Software Robustness Testing and Run-Time Monitoring of Autonomous Vehicles

Electrical & Computer ENGINEERING

Prof. Phil Koopman
koopman@cmu.edu

This material is based upon work supported by the Test Resource Management Center (TRMC) Test and Evaluation/Science & Technology (T&E/S&T) Program through the U.S. Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) under Contract No. W900KK-11-C-0025, "Stress Testing for Autonomy Architectures (STAA)".

Overview

- Very brief CMU overview
- Autonomous vehicle & robotic software safety
  - Goes beyond current software safety standards
- Automated robustness testing
  - Finds significant software defects
- Run-time safety monitors
  - Used on large autonomous vehicle to ensure safety
- ASTAA project: automated stress testing of robots
  - ASTAA = Robustness stress testing + simple safety monitors
- Some future challenges
  - Getting from demos to full scale deployment will be hard!

Carnegie Mellon University, Pittsburgh PA, USA

Electrical & Computer Engineering

Electrical & Computer ENGINEERING

Pittsburgh, PA.

BS, MS & PhD degrees in ECE

Contact: Ed Schlesinger, ed@ece.cmu.edu

Silicon Valley, CA.

MS degrees in Software Engineering, Soft. Management, IT, ECE & PhD in ECE

Contact: Martin Griss, martin.griss@sv.cmu.edu

ICTI, Portugal

PhD degrees in ECE

Contact: Jose Moura, moura@ece.cmu.edu

Dehli, India.

BS degree in ECE

Contact: Ed Schlesinger, ed@ece.cmu.edu

Shiv Nadar University

Carnegie Mellon University

Kigali, Rwanda.

MS degree in Information Technology

Contact: Bruce Krogh, krogh@ece.cmu.edu

Guangzhou, China.

Dual-degrees for MS & PhD in ECE

Contact: Jimmy Zhu, jzhu@ece.cmu.edu

Joint Institute of Engineering

SUN YAT-SEN UNIVERSITY

Carnegie Mellon University

Singapore

PhD in ECE

Contact: Ed Schlesinger, ed@ece.cmu.edu

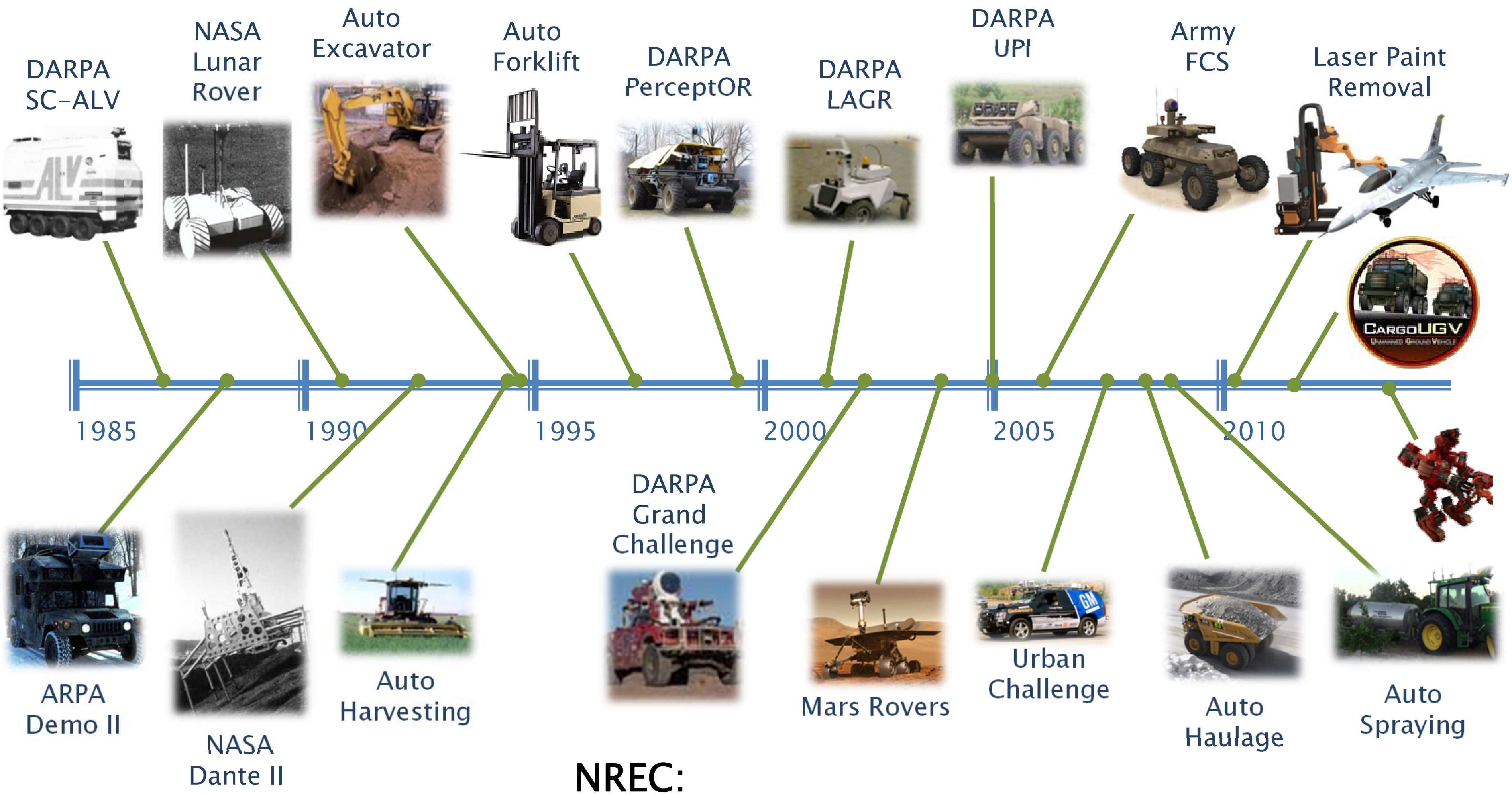
Agency for Science, Technology and Research

SINGAPORE

ECE Department:
~100 Faculty
~150 undergrads/yr
~500 grad students
(Note: Computer Science is a whole school)

National Robotics Engineering Center

NATIONAL ROBOTICS
NREC
ENGINEERING CENTER



NREC:
~175 Faculty, staff, students
Off-campus Robotics Institute facility, SCS
Engineering & Technology Transfer

How Well Tested Are Autonomy Features?

About the Administrator
Calendar
Congressional Testimony
Jobs at NHTSA
Presentations & Speeches
Press Releases
Programs & Grants
Print | Share | RSS Feed | Email

USDOT Proposes Updated Safety Standard to Prioritize Braking Control, Reduce Risk of High-Speed Unintended Acceleration for Nation's Cars

NHTSA 04-12
April 12, 2012
Contact: Karen Aldana, 202-366-9550

'Brake-Throttle Override' requirement will reduce the risk of high-speed unintend trapped accelerator pedal

WASHINGTON - The U.S. Department of Transportation's National Highway Traffic Safety Adminis ... both the bra ... will help red ... allowing th ... ey can easil ... Hood. "By u ... down while

Jaguar recalls 18,000 cars over cruise control software fault

Car system upgrade needed, but no hardware affected

By Leo King | Computerworld UK | Published 10:23, 24 October 11

+1 10 | Like 126 | Tweet 73

Jaguar has recalled nearly 18,000 X-type cars after it discovered a major software fault, which meant drivers might not be able to turn off cruise control.

The problem lies with engine management control software developed in-house by Jaguar. The problematic software is only installed on diesel engine X-Types, which were all produced between 2006 and 2010.

Some 17,678 vehicles have been recalled, as a result of the potentially dangerous problem. If the fault occurs, cruise control can only be disabled by turning of the ignition while driving - which would mean a loss of some control and in many cars also disables power steering. Braking or pressing the cancel button will not work.

"Jaguar has identified that should an error with certain interfacing systems be detected the cruise control system will be disabled and an error message displayed to the driver on the instrument cluster," the company said in a statement.

[Koopman 2013]

Part of complete coverage on **Powering the Planet**

Self-driving cars now legal in California

By Heather Kelly, CNN
updated 12:30 PM EDT, Tue October 30, 2012 | Filed under: Innovations

19k Recommend | 1,511 Tweet | 231 Share | 220 +1
Print | Email | More sharing

Driverless car now legal in California

STORY HIGHLIGHTS

- Gov. Jerry Brown signs a bill that will regulate self-driving cars in California
- Google co-founder Sergey Brin hopes to have self-driving cars on public roads within five years
- Brin says self-driving cars address safety, traffic and lifestyle issues

Read a version of this in Arabic

(CNN) -- California is the latest state to allow testing of Google's self-driving cars on the roads, though only with a human passenger along as a safety measure.

