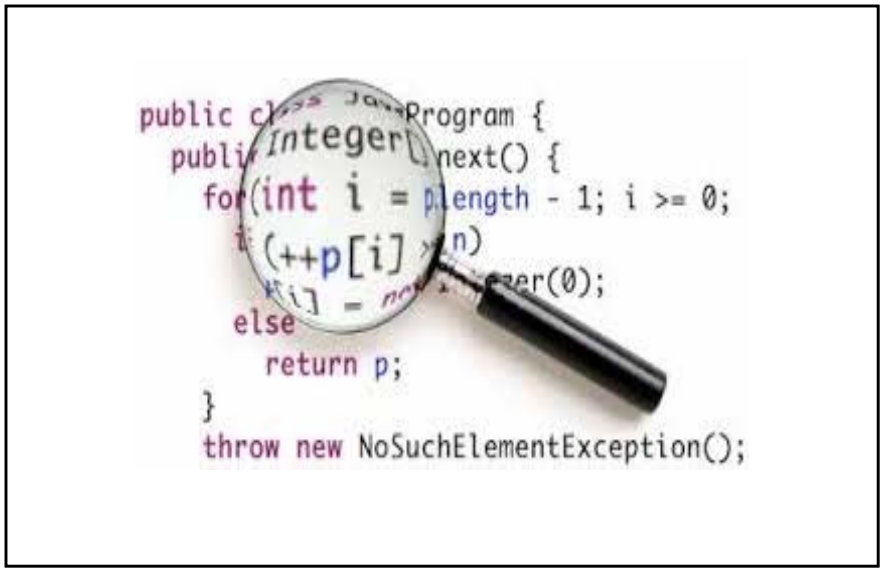
Software and GNSS Fault-Monitoring for Automated Aircraft



J.H. Rife, H. Huang, S.Z. Guyer

Tufts University

ION GNSS+ 2018

November 16, 2018

Tufts UNIVERSITY

Key Concepts

- Propose an online **bug-monitoring** approach
  - Use analogy to GNSS augmentation system

- Discuss potential impact for pre-service verification

Aviation Software at a Crossroads

Augustine's Law: Complexity of flight software grows an order of magnitude/decade

Will traditional verification approaches (e.g. RTCA DO-178) scale?

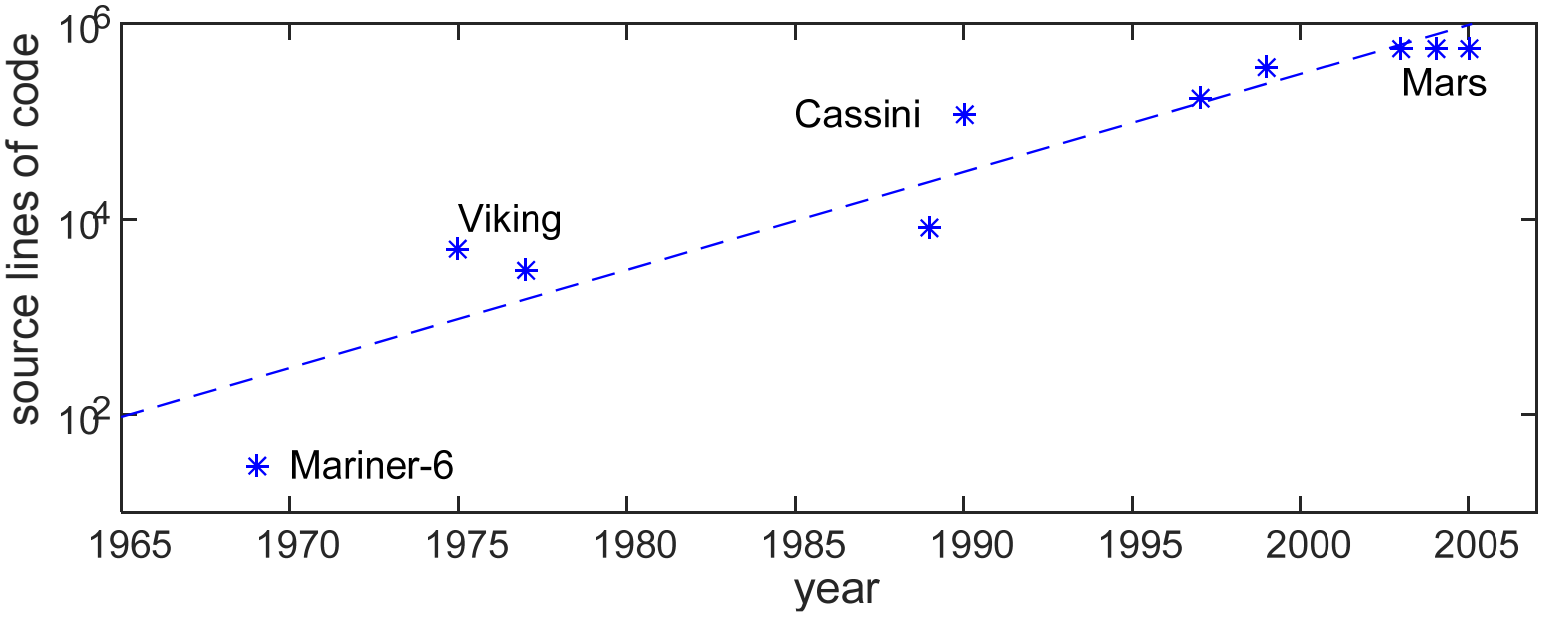


Figure adapted from Dvorak *et al.*,
NASA study on flight software complexity, 2009

