Assured Autonomy through Combinatorial Methods

6th IEEE Conference on Dependable and Secure Computing

Tampa, FL USA 7 – 9 November, 2023

Rick Kuhn
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
kuhn@nist.gov

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY U.S. DEPARTMENT OF COMMERCE

Outline

- Why current safety-critical testing isn't suitable
- Assurance based on input space coverage,
- Explainable AI as part of validation, and
- Transfer learning

Some problems in assured autonomy,
and potential solutions

What is NIST and why are we doing this?

- US Government agency, which supports US industry through developing better measurement and test methods
- 3,000 scientists, engineers, and staff including 4 Nobel laureates
- Broad involvement with industry and academia

What are interaction faults?

- NIST studied software failures in 15 years of FDA medical device recall data
- What causes software failures?
    - logic errors? calculation errors? inadequate input checking? interaction faults? Etc.

Interaction faults: e.g., failure occurs if

pressure < 10 & volume > 300

(interaction between 2 factors)

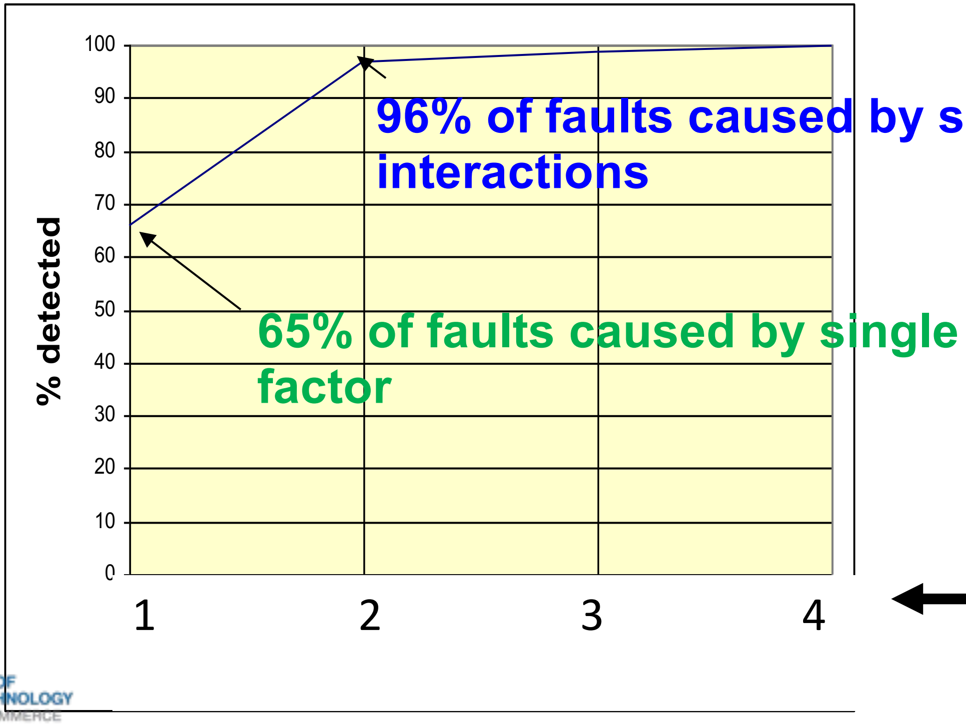
So this is a **2-way interaction**

=> testing all pairs of values can find this fault

How are interaction faults distributed?