Gov. Edmund "Jerry" Brown signed the autonomous-vehicles bill into law Tuesday afternoon alongside Google co-founder Sergey Brin and State Sen. Alex Padilla, who authored the bill, at Google's headquarters in Mountain View, California. The bill, SB 1298, will set up procedures and requirements for determining when the cars are road-ready.

Brin hopes that self-driving cars will be able to drive on public streets in five years or less.

Part of complete coverage on **Our Mobile Society**

Smartphone of the future will be in your brain
updated 9:47 AM EDT, Mon October 8, 2012

In the past 10 years we've seen cell phones transform into electronic Swiss army knives with a wild variety of functions and features.

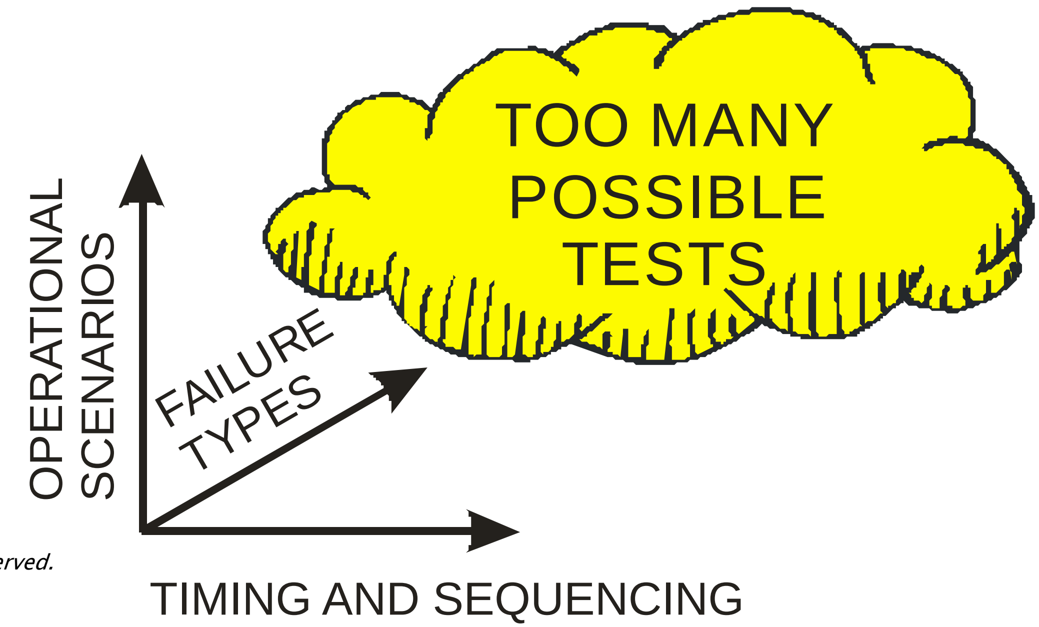
Diaries of 3 smartphone addicts
updated 8:03 AM EDT, Tue October 9, 2012

If you're like Derek Smith, you spend a lot of time on your smartphone. Then again, maybe nobody is quite like Derek Smith.

In defense of my stupidphone

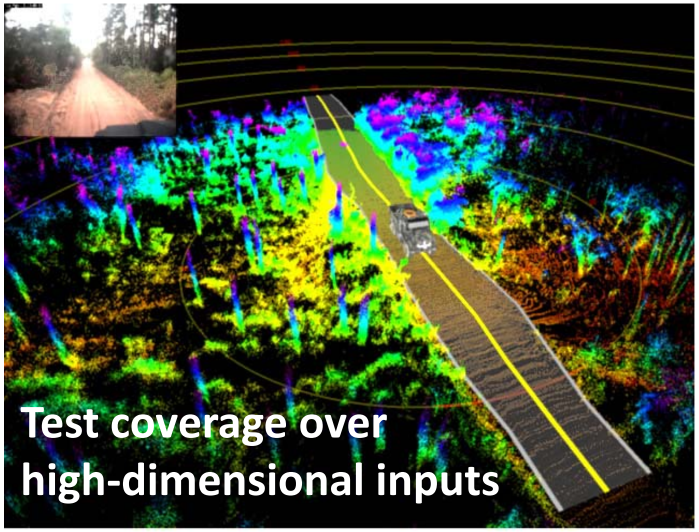
Testing Isn't Enough To Ensure SW Safety

- In current systems, system-level testing is useful and important
  - It can find unexpected component interactions
- ***But,*** it is impracticable to test everything at the vehicle/system level
  - There are too many possible operating conditions
  - There are too many possible timing sequences of events
  - There are too many possible faults
  - All possible combinations of component failures and memory corruptions
  - Multiple software defects activated by a sequence of operations

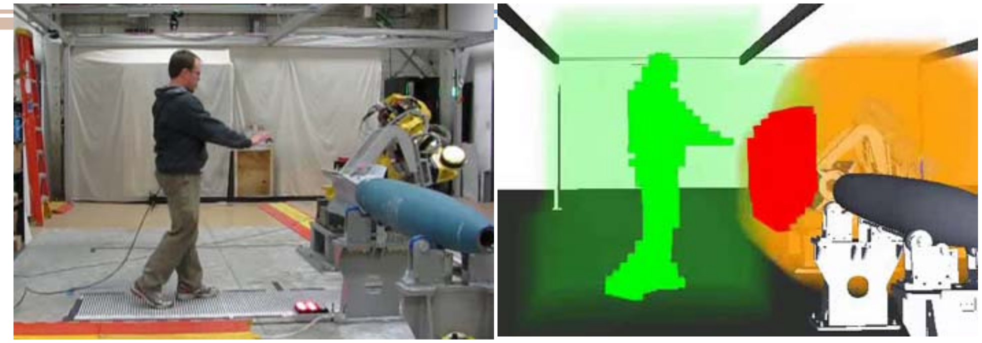


[Koopman 2013]

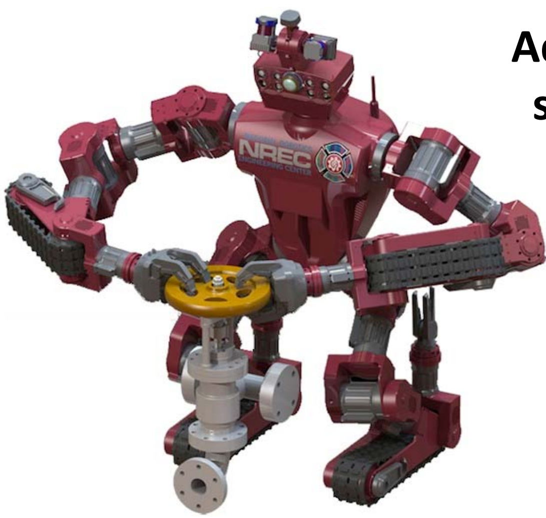
Robot Testing Is Even More Difficult



Test coverage over high-dimensional inputs



Sensitivity to calibration



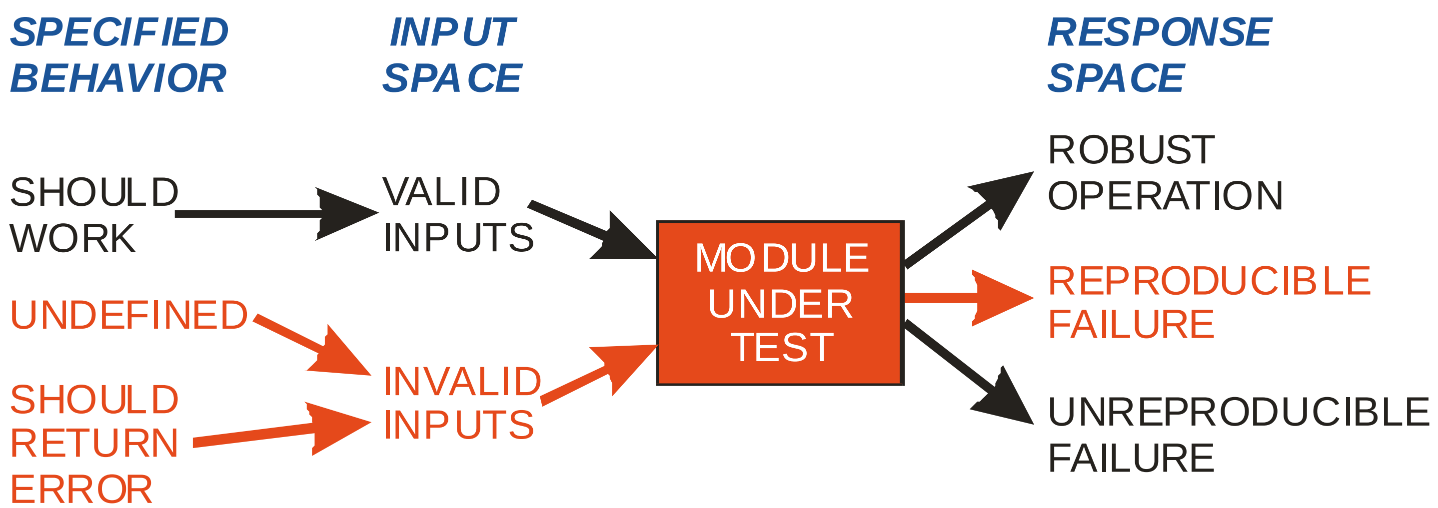
Adaptive systems

Non-linear motion planning

Validation of machine learning results

[Wagner/Koopman]