Aerospace, Meet Internet

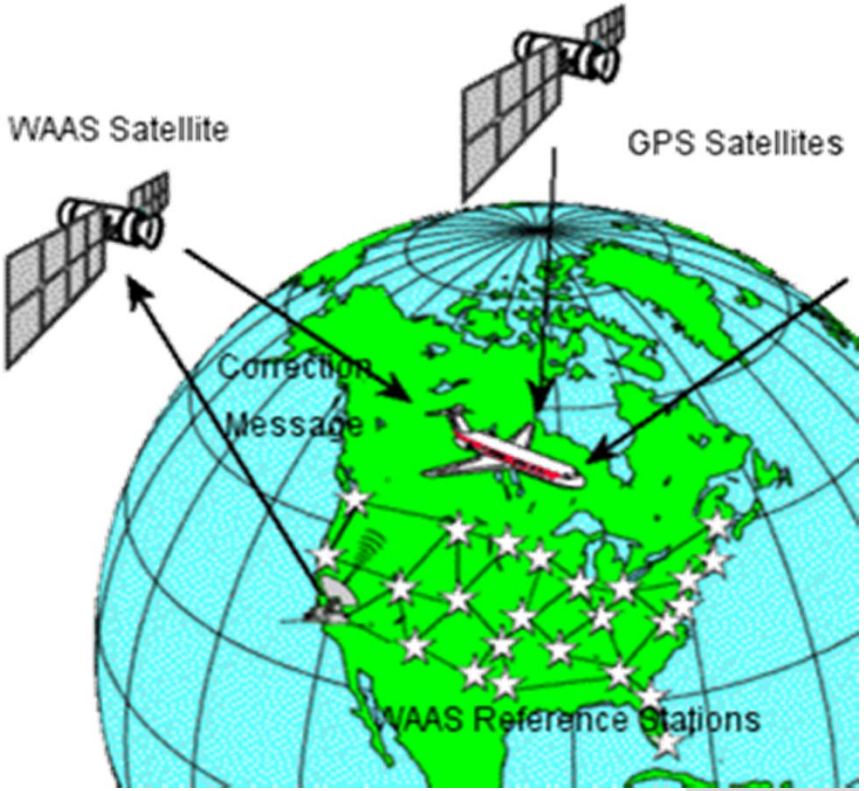
Tufts UNIVERSITY -ASAR Lab-

Continual Validation of Complex Systems

Inspection,
Nondestructive Evaluation

https://commons.wikimedia.org/wiki/File:Visual_inspection_of_compressor_blades-090108-F-9919G-746.jpg

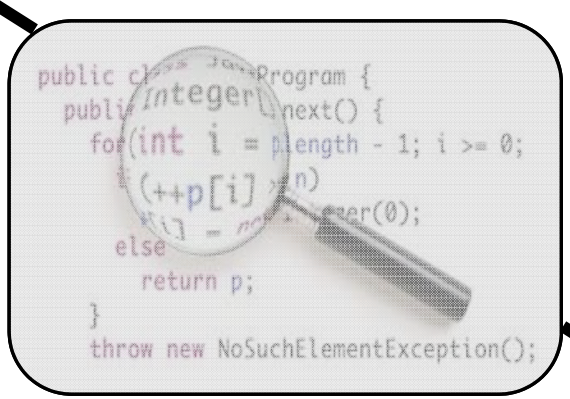
Monitoring of Navigation Signals
(e.g. SBAS)



