RSE/Ames intro

- **CAST: Core Avionics and Software Technologies**
  - cFS/cFE based FSW
  - Model-based development
  - Swarm simulation
  - Small spacecraft swarm
  - FSW Monitoring & Review

- **ASSET: Automated Software & System Engineering Technologies**
  - Formal methods
    - Requirement Elicitation/Analysis
    - Design Model Analysis
    - Static Code Analysis
    - Run-Time Analysis
  - Adaptive stress testing
  - Assurance Cases
  - Machine learning V&V

- **ARMD**
  - **AOSP: System-Wide Safety**
    - Aviation software assurance
    - Autonomous systems assurance
  - **AAM: Automated Flight and Contingency Management**
  - **Low Boom Flight Demonstrator**

- **Others**
  - **Gateway contractors V&V**
  - **BlueOrigin: ROS 2 V&V**
  - **DoD Autonomous Systems T&E**

- **STMD**
  - **LCROSS, LADEE**
  - **BioSentinel**
  - **Starling, CASAS, DSA**
  - **RP, VIPER**

**Emerging Markets**
- **Urban Air Mobility**
- **Plant inspection, disaster response, precision agriculture, USGS**

**Traditional Aviation**
- **US Airframers, OEMs, Airlines**
- **FAA, NTSB**
- **AFRL**
V&V for Traditional Systems

• Motivation for funding: faster and cheaper Aviation certification through V&V earlier in the lifecycle
  • Focus on flight software

• Applications in our own group working on flight software for small spacecraft
  • Using a model-based, agile process
  • Streamline cost

• Tried it out when “auditing” defense contractors
  • Reluctance to using new methods
V&V for certification

• Motivation for funding: faster and cheaper Aviation certification through V&V earlier in the lifecycle
  • Focus on flight software

  DO-178C

• Current standard for getting software certification credits
  • Traceability throughout
  • Track, record, and document

  DO-333

• Formal methods supplement:
  • Model checking
  • Abstract interpretation
  • ...

Motivation for funding: faster and cheaper Aviation certification through V&V earlier in the lifecycle

• Focus on flight software

• Current standard for getting software certification credits
  • Traceability throughout
  • Track, record, and document

• Formal methods supplement:
  • Model checking
  • Abstract interpretation
  • ...
V&V for certification

- Current standard for getting software certification credits
  - Traceability throughout
  - Track, record, and document

- Analysis available as a V&V means
  - Model checking
  - Abstract interpretation
  - ...

- Applications in our own group working on flight software for small spacecraft
  - Using a model-based, agile process
  - Streamline cost

NPR-7150.2c

NASA-STD-8379.8a
V&V Strategy

- Heavy use of formal methods to target early lifecycle stages
- Smarter testing techniques

**Cocosim**
Safety requirement verification on Simulink models

**FRET**
Requirement elicitation, formalization and consistency checking

**IKOS**
Static code analyzer for C/C++ with low false positive rate

**AdaStress**
Adaptive stress testing to find corner cases

**MARGInS**
Statistical testing to identify safe/unsafe modes of operation
FRET at a Glance

Requirement ID
FSM-001

Parent Requirement ID
LM_requirements

Project

Rationale
Exceeding sensor limits shall latch an autopilot pullup when the pilot is not in control (not standby) and the system is supported without failures (not a fail).

Requirement Description

A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

user types
requirement

FRET generates formal semantics

parser recognizes fields and color codes them dynamically

Semantics

Always, the component "FSM" shall satisfy ((limits & autopilot) => pullup).

beginning of time

Response = ((limits & autopilot) => pullup).

Diagram Semantics

Formalizations

Future Time LTL

G ((limits & autopilot) => pullup)

Target: FSM component.