Software Stress Testing may increase test coverage



- Fuzz testing [Miller98] uses a random input stream
  - Finds interesting failures
  - But can be inefficient
- Ballista (1996..2008) uses "dictionaries" of values
  - Combinations of exceptional and ordinary values
  - More efficient, but still scalable, approach to robustness testing

Ballista Scalable Test Generation

API `write(int filedes, const void *buffer, size_t nbytes)`

TESTING OBJECTS

FILE DESCRIPTOR TEST OBJECT	MEMORY BUFFER TEST OBJECT	SIZE TEST OBJECT
FD_CLOSED	BUF_SMALL_1	SIZE_1
FD_OPEN_READ	BUF_MED_PAGESIZE	SIZE_16
FD_OPEN_WRITE	BUF_LARGE_512MB	SIZE_PAGE
FD_DELETED	BUF_XLARGE_1GB	SIZE_PAGEx16
FD_NOEXIST	BUF_HUGE_2GB	SIZE_PAGEx16plus1
FD_EMPTY_FILE	BUF_MAXULONG_SIZE	SIZE_MAXINT
FD_PAST_END	BUF_64K	SIZE_MININT
FD_BEFORE_BEG	BUF_END_MED	SIZE_ZERO
FD_PIPE_IN	BUF_FAR_PAST	SIZE_NEG
FD_PIPE_OUT	BUF_ODD_ADDR	
FD_PIPE_IN_BLOCK	BUF_FREED	
FD_PIPE_OUT_BLOCK	BUF_CODE	
FD_TERM	BUF_16	
FD_SHM_READ	BUF_NULL	
FD_SHM_RW	BUF_NEG_ONE	
FD_MAXINT		
FD_NEG_ONE		

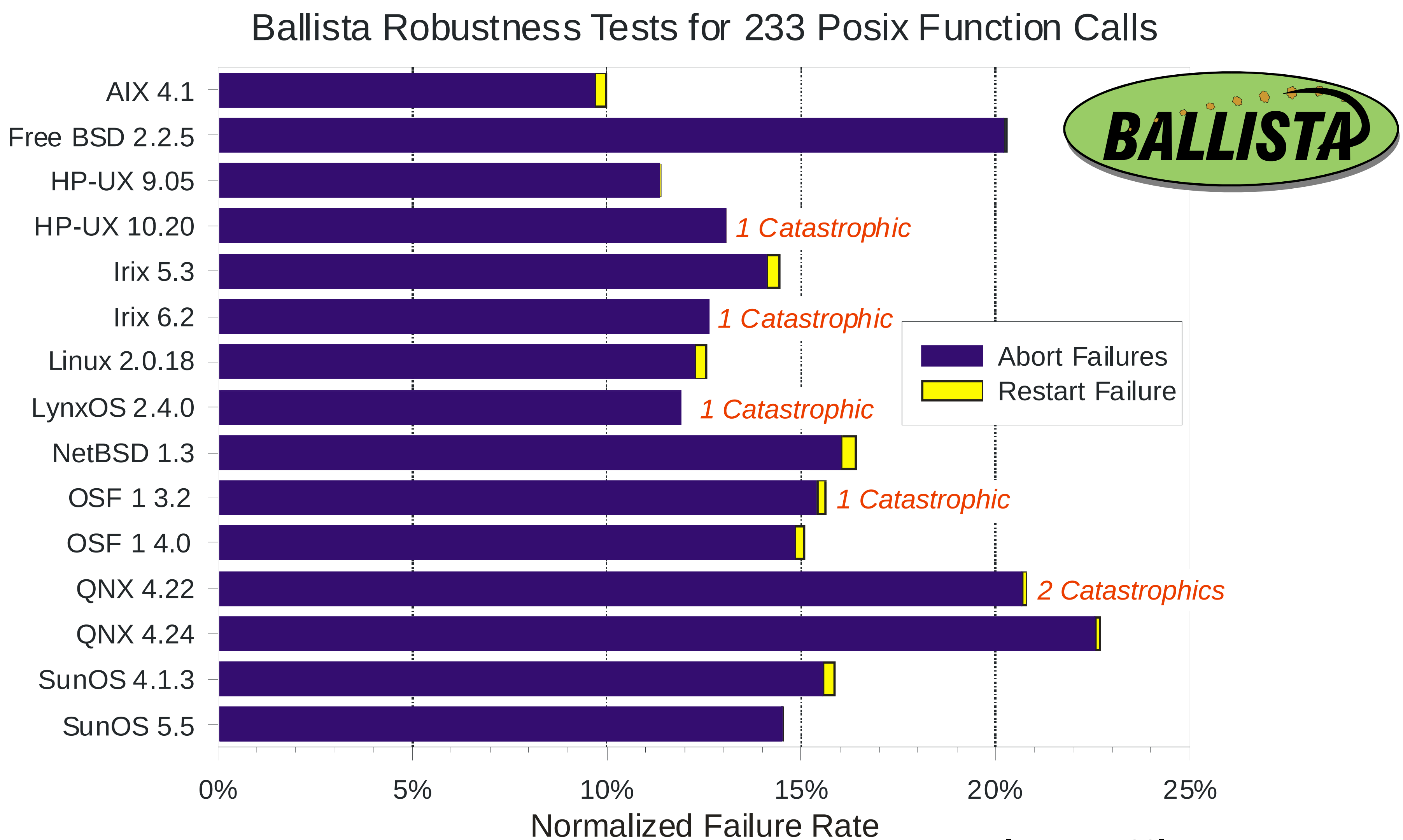
TEST VALUES

TEST CASE `write(FD_OPEN_RD, BUFF_NULL, SIZE_16)`

- Generates test cases based on parameter data types
  - Ignoring functional 'correctness' provides scalability

[Koopman / Ballista]

Ballista Found Plenty of Robustness Issues!



[Koopman99]

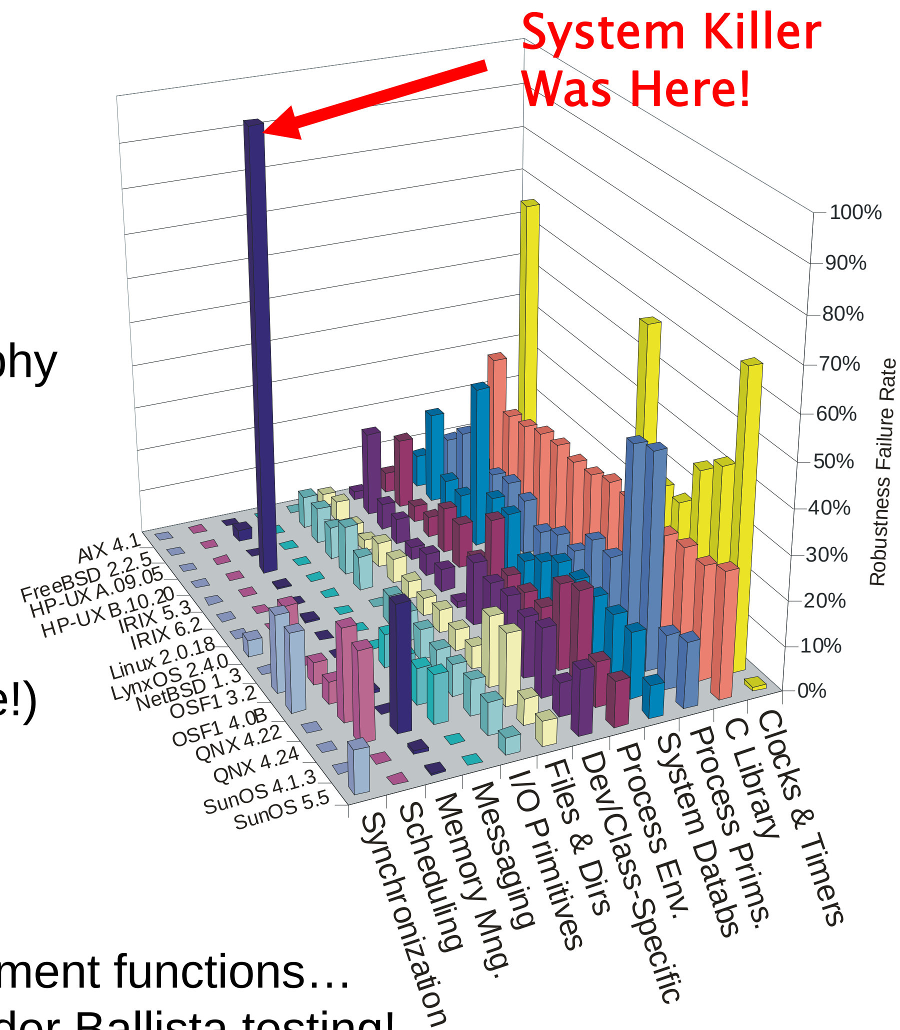
[Koopman / Ballista]