https://commons.wikimedia.org/wiki/File:Waas.png

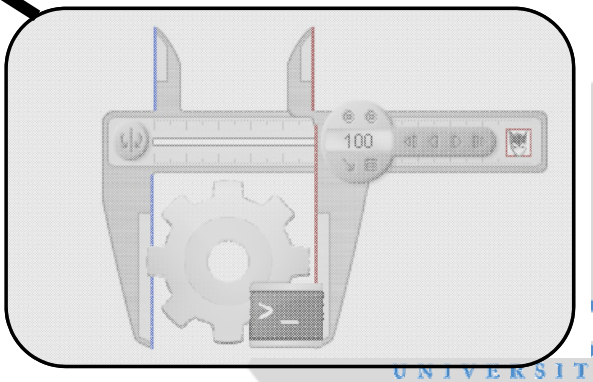
Augmenting the Design Toolbox

Language

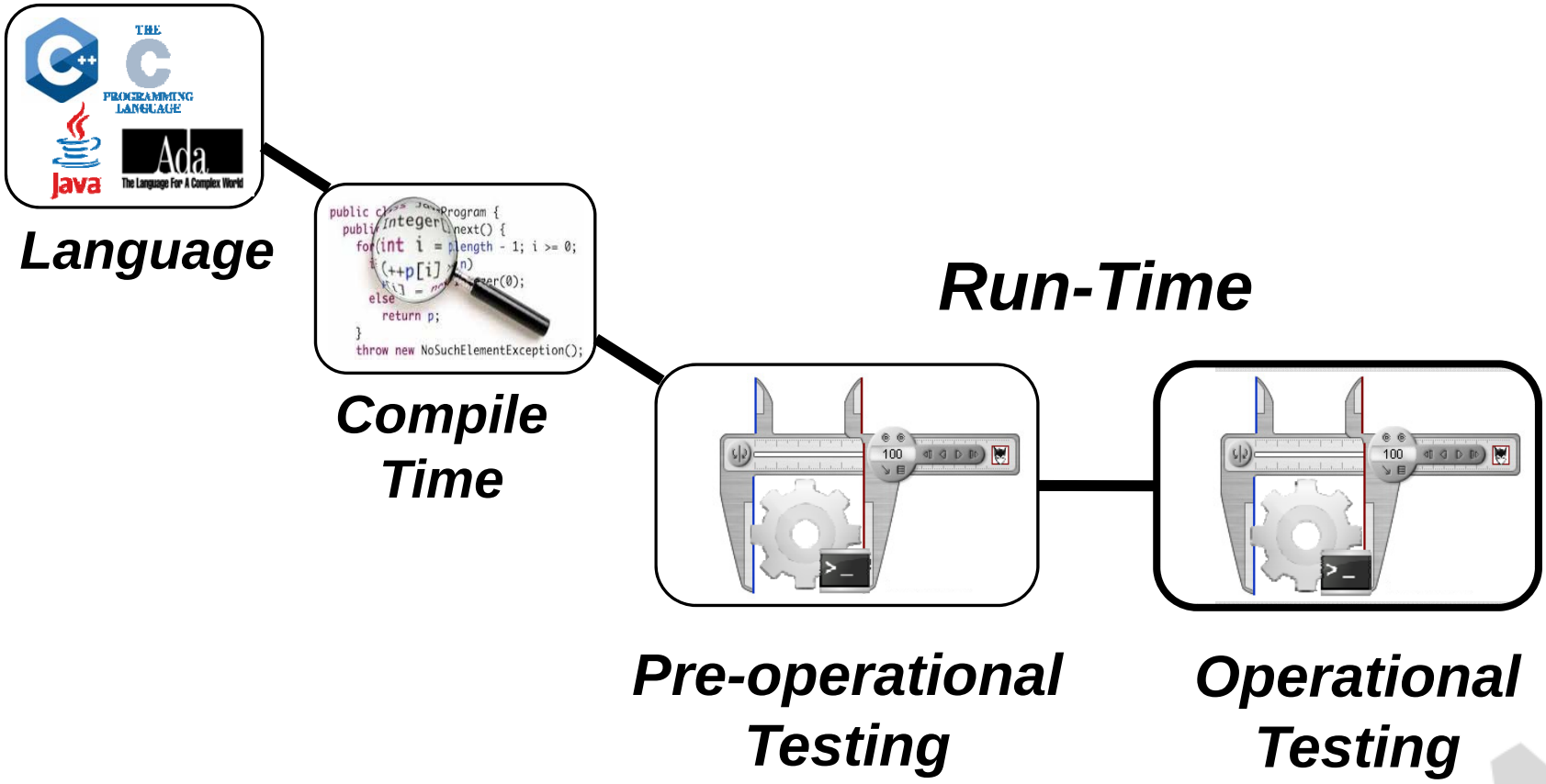


Compile Time

Run-Time



An Opportunity Exists



Open Questions

- How can we *design a bug monitor* analogous to monitors used for mechanical and electrical systems?
- How do we *train a monitor* to recognize what is normal behavior given that our software contains unknown bugs?
- How much can bug monitoring *reduce the burden* of pre-service verification?

Anonymous: https://pxhere.com/de/photo/689545

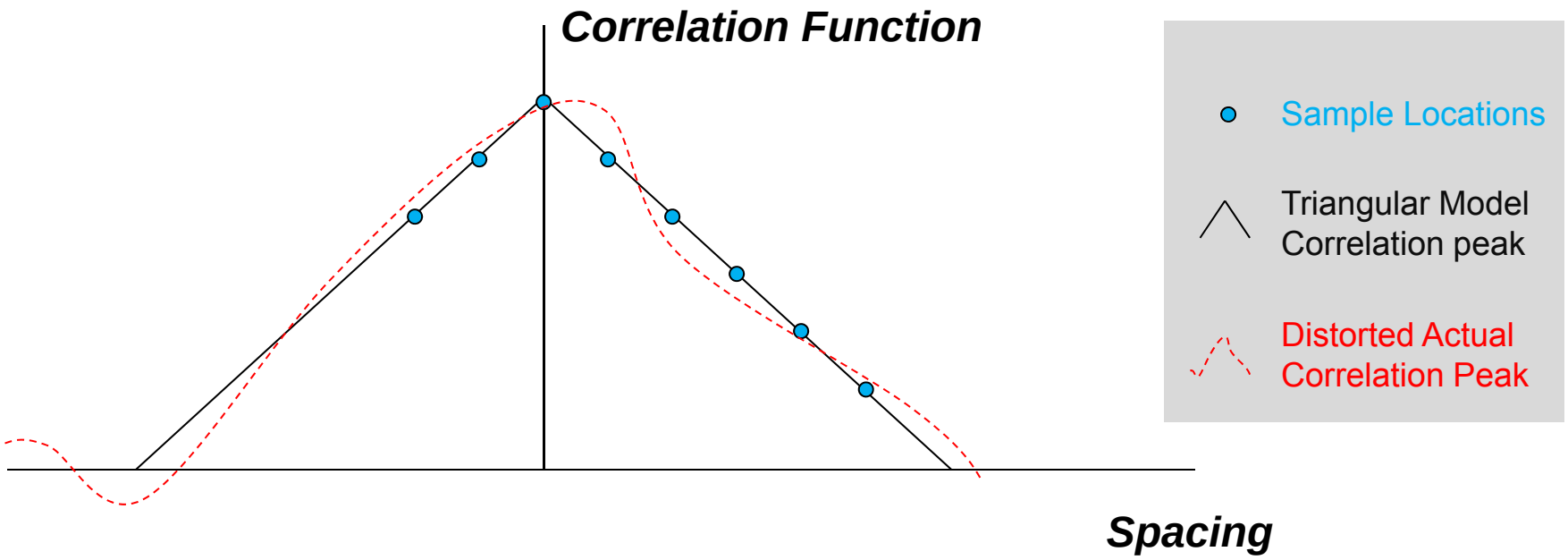
Open Questions

- How can we *design a bug monitor* analogous to monitors used for mechanical and electrical systems?
- How do we *train a monitor* to recognize what is normal behavior given that our software contains unknown bugs?
- How much can bug monitoring *reduce the burden* of pre-service verification?

