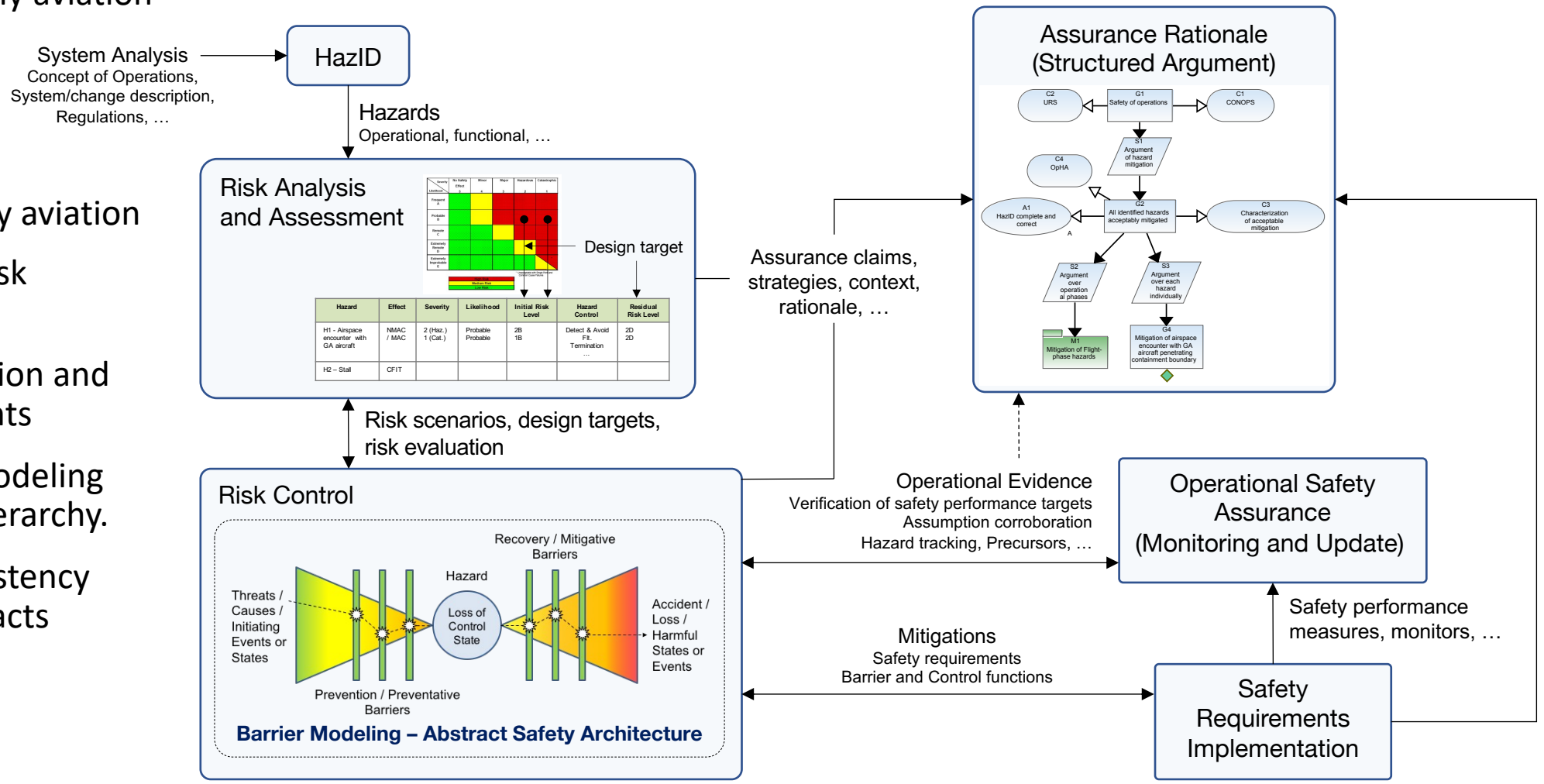


Fig. 6: Flight deviation from assigned RNAV STARs detected at SFO.