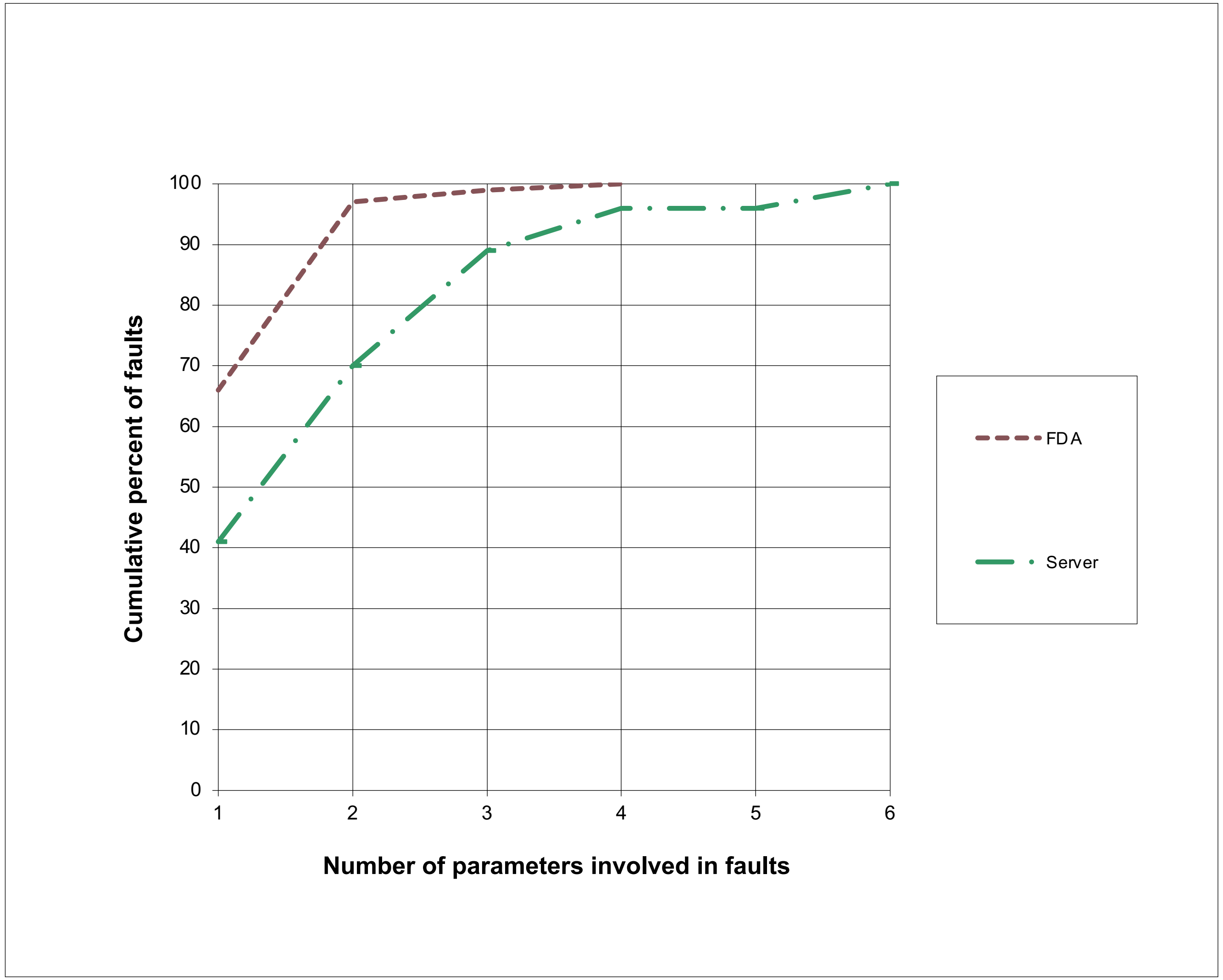
- Interactions   e.g., failure occurs if
  - pressure < 10   (1-way interaction)
  - pressure < 10 & volume > 300   (2-way interaction)
  - pressure < 10 & volume > 300 & velocity = 5   (3-way interaction)