Tufts UNIVERSITY -ASAR Lab-

Monitor Design

Signal Deformation Monitoring (SDM)



Sample correlation peak at 8 locations,
Difference 7 neighboring pairs, and
Compare to nominal values

Tufts UNIVERSITY -ASAR Lab-

SDM in GBAS

The 7 errors (measured pairs less reference) are normalized and compiled in the vector **y**

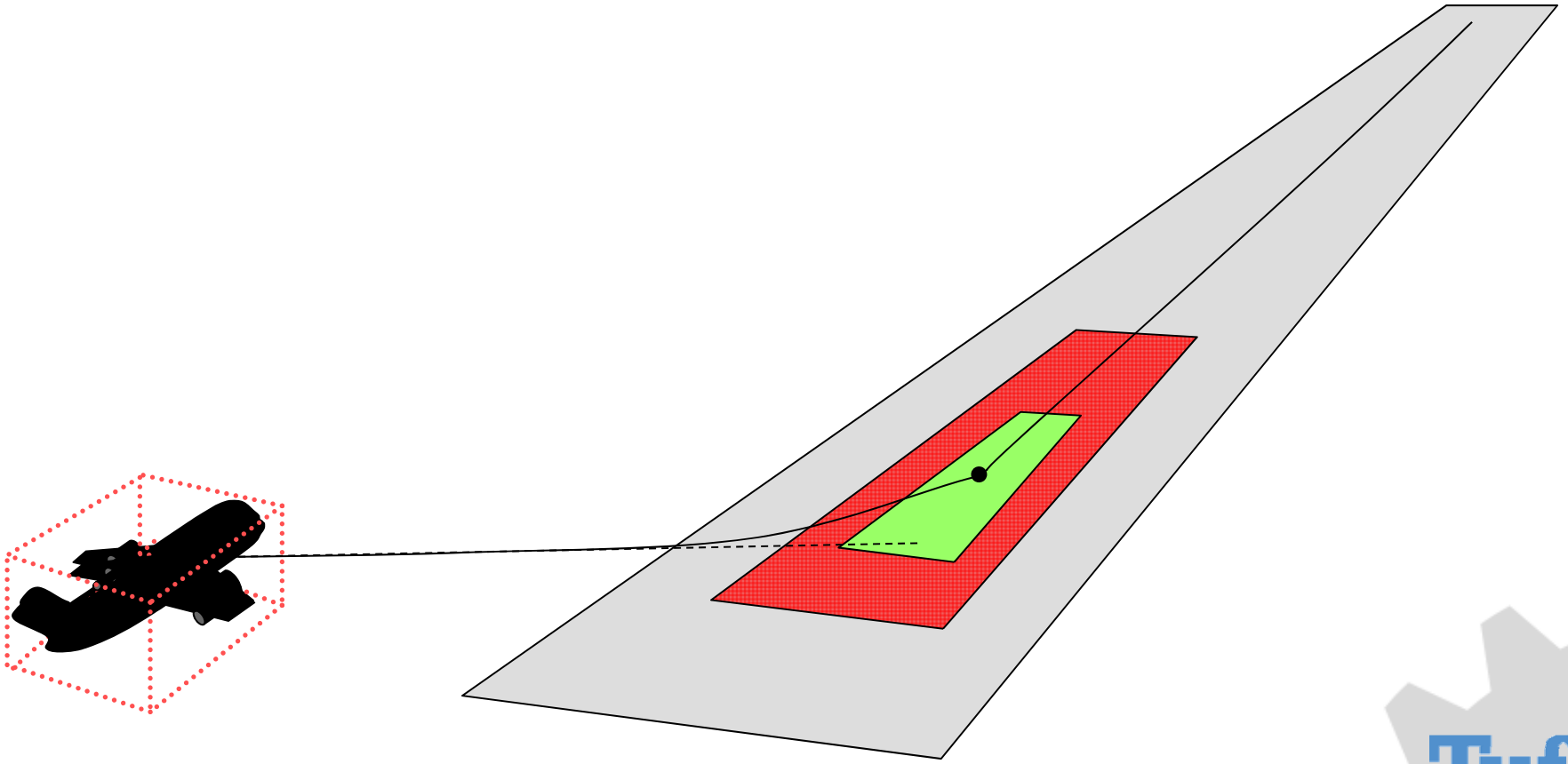
$$\mathbf{y} \in \mathbb{R}^7$$ ← *Input signal vector has expected value of zero and unity variance*

$$m = \mathbf{y}^T \mathbf{y}$$ ← *Test statistic (m) fuses input signal values (**y**)*

$$\text{alert if } m > T$$ ← *Alert issued if test statistic (m) exceeds threshold (T)*