Abort Failures Might Predict Bad Software Quality

- "Abort" failures are a core dump
  - Individual process crash rather than system crash
  - Whether a process crash matters depends upon your system & philosophy

- Most failures found were highly repeatable, "one-liner" calls
  - Not race conditions (surprise!)
  - Not long complex sequences (surprise!)

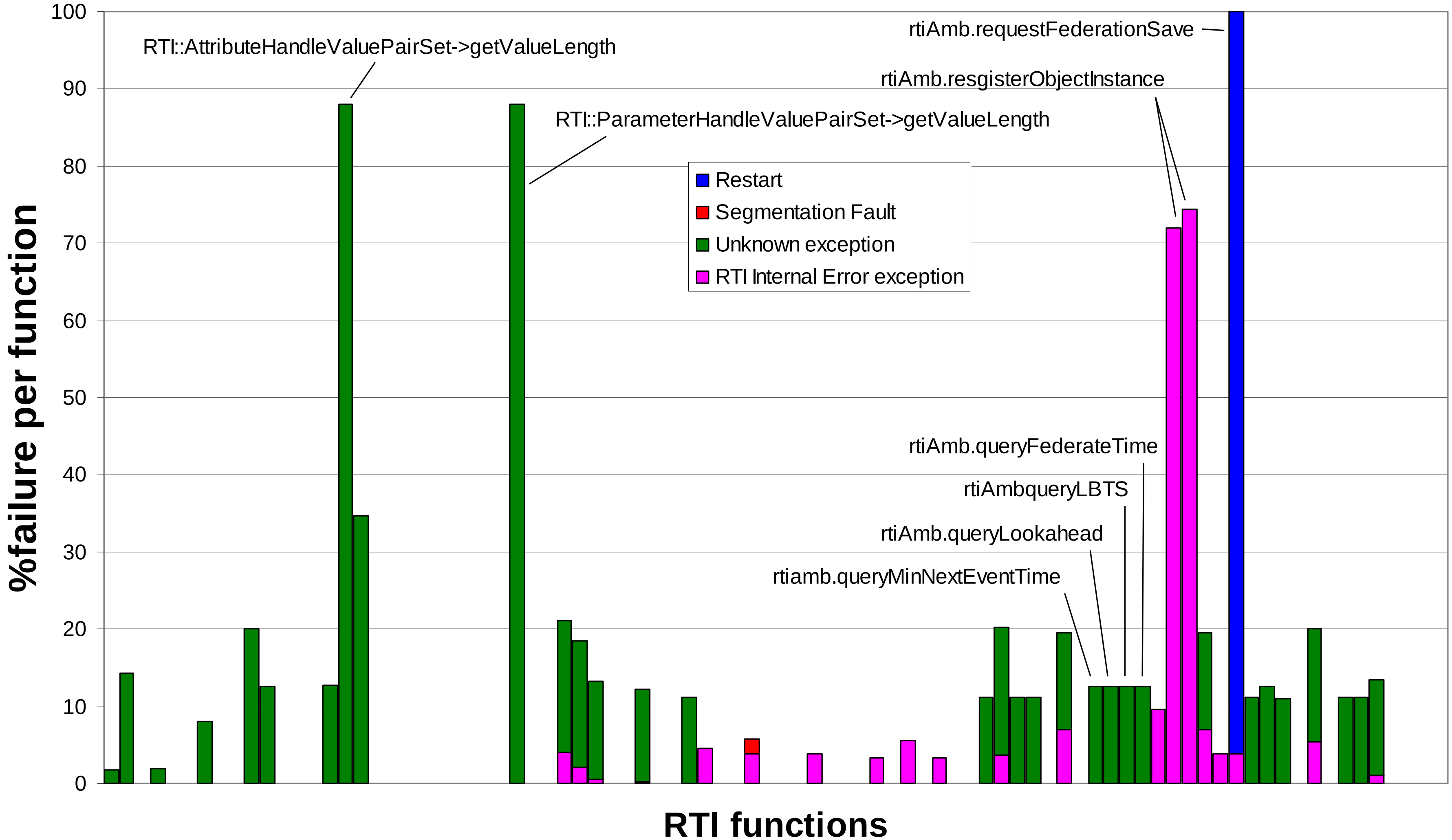
- HP-UX gained a system-killer in upgrade from Version 9 to 10
  - In newly re-written memory management functions…
    … which had a 100% failure rate under Ballista testing!



[Koopman / Ballista]

RTI-HLA Simulation Backplane/Middleware

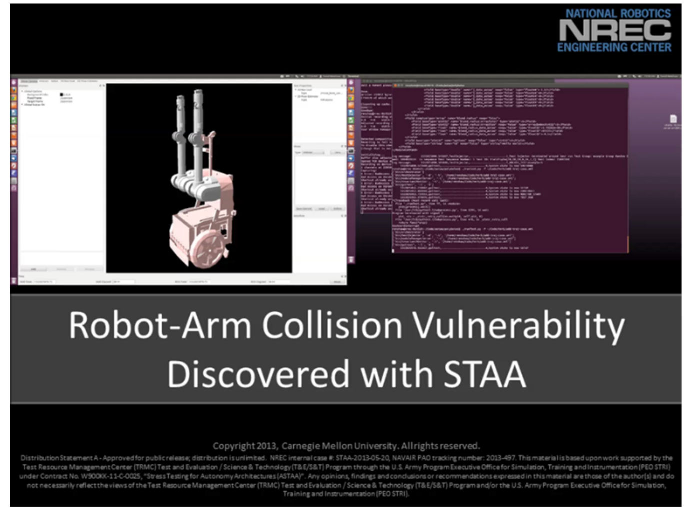
Robustness Failures of RTI 1.3.5 for Digital Unix 4.0



[Koopman / Ballista]

STATED GOAL OF HLA: 100% Robust

Stress Testing Finds Bugs On Robots Too…



Important vulnerabilities have been found in over twenty systems tested on our project so far

…

more to come

…


CRUSHER

But, safety standards might not apply: (Example from IEC-61508)

	Technique/measure	Ref	SIL3	Interpretation in this application
1	Fault detection and diagnosis	C.3.1	HR	Used as far as dealing with sensor, actuator and data transmission failures and which are not covered by the measures within the embedded system according to the requirements of IEC 61508-2
2	Error detecting and correcting codes	C.3.2	R	Only for external data transmissions
3a	Failure assertion programming	C.3.3	R	Results of the application functions are checked for validity
3b	Safety bag techniques	C.3.4	R	Used for some safety related functions where 3a and 3c are not used
3c	Diverse programming	C.3.5	R	Used for some functions where source code is not available
3d	Recovery block	C.3.6	R	Not used
3e	Backward recovery	C.3.7	R	Not used
3f	Forward recovery	C.3.8	R	Not used
3g	Re-try fault recovery mechanisms	C.3.9	R	Not used
3h	Memorizing executed cases	C.3.10	R	Not used (measures 3a, 3b and 3c are sufficient)
4	Graceful degradation	C.3.11	HR	Yes, because of the nature of the technical process
5	Artificial intelligence - fault correction	C.3.12	NR	Not used
6	Dynamic reconfiguration	C.3.13	NR	Not used

APD (Autonomous Platform Demonstrator)

How did we make this scenario safe?

NATIONAL ROBOTICS
NREC
ENGINEERING CENTER

TARGET GVW: 8,500 kg
TARGET SPEED: 80 km/hr

Approved for Public Release. TACOM Case #20247 Date: 07 OCT 2009

APD Safety System

The Autonomous Platform Demonstrator (APD) was the first UGV to use a Safety Monitor as part of its safety case.