- Surprisingly, no one had looked at interactions > 2-way before



Number of factors involved

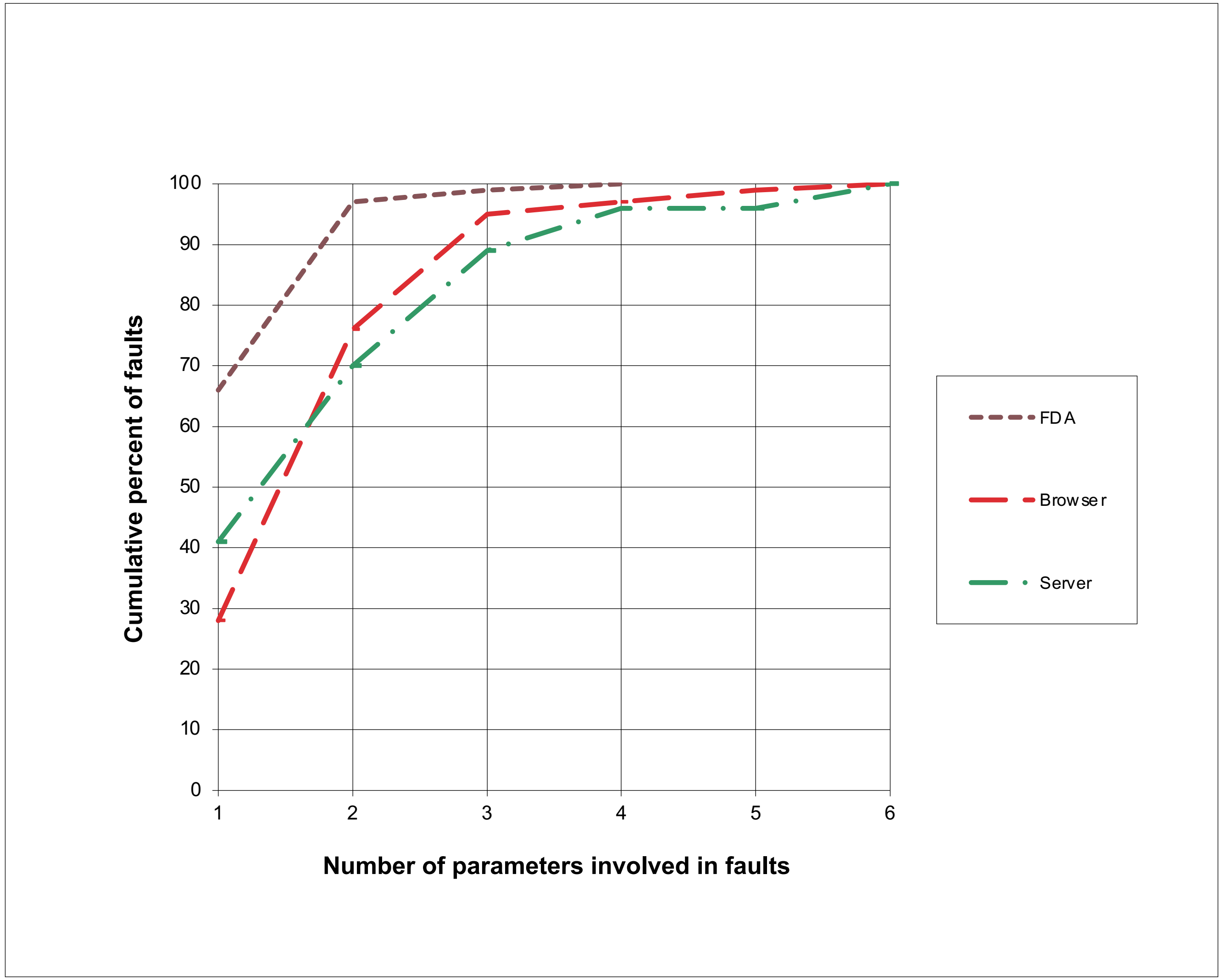
Server



These faults more complex than medical device software!!

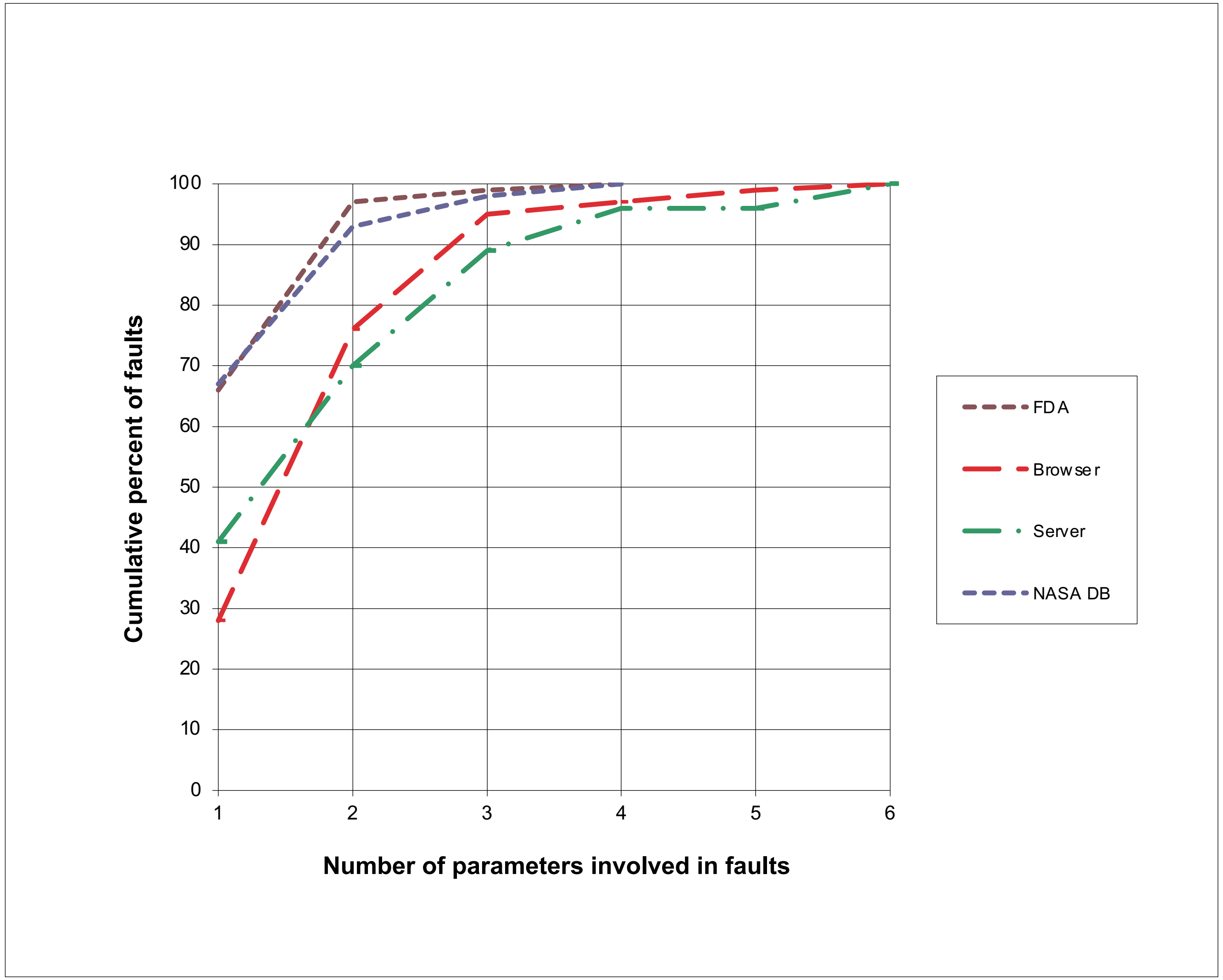
Why?

Browser



Curves appear to be similar across a variety of application domains.