Tufts UNIVERSITY -ASAR Lab-

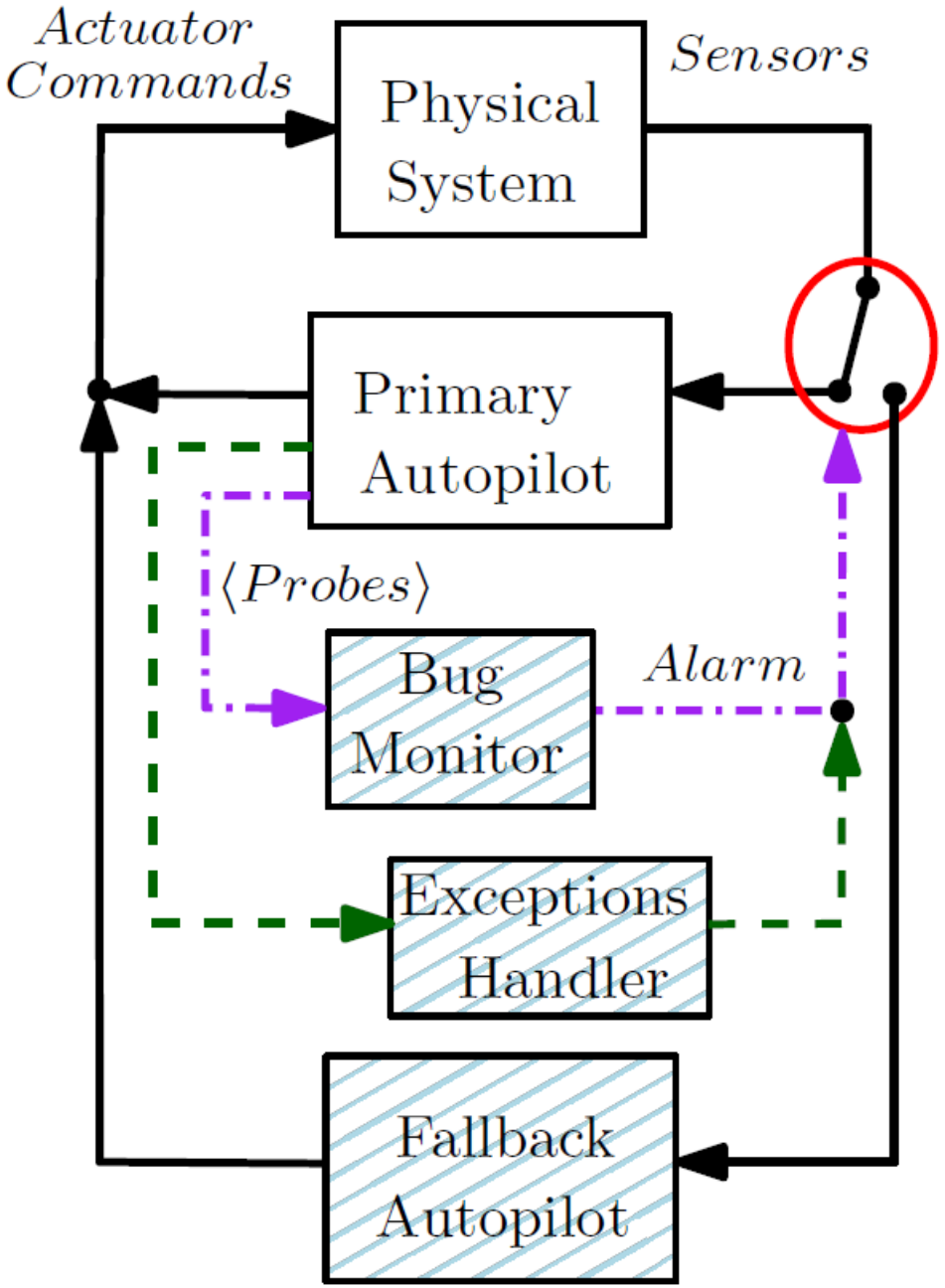
SDM (and other monitors) ensure integrity for safety-critical operations



A Bug Monitor…

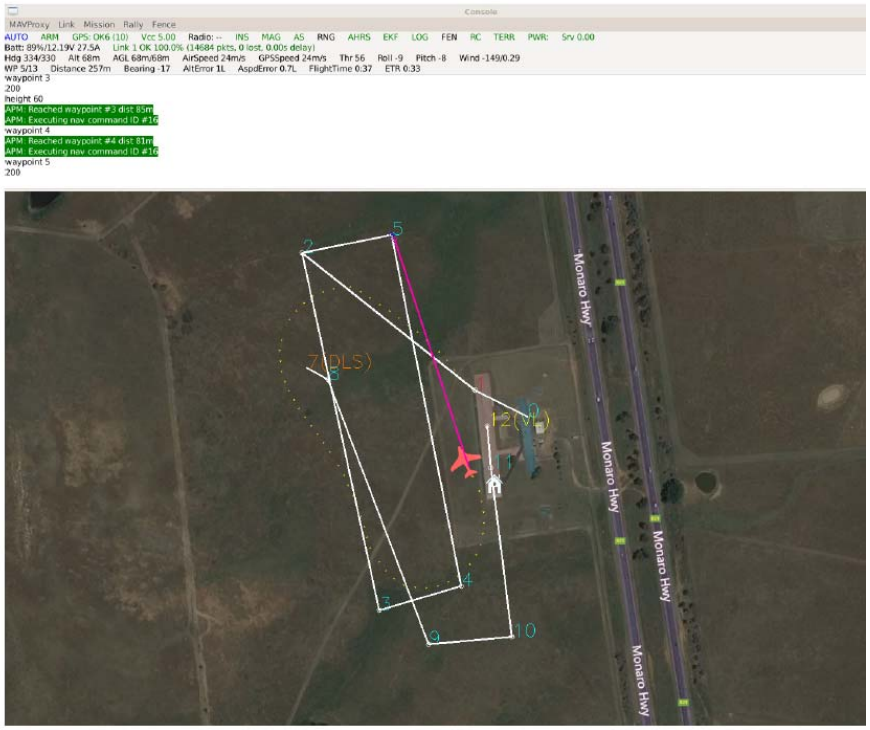
A Bug Monitor

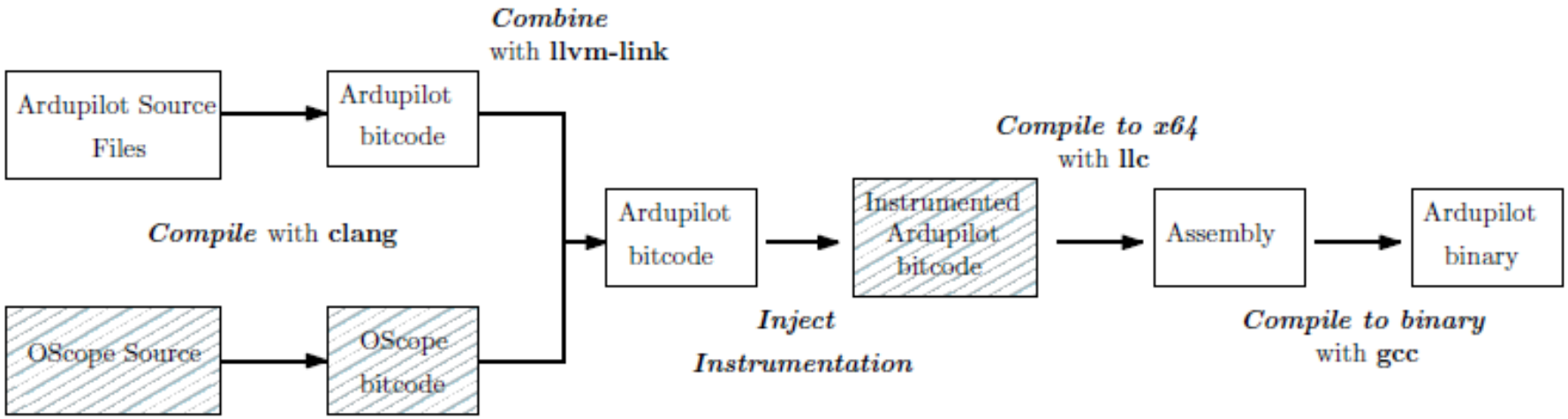
- Bug monitor looks for anomalous variable values
- Designer chooses a set of variables (or PROBES) to scan
- Choice of probes is critical
  - Maximize sensitivity
  - Minimize overhead