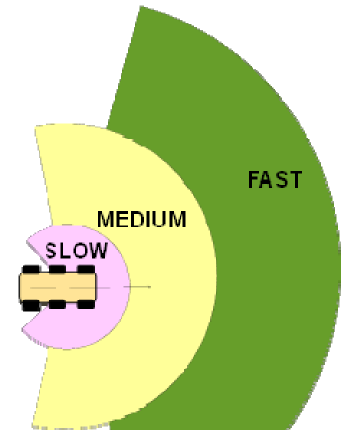
As a result, the U.S. Army approved APD for demonstrations involving soldier participation.

U.S. Army cites high quality of APD safety case and turns to NREC to improve the safety of unmanned vehicles.

Objective: Enforce and control safe standoff distance between APD and nearby personnel.

Approach:

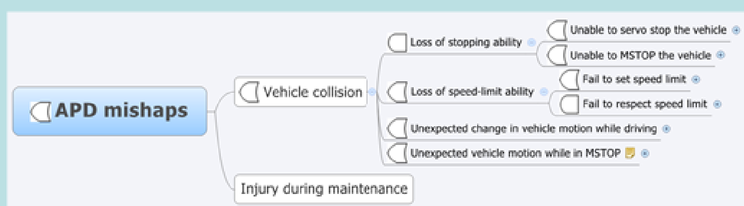
- Provide fail-safe braking mechanisms with well-modeled stopping distance.
- Incorporate Safety Monitor for redundant, high-reliability means of restraining vehicle speed.
- Identify and mitigate risks that could lead to failures of braking and speed-limiting.



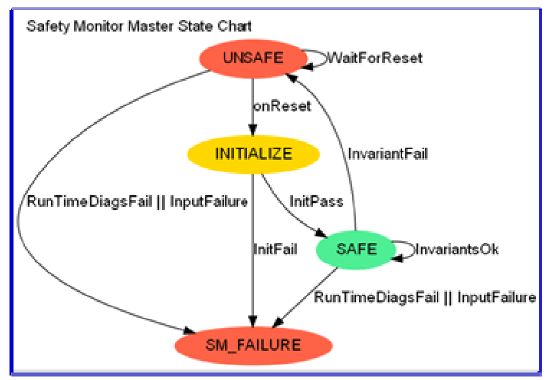
Reliable speed limiting allows safe standoff distances to be decreased

Techniques:

- Identifying hazards that lead to safety mishaps.
- Modeling of correlation between latent hazards with rich instrumentation.
- Firewalling safety-criticality to a subset of vehicle components.
- Developing & testing fault-resistant software for speed limiting.
- V&V testing traced to safety requirements.



Careful analysis of mishaps drives safety system design

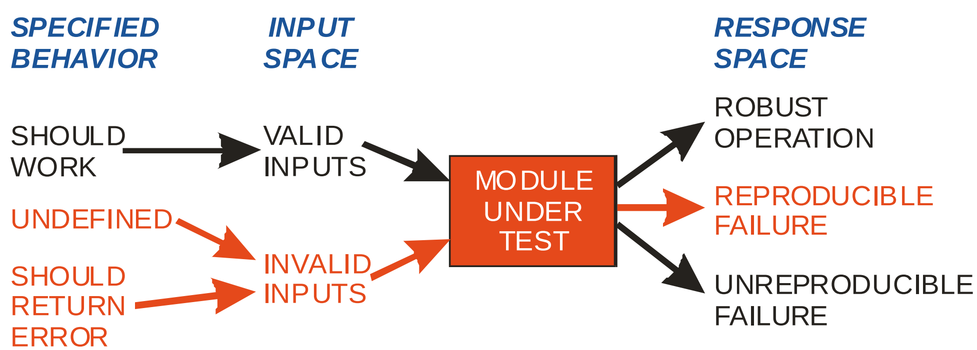


Safety Monitor ensures that safety invariants are maintained

NATIONAL ROBOTICS ENGINEERING CENTER

TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

How Can We Combine These Ideas?



+

Ballista Stress-Testing Tool

Robustness testing of defined interfaces

- Most test cases are exceptional
- Test cases based on best-practice software testing methodology
- Detects software hanging or crashing

Earlier work looked at stress-testing COTS operating systems

Uncovered system-killer crash vulnerabilities in top-of-the-line commercial operating systems

NREC Safety Monitor

Monitors safety invariants at run-time

- Designed as run-time safety shutdown box for UAS applications

Independently senses system state to determine whether invariants are violated

Firewalls safety-criticality into a small, manageable subset of a complex UAS; prototype deployed on Autonomous Platform Demonstrator (APD), a 9-ton UGV capable of reaching 80 km/hr

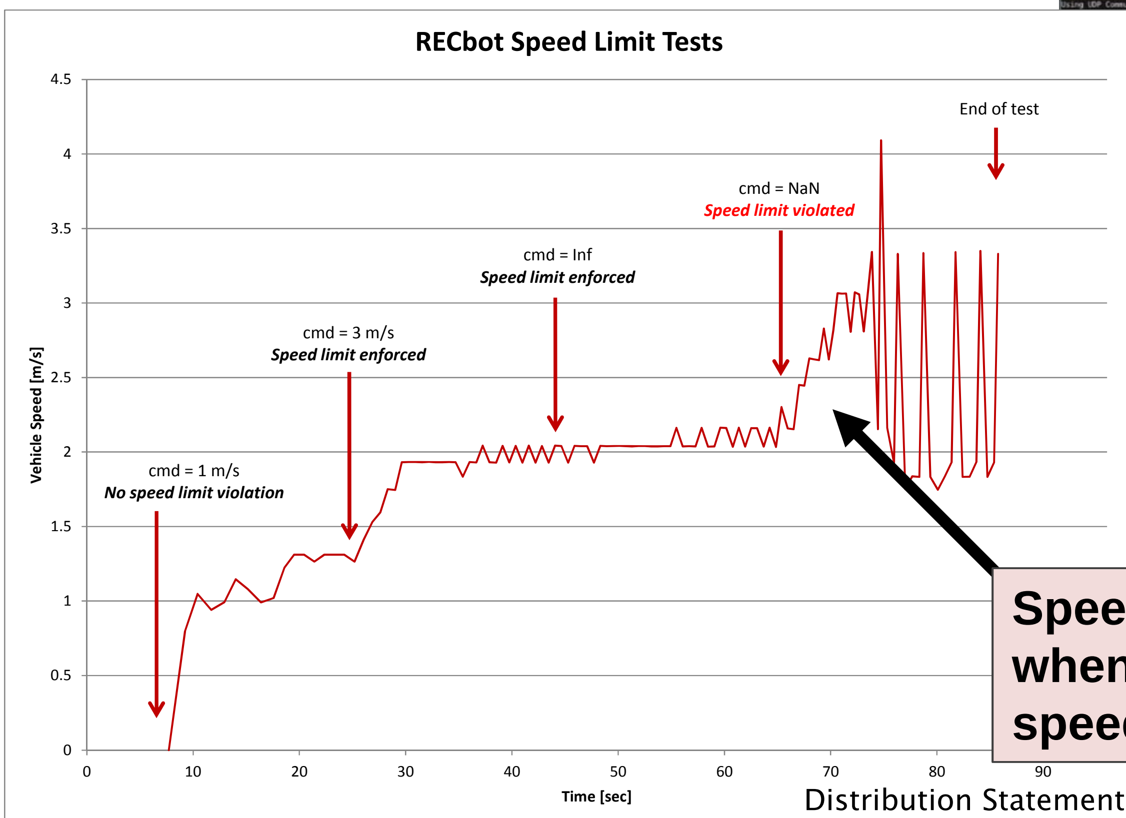
The ASTAA Project

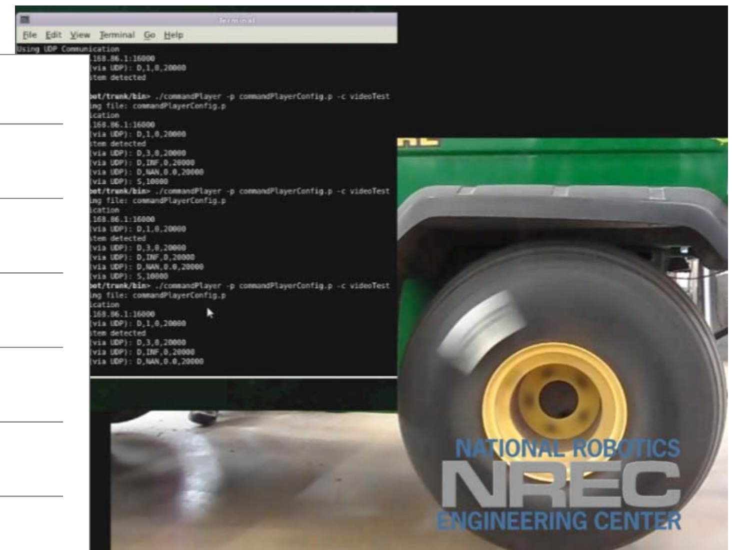
- Automated Stress Testing of Autonomy Architectures
  - Three-year project sponsored by the Test Resource Management Center within the Office of the Secretary of Defense
  - The project continues through September 2014