NASA distributed database

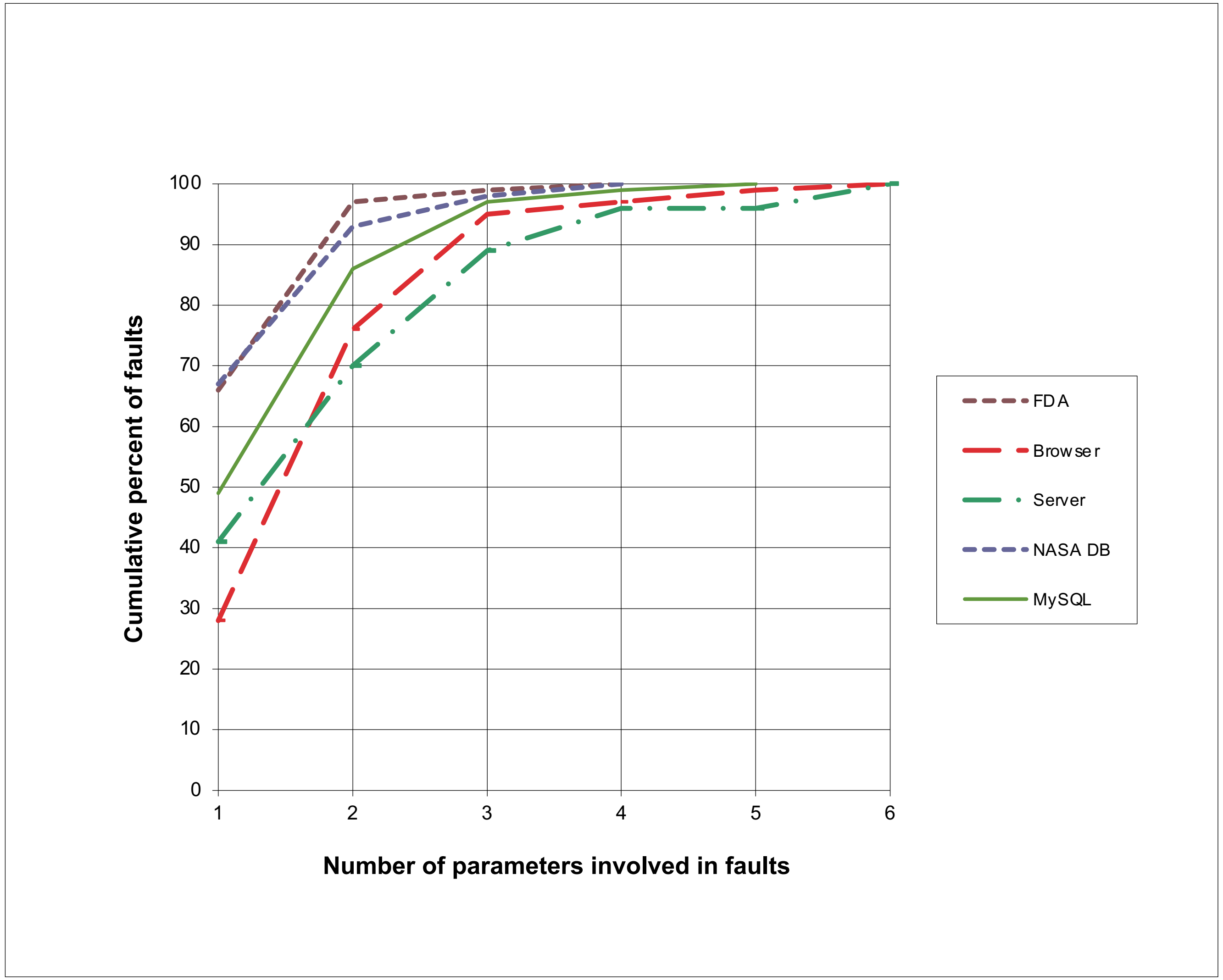


Note: <u>initial testing</u>

but ….