AdvoCATE

The Assurance Case Automation Toolset (AdvoCATE) supports the development and management of safety/assurance cases, providing novel capabilities in automating their production, with applicability to safety-critical applications in general, and especially aviation

- general, and especially aviation
- Hazard analysis and risk assessment:
- Capture of risk reduction and assurance requirements
- Safety architecture modeling using modules and hierarchy.
- Traceability and consistency between related artifacts
- Assurance analytics



Overarching Properties

Motivation and Objective

- Investigate the feasibility and efficacy of creating a viable path for certification / approval of airborne and ground-based systems based on the principles and methods of argumentation to show a system possesses the Overarching Properties (OPs)

- **Intent**: The *defined intended behavior* is correct and complete with respect to the *desired behavior*.
- **Correctness**: The *implementation* is correct with respect to its *defined intended behavior*, under *foreseeable operating conditions*.
- **Innocuity**: Any part of the *implementation* that is not required by the *defined intended behavior* has no *unacceptable impact*.

Case studies and Impact

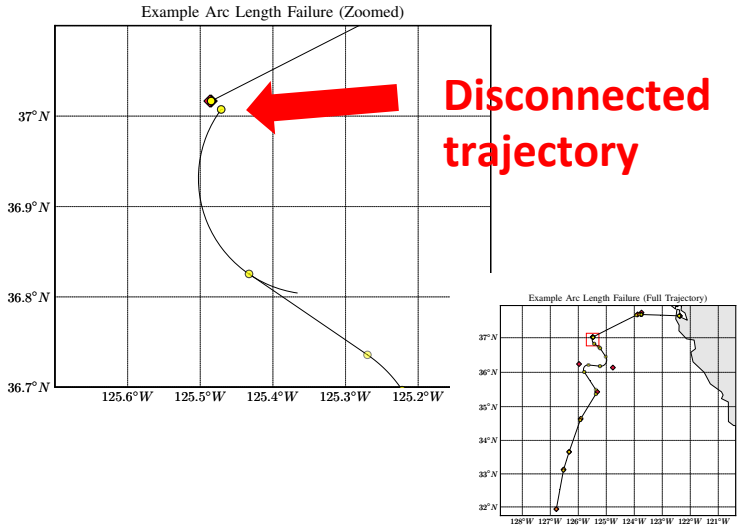
- Industry-led case studies
  - Collins Aerospace / Raytheon: formal methods, CoCoSim
  - GE: AdvoCATE, AdaStress
  - Boeing: many tools presented to large audience
- In-house case studies
  - CoPilot used in TC2
  - Troupe project integrated many tools in development process

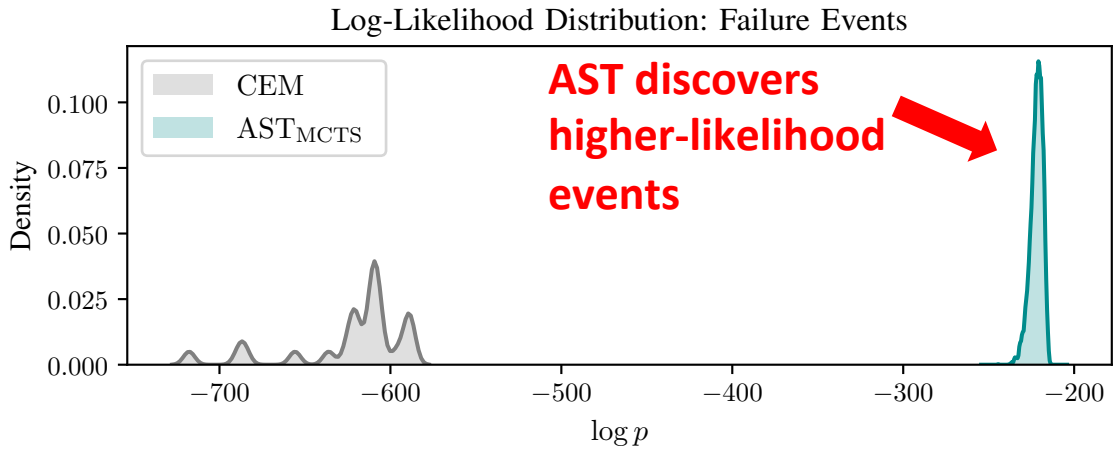
Collins / Raytheon Experience

- Use of formal methods to do fault analysis for an AutoThrottle example.
  - Approach shows that more faults could be found that way over traditional approaches.
  - The formal approach also proves less costly (time savings)
- Use of CoCoSim and Design verifier to prove formal properties on FADEC model in Simulink.
  - The experience proved that formal methods are useful and can find interesting problems.
  - It also showed that scalability and usability improvements are still needed.
  - Also provided feedback on FRET, the requirement formalization tool.