- Project goals:
  - Use automatic software stress-testing to uncover safety problems in unmanned systems that **wouldn't otherwise be found during system testing**
  - Implement testing tools that interface with software components in an unobtrusive way

Do Robots Have Robustness Problems? (yes)

- Mature (6 years old) “RECBot” vehicle tested with initial tool set
  - No access to source code or design details; just interface specification
  - **ASTAA elicited a speed-limit violation**

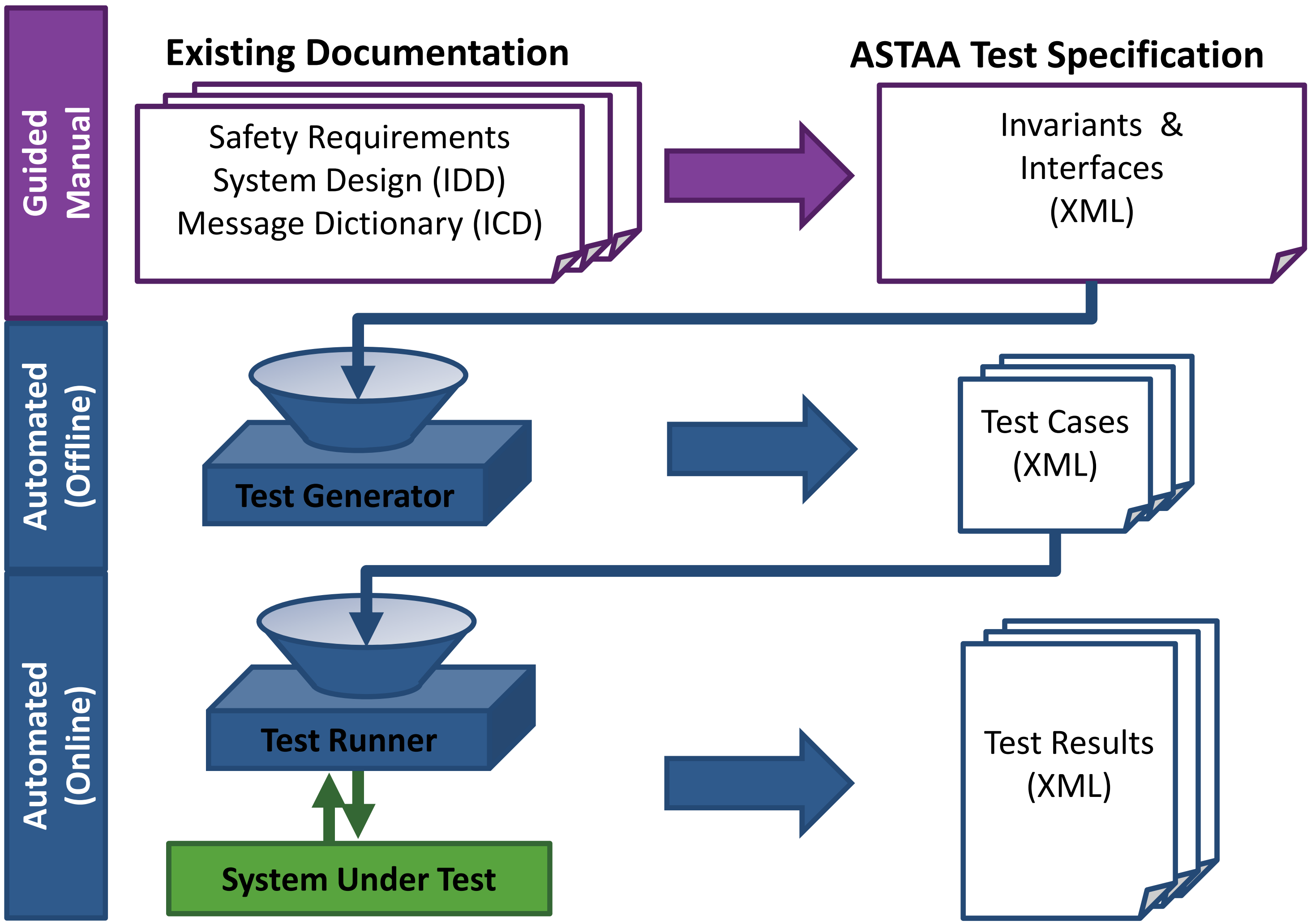




Speed-limit violation occurred when exceptional input sent as speed command

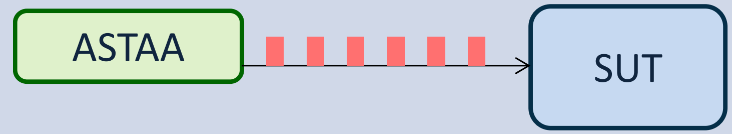
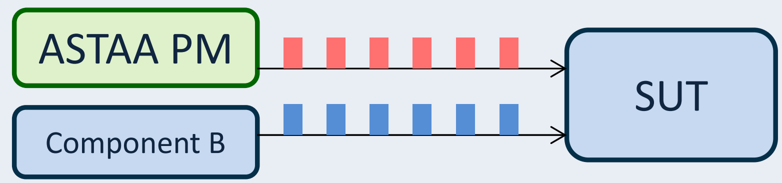
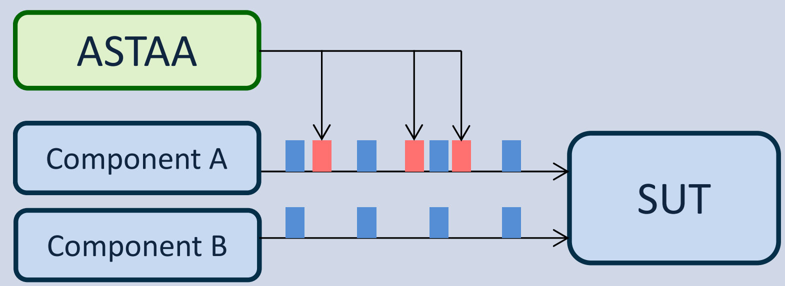
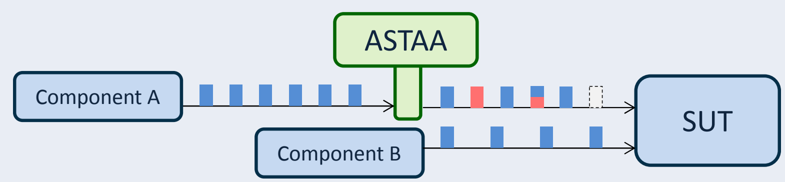
Distribution Statement A – Approved for public release; distribution is unlimited. NAVAIR Public Affairs Office tracking number 2013-74, NREC internal case number STAA-2012-10-23