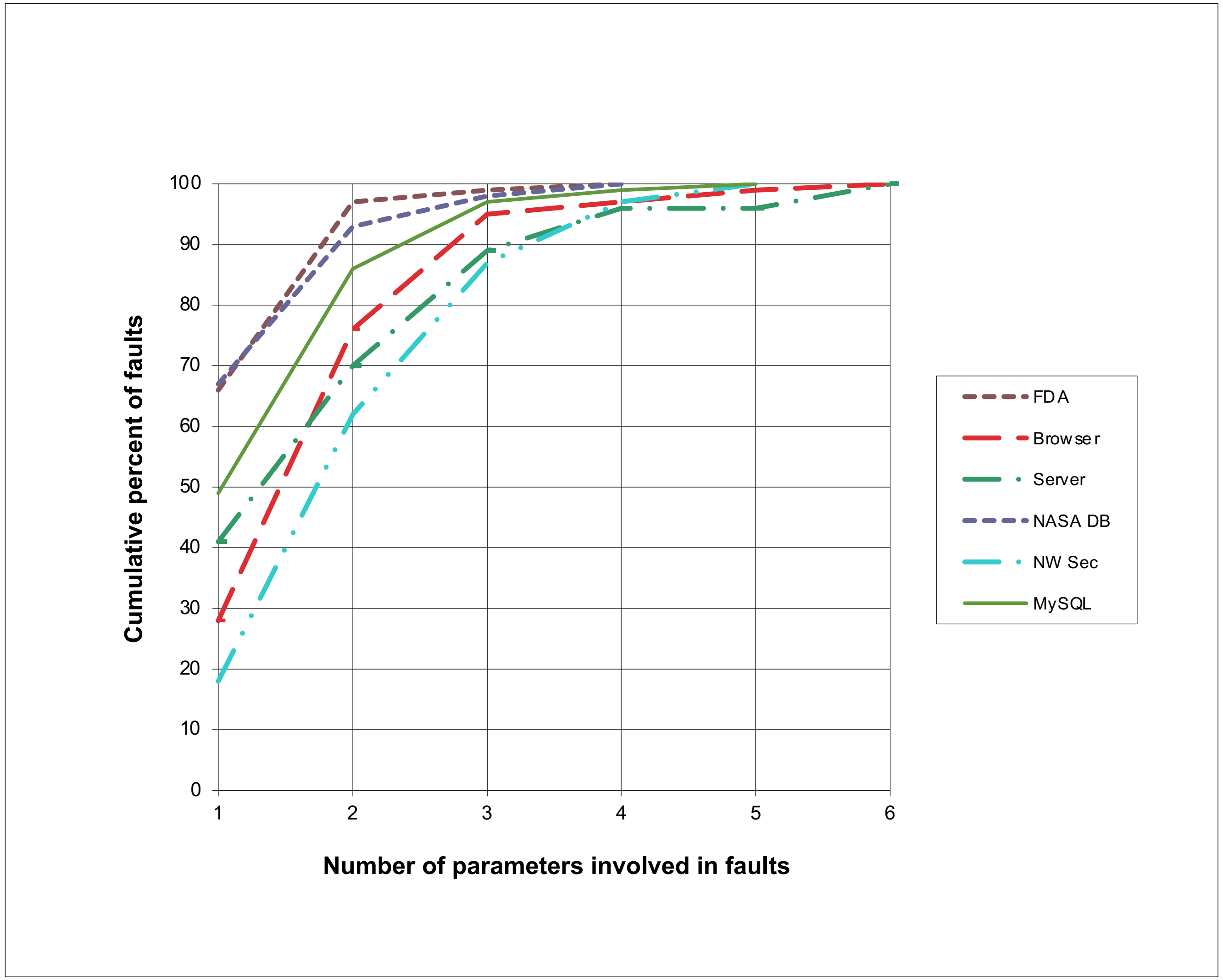
Fault profile better than medical devices!

MySQL



National Institute of Standards and Technology

TCP/IP



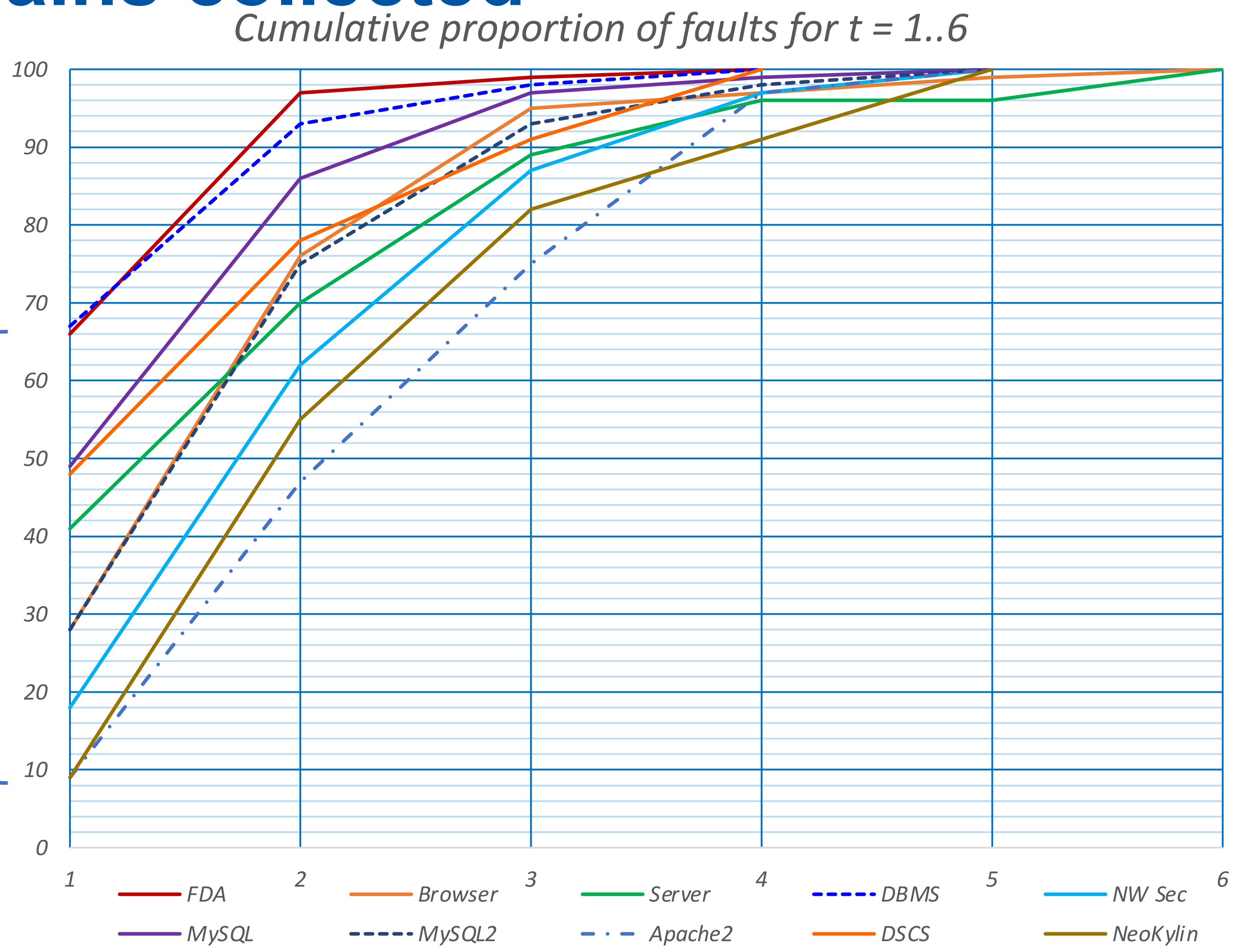
NIST
National Institute of
Standards and Technology