GE Experience

- Application of adaptive Stress Testing (AdaStress) to a flight management system
- GE decided to fold AdaStress into their continuous test framework for aviation products

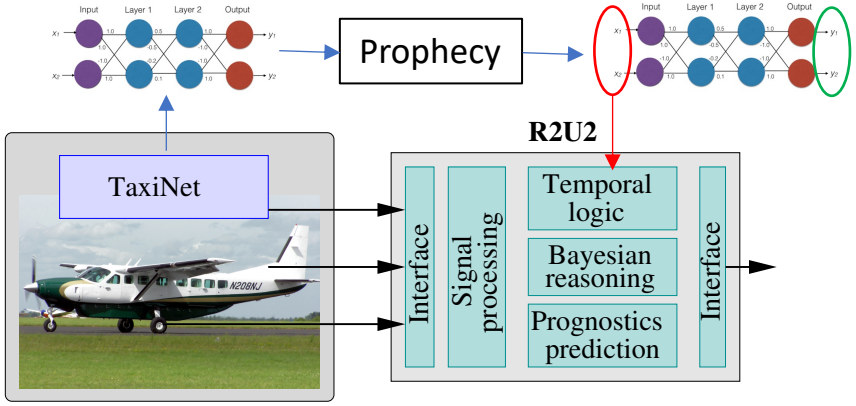




Boeing presentations

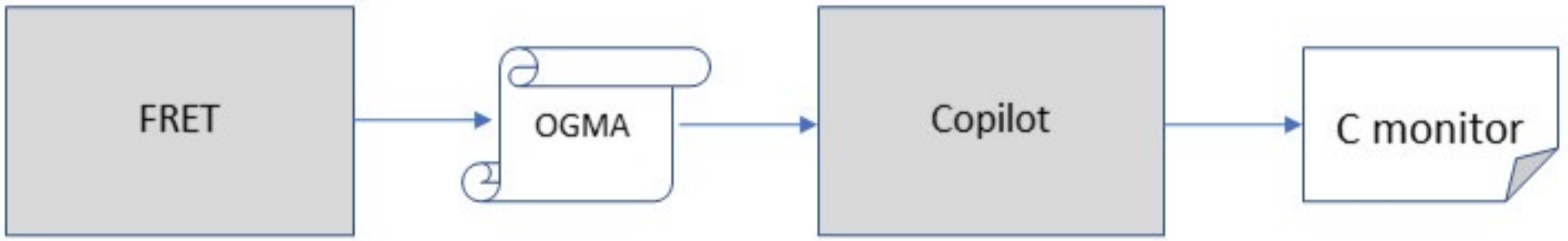
- Boeing: taking a hard look (with their evaluation) of many NASA-develop tools
  - Reaches across many Boeing orgs
  - Could revolutionize the way Boeing does V&V
- NASA gave a series of virtual talks to Boeing, which were very well attended (80+ attendees)
  - IKOS, CoCoSim, FRET, MARGInS, RACE, MESA
- This is resulted in an on-going deeper and tighter collaboration on assurance for autonomy under a Space Act Agreement.
- Example: Using Prophecy with R2U2

on Taxinet



CoPilot used in TC2

- The goal is to develop run-time verification capabilities:
  - Ultra-critical systems require high-level assurance
  - They cannot always be verified during compile time
  - Runtime verification enables monitoring during runtime
  - RV Detects property violations early
  - Limits the potential consequences of violations
  - Facilitates writing RV monitors in high-level languages
- An associated tool, OGMA, takes requirements capture files from FRET & Collins Lustre and transforms them to Copilot specs



Conclusions

- TC3 was a very successful project. It produced many tools and techniques for the assurance of critical aviation software systems
- We have covered all aspects of the V diagram
- Some of the tools have been used in industry and have shown promises to reduce assurance cost.
- These tools and techniques are still needed as we turn our focus towards autonomous systems
  - Many are being extended to deal with autonomy
  - New tools are also being developed