ASTAA Workflow



DISTRIBUTION A – NREC case number STAA-2013-10-02

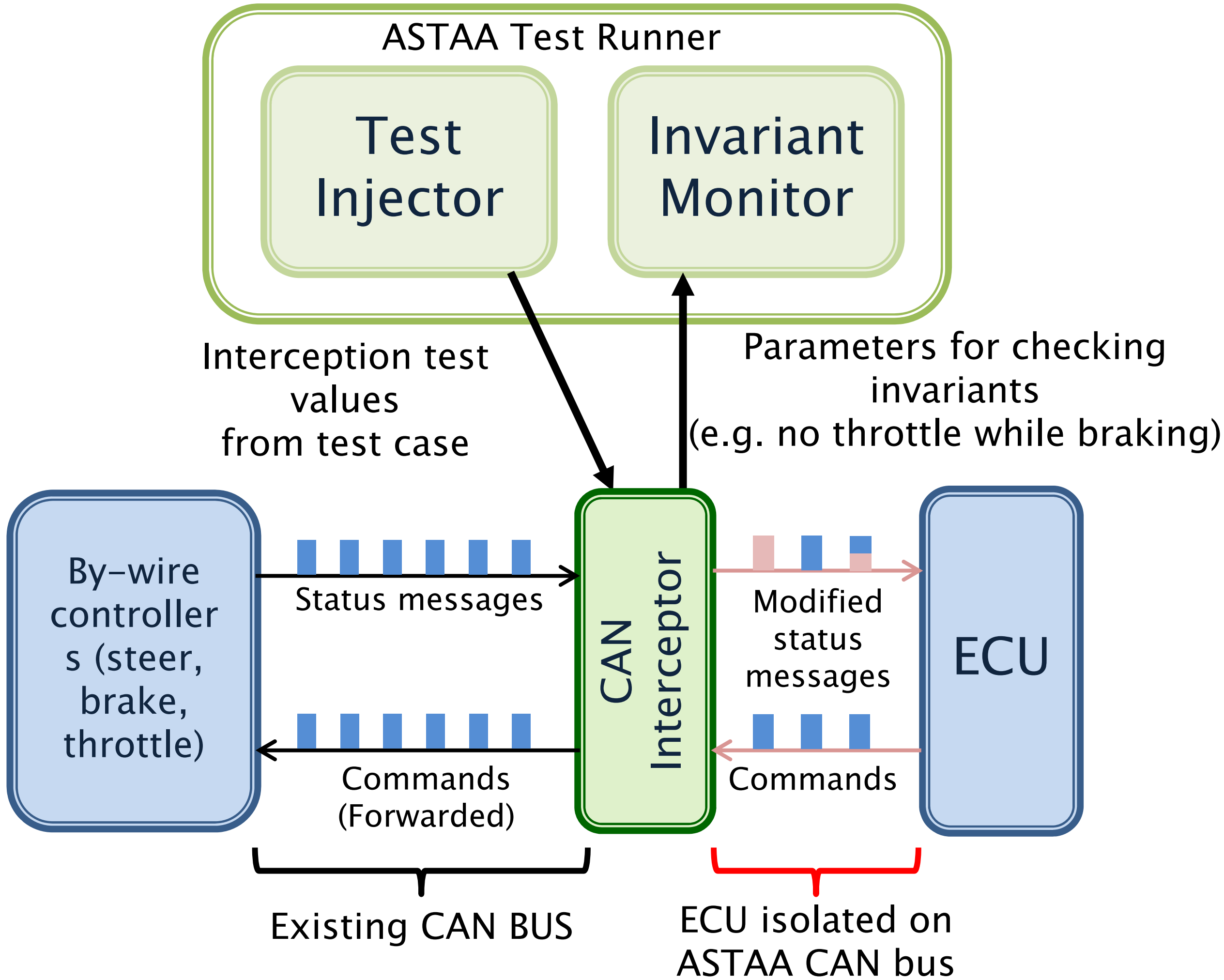
Methods of test execution

Injection	
Injection with log replay or running component	
Interleaving during injection	
Interception	

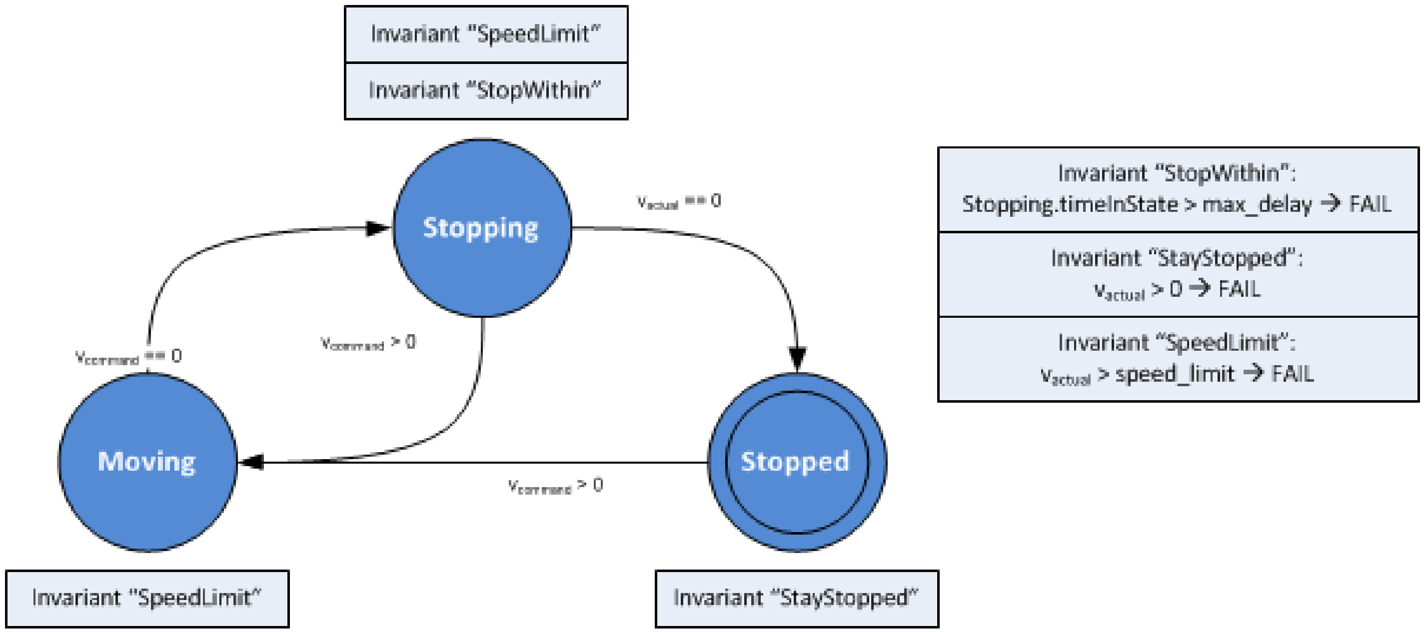
Example: CAN / J1939 interception

In this example:

- CAN Interceptor
  - Isolates actuators from ECU by splitting the CAN bus
  - Modifies J1939 status messages from by-wire controllers before forwarding to ECU
  - Reads messages for invariant evaluation
- ASTAA Test Runner
  - Instructs CAN interceptor about how to modify incoming CAN messages
  - Monitors invariants