Various domains collected

Wide variation in percent of failures caused by single factor

Variability decreases as number of factors increases

More testing or users
=> harder to find errors, fewer single factor failures



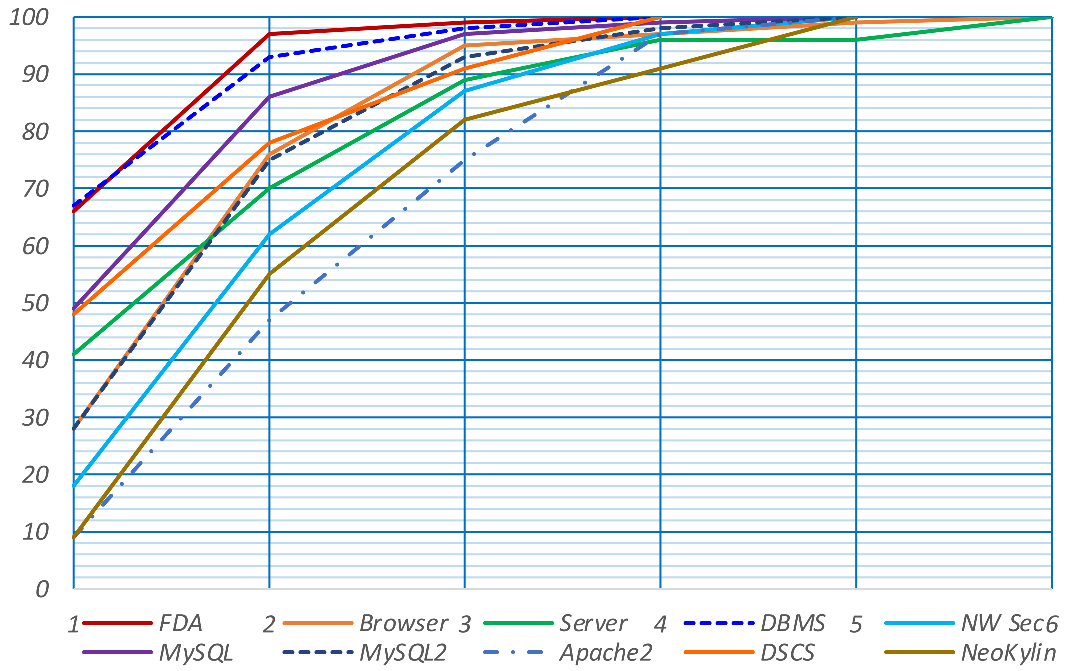
- Number of factors involved in failures is <u>small</u>
- No failure involving more than 6 variables has been seen

National Institute of Standards and Technology

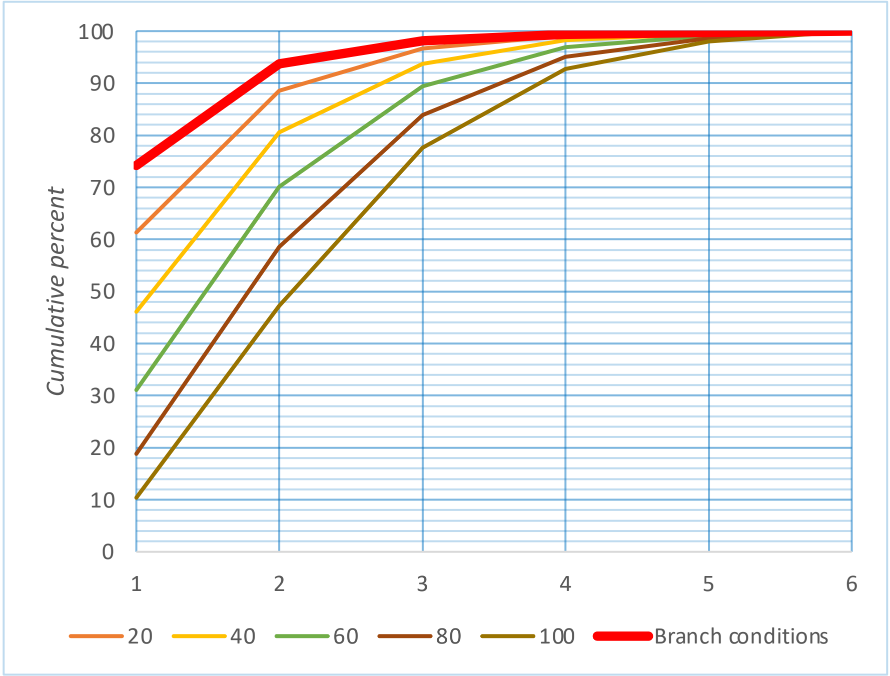
Fault distribution as testing progresses