Past Time LTL

((limits & autopilot) => pullup) & ((limits & autopilot) => pullup) & FTP

Target: FSM component.
Lessons Learned: FRET

- Going from natural English to formal logic is hard
  - FRET uses English patterns/modes
- Having enough patterns to cover most requirements possible is hard
  - We work with industry and NASA missions to see if FRET covers their needs

FRET
Requirement elicitation, formalization and consistency checking
CoCoSim at a glance

• 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
Lessons Learned: CoCoSim

- Formalizing properties is hard but we solved the problem by coupling with FRET.
- Covering all possible Simulink block is impossible
  - We are falling short of covering all blocks used in industry
- Still can have scalability issues in analysis
  - Same for Design Verifier, even if it’s for different models

Cocosim
Safety requirement verification on Simulink models
IKOS at a Glance

• 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
Lessons Learned: IKOS

- IKOS works/scales better for C than C++
- Possible to achieve less than 1% warnings
- Some users won’t believe the results
  - Need better explanations on why a bug is a bug
- Keeping up with language evolution is exhausting

IKOS
Static code analyzer for C/C++ with low false positive rate

<table>
<thead>
<tr>
<th>Module</th>
<th>Analysis Time</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS</td>
<td>37mn</td>
<td>97%</td>
</tr>
<tr>
<td>ACT</td>
<td>35mn</td>
<td>99%</td>
</tr>
<tr>
<td>BCS</td>
<td>108mn</td>
<td>97%</td>
</tr>
<tr>
<td>CMP</td>
<td>89mn</td>
<td>99%</td>
</tr>
<tr>
<td>HIO</td>
<td>7mn</td>
<td>98%</td>
</tr>
<tr>
<td>SMC</td>
<td>37mn</td>
<td>98%</td>
</tr>
<tr>
<td>TCS</td>
<td>103mn</td>
<td>99%</td>
</tr>
</tbody>
</table>
MARGInS at a Glance

• 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)

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.
Lessons Learned: MARGInS

- Not a fully automated process
  - User needs to tell what is a result of interest
- Useful to find boundaries for safe operation
  - One industry partner is using it to define ranges in requirements
- Speed depends heavily on simulation speed

MARGInS
Statistical testing to identify safe/unsafe modes of operation
AdaStress at a Glance

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

Technical POC: Ritchie Lee
Lessons Learned: AdaStress

- Has found very useful corner cases for
  - ACAS-X suite of software
  - Industrial flight management system
- Speed depends heavily on simulation speed
Move to autonomous systems

• Asked to address increasingly autonomous systems

• Different customers
  • Traditional: safety conscious but slow
  • New entrants: don’t know about safety and want to go fast
    • Strategy: deploy asap, update regularly, very quick cycles
    • Doesn’t fit with current certification processes
AI is a big world
New goals

• Work on V&V techniques for autonomy
  • Adapt existing tools for autonomy
  • Solidify use of runtime monitoring
  • Rely on adaptive stress and statistical testing
  • Research formal V&V for Machine Learning (ML)
  • Explore the notion of explainability

• Deliver draft for new standards for autonomy
  • Practical: focus on ML-Enabled Components, then move to systems
Driving case studies

• Centerline tracking system for autonomous surface operations
  • DNN for image analysis

• Suite of ACAS-X software
  • Multi-optimization-based software
  • NN implementation

• RAV rover
  • Centerline tracking, fusion of various sensor data

• Multi rover coordination
  • TBD
  • Lots of V&V tools will be experimented on this (internal) mission
Prophecy: Contract Inference For NN

• Key Idea:
  • Infer “likely” properties, aka contracts, of a NN
  • Prove them using a decision procedure

• What is a contract? $\sigma \Rightarrow P$
  • $\sigma$ is a precondition (“safe region”)
  • $P$ is a postcondition; desired output behavior (e.g. some prediction)
R2U2 is a run-time monitoring and V&V tool that combines Metric Temporal Logic observers, Bayesian Network reasoners, and model-based prognostics.
Conclusions

• Lots of experience building V&V tools for traditional aviation systems
  • Most are being evaluated by industry right now
  • Also worked with AFRL

• Need to address autonomy
  • Our primary focus is Machine Learning
  • We’ll go from a focus on specialized components to systems involving the use of ML in various places

• The end product is a draft for ML standards.
  • what V&V can be done will inform that