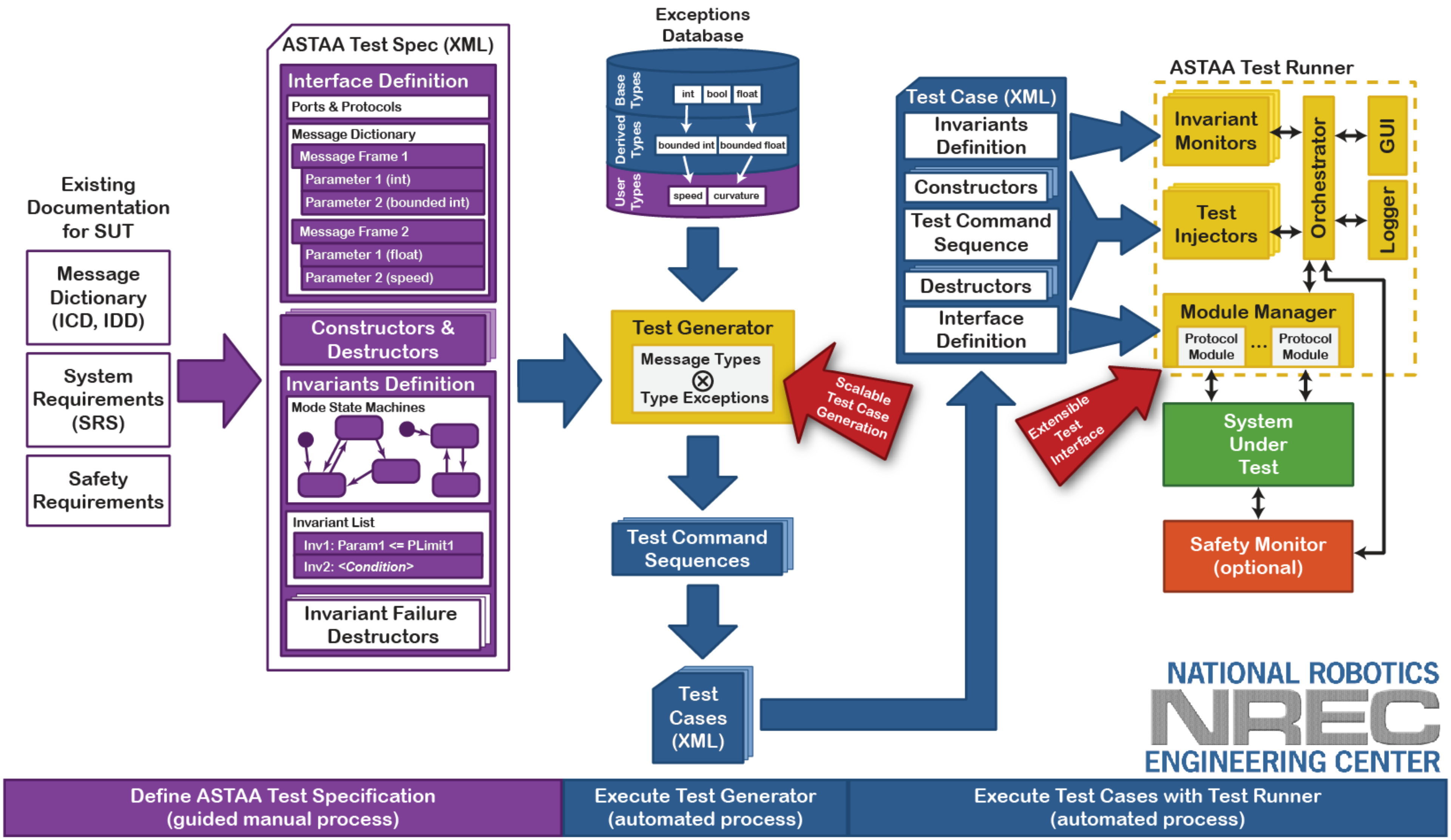


Architecture Details: Invariant Monitor



- An invariant is an expression involving SUT state that takes the form of a guard and predicate ("FAIL" or "WARN")
- State machines track the system's state
  - Transition guards are inputs from the SUT
- Each state activates potentially different invariants

Test Specification and Execution Overview



Types of components tested so far

- Communications: Message serialization and routing
- Control: motion control, I/O
- Perception: terrain perception, terrain classification, obstacle detection, map building
- Planning: path tracking, motion planning, obstacle avoidance

- **Stress testing finds bugs in autonomy software**
  - **Over 50 vulnerabilities have been found in over twenty systems tested on our project so far**

Root causes of robustness vulnerabilities include…

- **Improper handling of floating-point numbers**
  - Failure to handle exceptional values (e.g., NaN, Inf)
  - Normalization of floating-point angles
- **Array indexing and allocation**
  - E.g., images, point clouds, evidence grids
  - Segmentation faults due to arrays that are too small
  - Many forms of buffer overflow, especially dealing with complex data types
  - Large arrays and memory exhaustion
- **Time**
  - Time flowing backwards, jumps
  - Not rejecting stale data
- **Problems handling dynamic state**
  - E.g., lists of perceived objects or command trajectories
  - Race conditions permit improper insertion or removal of items
  - Vulnerabilities in garbage collection allow memory to be exhausted or execution to be slowed down
- **Assertions that have not been disabled**

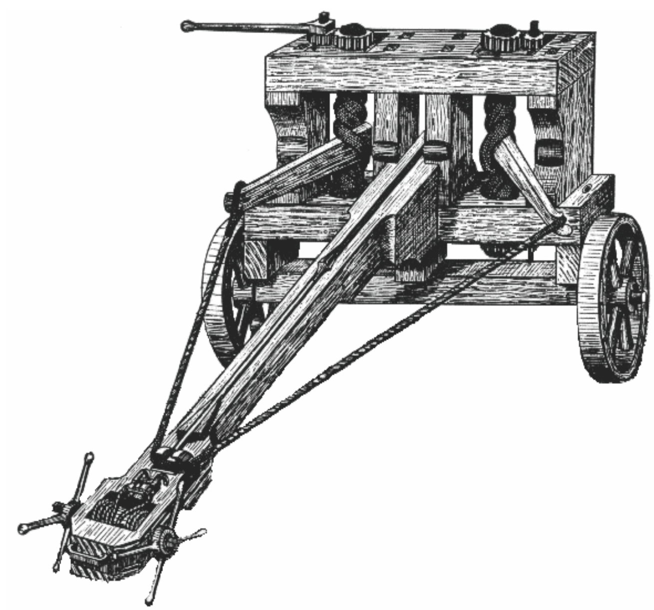
The Ballista/ASTAA Team

Ballista Robustness Testing (1997 – 2002)

Safety and Security for Embedded Systems (1997 –)

System Safety for Autonomous Robots (2008 –)

Automated Stress Testing of Autonomy Architectures (2011 –)



A Ballista is an ancient siege weapon for hurling large projectiles at fortified defenses.

Making "Easier" Systems Safe

- Elevators
  - Building codes describe required mechanisms
  - Electromechanical safeties (avoid trusting SW)

- Rail systems
  - Dual redundant hardware protection systems
  - Rigorously developed software EN-50126/8/9
    - Customers typically require these standards
    - "Safety net" architecture minimizes critical SW
  - Fail-stop approach – shut down if unsafe

[Koopman 2014, Transportation CPS Workshop]

Why HW Safety Is Difficult

- "Safe" might be 1e-9/hr catastrophic failures
  - (It is easy to argue cars must be safer than that)
  - Single fatalities at perhaps 1e-7/hr (probably less)
  - Simplex hardware tends to fail at 1e-5 to 1e-6/hr
    - Cosmic rays result in bit flips (yes, really!)
    - Other things go wrong at about this rate
  - Thus, need **redundancy** to be safe
    - No single point failure end-to-end in the system
    - Takes some effort to get redundant components to properly synch.

- **Infeasible to test** to 1e-9/hr
  - Need testing time 3x-10x longer than failure rate

[Koopman 2014, Transportation CPS Workshop]

Making "Harder" Systems Safe

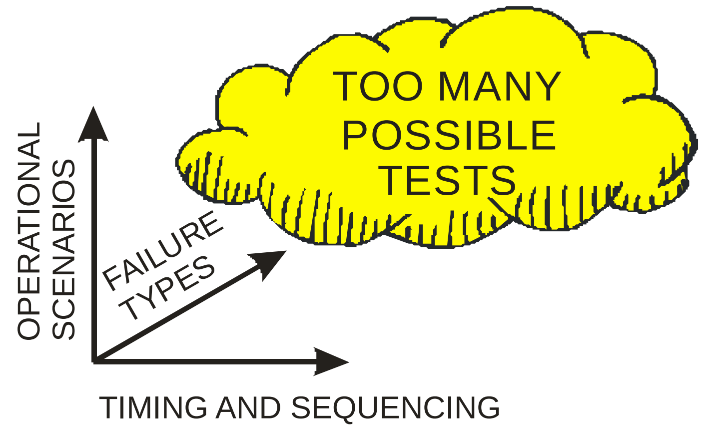
- Aviation
  - Do-178 and other FAA standards
  - Federal certifying agency (FAA)
    - Testing + examination of how system is designed
  - Fail operational; significant redundancy

- Automotive
  - NHTSA does not proactively certify safety
    - FMVSS don't really address SW safety
  - Some redundancy; tough cost constraints
    - Steering & brakes must fail (partially) operational
  - MISRA Guidelines ➔ ISO 26262 safety standard
    - But neither is really intended to cover autonomous vehicles

[Koopman 2014, Transportation CPS Workshop]

Why SW Safety Is Difficult

- Testing does not make software safe!
  - You can't test all SW corner cases
  - Proving correctness is not enough for safety either
    - How do you know your requirements are correct?
    - Have you proven correctness under all fault conditions?
- Software safety requires process in addition to testing
  - Follow standards (e.g., ISO 26262)
    - List of practices based on SW criticality
    - Ensure development process quality
  - Testing checks you really did it right
    - Testing is not "debugging" – test for absence of bugs
  - Adaptive/robot software can go beyond existing SW safety



[Koopman 2014, Transportation CPS Workshop]

The World Is Full Of Unexpected Situations…

Extreme contrast

No lane infrastructure

Poor visibility

Unusual obstacles

Construction

Water (note that it appears flat!)

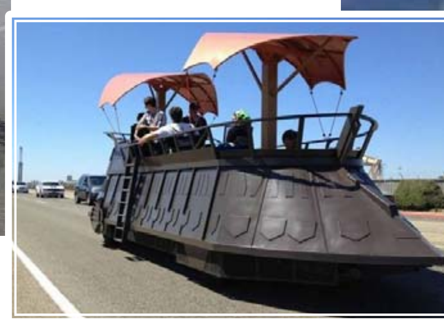
So just getting all the obvious cases covered is challenging

[Wagner 2014]

NOBODY Has Seen It ALL!





[Koopman14]

Autonomy Validation Challenges

- Specifying safety
  - Artfully select subset of functionality to equal safety
  - Need a realistic role for human operator
- Unconstrained environments
  - Uncontrolled, unpredictable urban roadways
  - Can inductive-based algorithms cover enough corner cases?
- Trusting validation
  - How do you know you are really safe?
  - How do you know someone else's system is really safe when you cooperating with it?

[Koopman 2014, Transportation CPS Workshop]

Questions?