Implementation

Monitor Ardupilot +
JSBSim flight simulator





Choice of Probes

- In current work, consider two probe choices
  - Random: Sets of 10 variables randomly selected
  - Heuristic: Sensor/actuator interfaces with physical system

- In future work, seek to optimize probe selection

$$\begin{aligned} \min_{P \subset V} \quad & \text{overhead}(P) \\ \text{s.t.} \quad & \text{sensitivity}(P) \\ & \text{specificity}(P) \\ & \text{coverage}(P) \\ & \text{alert\_time}(P) \end{aligned}$$

Safety Assurance

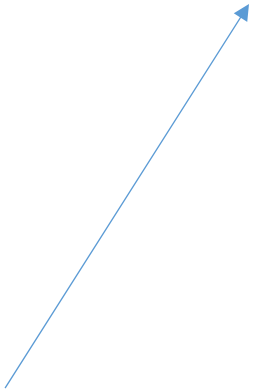
Open Questions

- How can we *design a bug monitor* analogous to monitors used for mechanical and electrical systems?
- How do we *train a monitor* to recognize what is normal behavior given that our software contains unknown bugs?
- How much can bug monitoring *reduce the burden* of pre-service verification?

Tufts UNIVERSITY -ASAR Lab-

Fault Tree Concepts: Single point of failure

System failure due to hidden bug

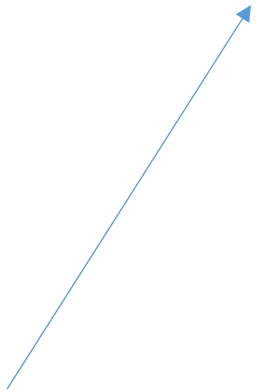


Pre-Service Verification

misses hazard

Fault Tree Concepts: Single point of failure

System failure due to hidden bug

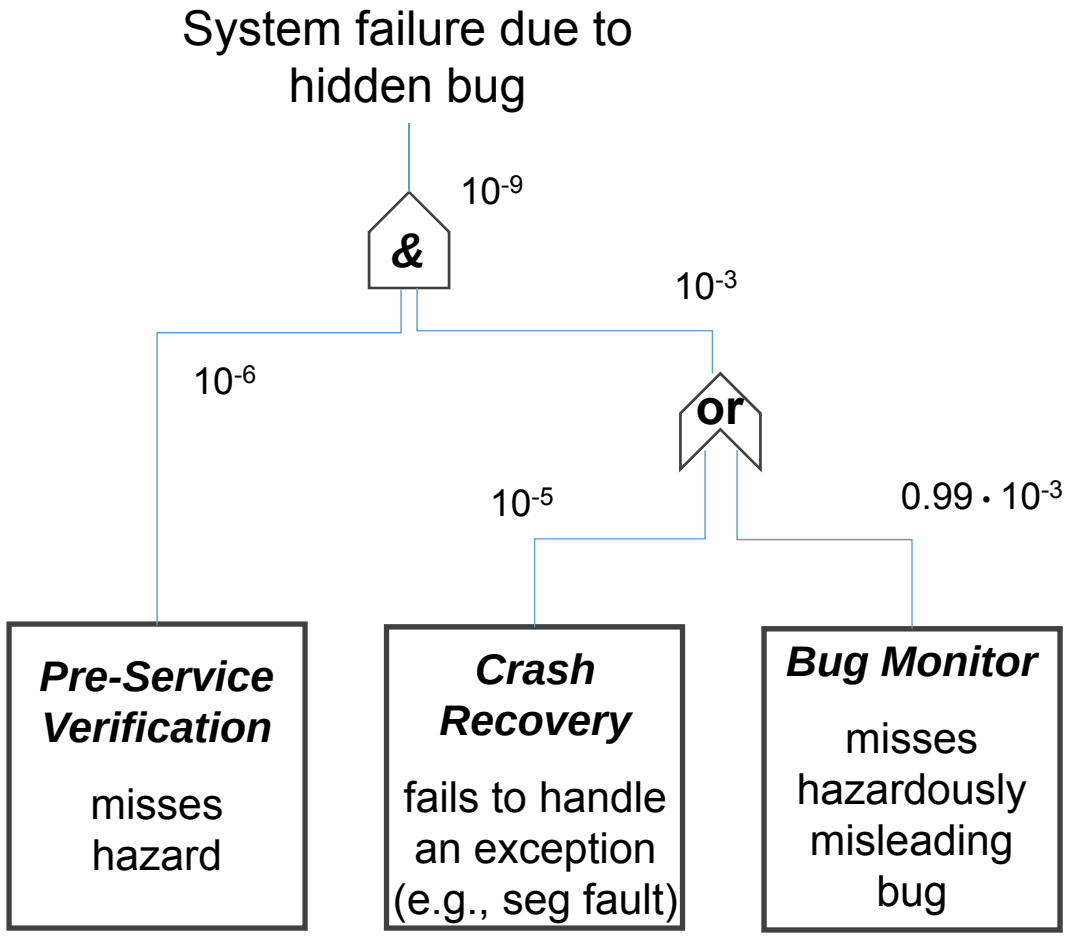


Could assign an allowable probability, such as 10^{-9}

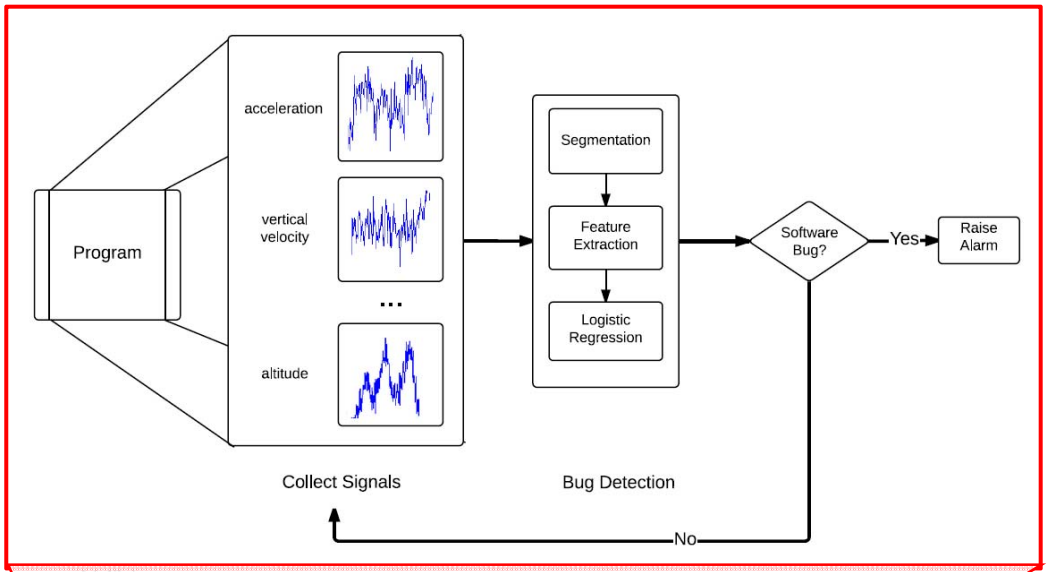
Pre-Service Verification

misses hazard

Fault Tree Concepts: Redundancy