- for testing cycles, starting from distribution of branch conditions; curve moves down and to the right with more inputs/usage; close to empirical data

Empirical data



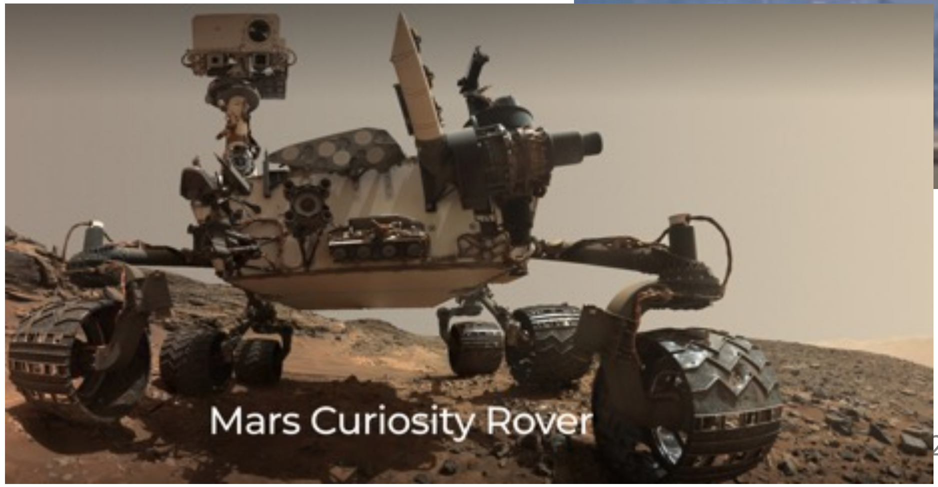
Model



Empirical data

Model

How is all this related to autonomous systems?



(Slide from Darryl Ahner, US Air Force Institute of Technology)

Defense Science Board Study

STAT T&E COE: Scientia Prudentia et Valor

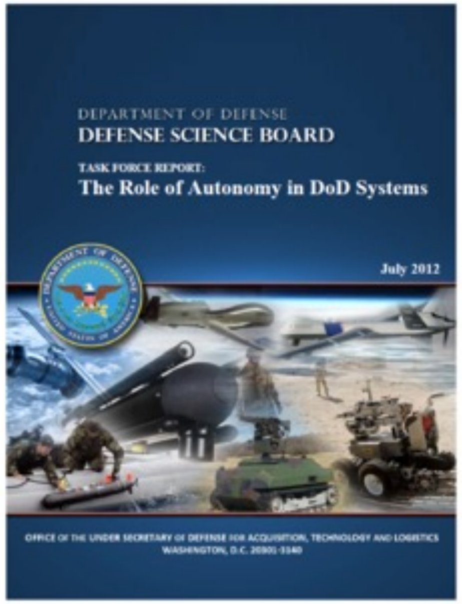
DSB 2012 The Role of Autonomy in DoD Systems Study recommends:

"USD(AT&L) to create developmental and operational T&E techniques that focus on the unique challenges of autonomy (to include developing operational training techniques that explicitly build trust in autonomous systems)."

Recommendation:

USD(AT&L) establish developmental and operational T&E techniques that focus on the unique challenges of autonomy

- Coping with the difficulty of enumerating all conditions and non-deterministic responses
- Basis for system decisions often not apparent to user
- Measuring trust that the autonomous system will interact with its human supervisor as intended
- Leverage the benefits of robust simulation



AFIT

NIST National Institute of Standards and Technology

Software safety assurance is already *very* expensive

Consumer level software cost: about 50% code development, 50% testing and verification

For aviation life-critical, 12% code development, 88% testing and verification (Software is about 30% of cost for new civilian aircraft, higher for military)

Autonomy makes the problem even harder!

V&V cost and Certification

For FAA compliant DO-178B Level A software, the industry usually spends 7 times as much on verification (reviews, analysis, test). So that's about 12% for development and 88% for verification.

Level B reduces the verification cost by approximately 15%. The mix is then 25% development, 75% verification.

Randall Fulton
FAA Designated Engineering Representative
(private email to L. Markosian, July 2008)

13 April 2010 NFM 2010

Autonomy makes the problem even more expensive!