Context: Monitor Function



System failure due to hidden bug

10^{-9}

&

10^{-4}

or

10^{-7}

$0.99 \cdot 10^{-5}$

Pre-Service Verification

misses hazard

Failsafe

misses clear fault (e.g., seg fault)

Bug Monitor

misses hazardously misleading bug

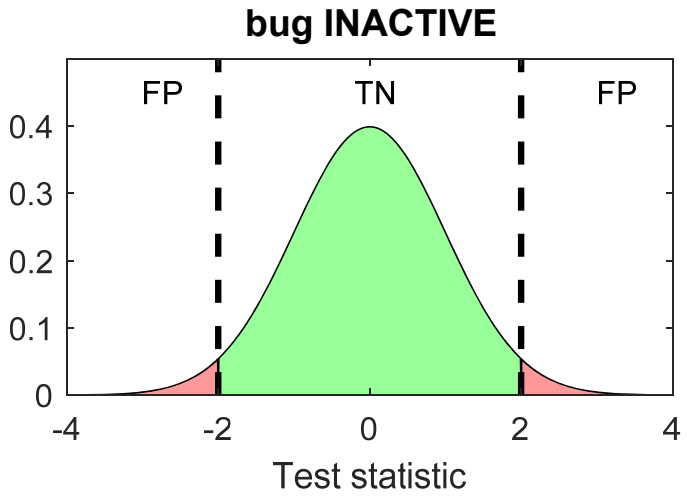
Tufts UNIVERSITY -ASAR Lab-

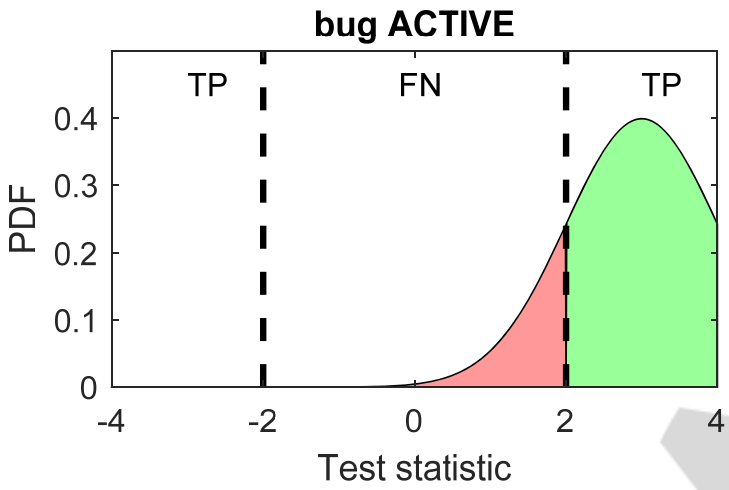
Monitor Performance

- **Alarm rate**: Rate of both false and true alarms
- **Bug-detection envelope**: Minimum "magnitude" bug that can reliably be detected
- **Loss-of-integrity risk**: Rate of undetected, severe bugs given pre-service verification and monitoring

Modeling Monitor to Quantify Performance

- Monitor looks for
  - Bugs that appear *infrequently* and are *difficult to find* ("Heisenbugs")
- Monitor warns of anomalies in real time
  - Test statistic compares machine-learned model to code outputs
  - Alarm occurs if statistic exceeds threshold
- As a starting point, simplify model of test-statistic noise
  - Assume distribution unchanged when bias active, *except* for mean
  - For approximate quantification only, assume a 1D Gaussian distribution





Performance Characterization

- **Alarm rate**:

$$P_a = P_{vf}\left(1 - P_{FN}\right) + \left(1 - P_{vf}\right)P_{FP}$$

- **Detection envelope**:

$$\mu = \text{solve}\left[P_{FN} = \int_{-T}^{T} p\left(x - \mu\right) dx \right]$$

- **Loss-of-integrity risk**:

$$P_{LOI} = P_{vf} P_{FN}$$

Threshold: $$T = \text{solve}\left[P_{FP} = 1 - \int_{-T}^{T} p\left(x\right) dx \right]$$

Problem parameterized by

$$P_a = P_{vf}\left(1 - P_{FN}\right) + \left(1 - P_{vf}\right)P_{FP} \qquad P_{LOI} = P_{vf}\,P_{FN}$$

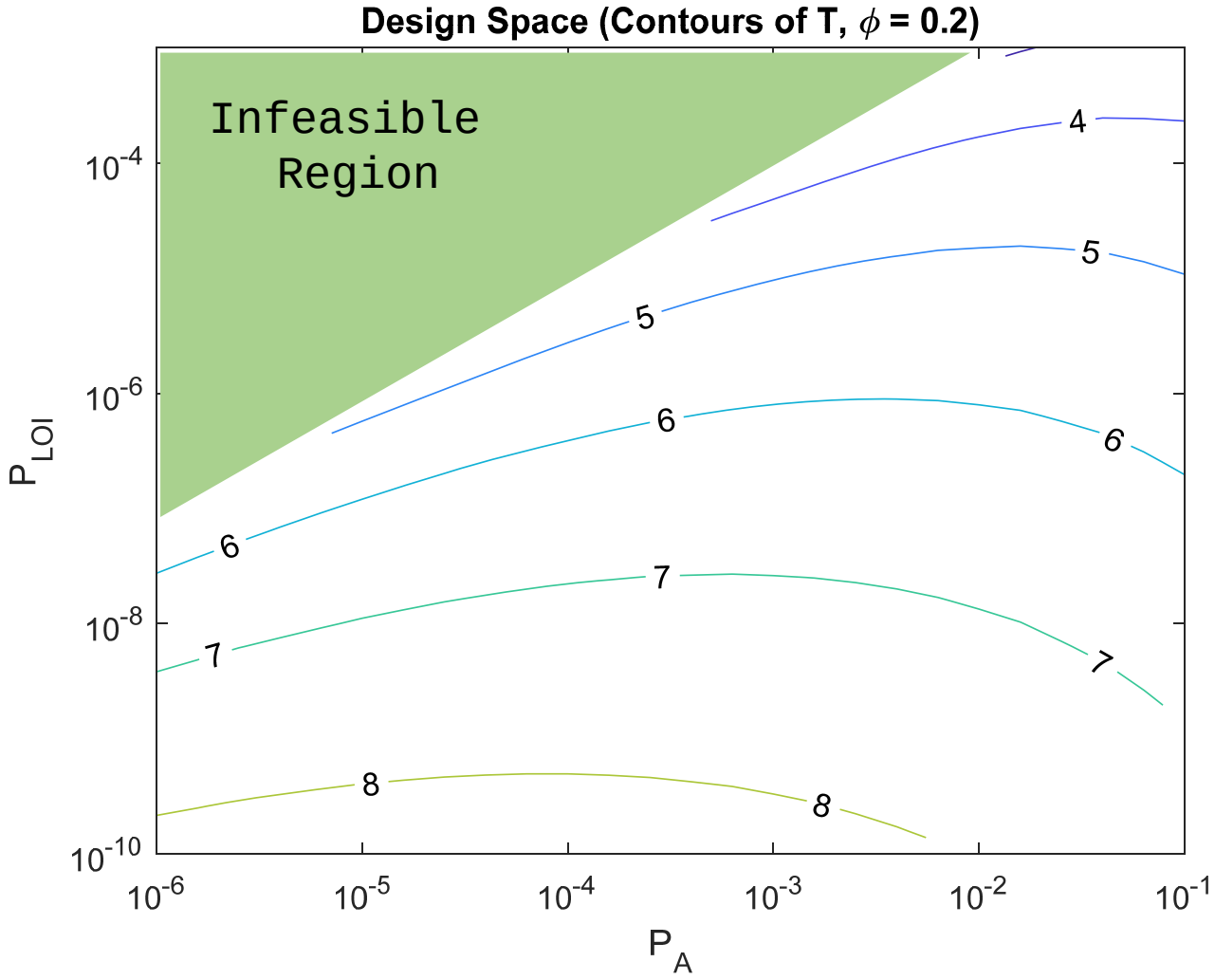
$$\mu = \text{solve}\left[P_{FN} = \int_{-T}^{T} p\left(x - \mu\right)dx\right] \qquad T = \text{solve}\left[P_{FP} = 1 - \int_{-T}^{T} p\left(x\right)dx\right]$$

- **P_{FP}: False-positive rate**
- **P_{FN}: False-negative rate**
- **P_{vf}: Verification fault probability**

Design Problem

- Find an acceptable set of output parameters (P_a, μ, P_{LOI})
- Selecting from space of input parameters (P_{FP}, P_{FN}).
  - Note: Low sensitivity to P_{vf} if $P_{vf} < 0.8\ P_a$

Achievable Designs



Contours relate three performance criteria:

$$\mu = f\left(P_a, P_{LOI}\right)$$

For details see: Rife, Hu, and Guyer. ION GNSS+ 2018.

Summary

- Bug monitoring demonstrated for real bugs in flight software
  - 70% sensitivity demonstrated using snapshot monitor and (!) random probes
  - Reasonable to infer higher sensitivity possible with guided probe selection and/or sequential monitoring

- Relaxing verification seems possible, even with low-sensitivity monitors
  - The Gaussian probability model is clearly very approximate
  - New challenge: Showing software robust to “small bugs”
  - Bottom line: For a given level of safety (e.g. loss-of-integrity probability), pre-service verification can be relaxed by *carefully* introducing bug monitoring

Acknowledgements

Support for research provided by NSF
(NSF-1329341 and NSF-1836942)