DARPA Assurance for Autonomous Systems is Hard

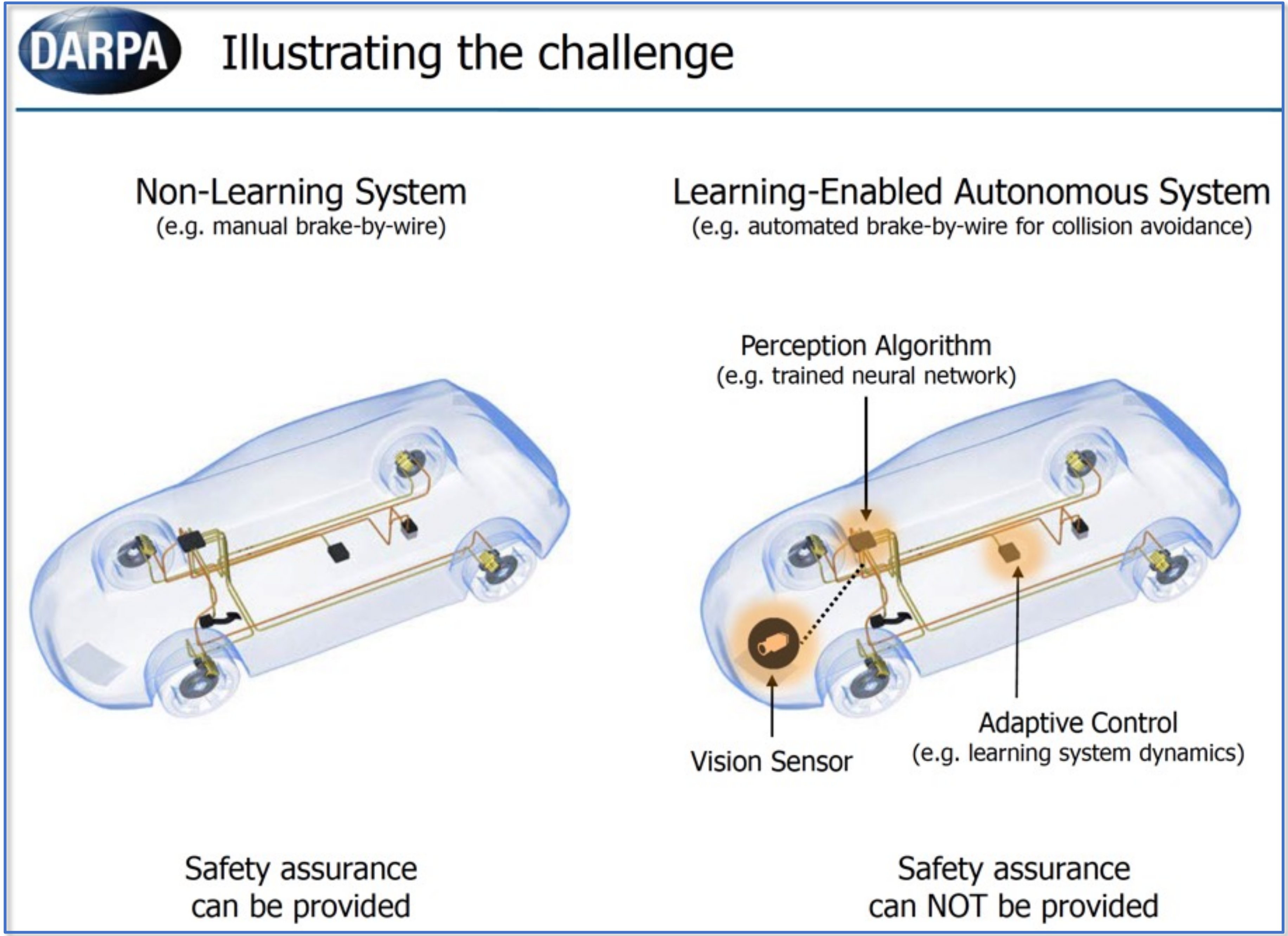
Traditional testing will require exorbitant time and money:
11B miles, 500 years, $6B
- Driving to Safety, RAND Corp. Report, 2016

Table 1. Examples of Miles and Years Needed to Demonstrate Autonomous Vehicle Reliability

Statistical Question	Benchmark Failure Rate		
How many miles (years[a]) would autonomous vehicles have to be driven...	**(A) 1.09 fatalities per 100 million miles?**	**(B) 77 reported injuries per 100 million miles?**	**(C) 190 reported crashes per 100 million miles?**
(1) without failure to demonstrate with 95% confidence that their failure rate is at most...	275 million miles (12.5 years)	3.9 million miles (2 months)	1.6 million miles (1 month)
(2) to demonstrate with 95% confidence their failure rate to within 20% of the true rate of...	8.8 billion miles (400 years)	125 million miles (5.7 years)	51 million miles (2.3 years)
(3) to demonstrate with 95% confidence and 80% power that their failure rate is 20% better than the human driver failure rate of...	11 billion miles (500 years)	161 million miles (7.3 years)	65 million miles (3 years)

[a] We assess the time it would take to compete the requisite miles with a fleet of 100 autonomous vehicles (larger than any known existing fleet) driving 24 hours a day, 365 days a year, at an average speed of 25 miles per hour.

NIST National Institute of Standards and Technology



Conventional safety assurance methods don’t work well for this

NIST
National Institute of Standards and Technology

Why can’t we use same processes as other safety-critical software ?

- Conventional critical software testing is based on structural coverage – ensuring that conditions, decisions, paths are covered in testing

- Life-critical aviation software requires MCDC testing, white-box criterion that doesn’t fit neural nets and other black-box methods where <u>input</u> is what matters

High level DARPA Assured Autonomy Goals

- Increase scalability of design-time assurance
  - What is the baseline capability of the proposed methods, in terms of the hybrid state-space and number and complexity of learning-enabled components
  - How do you plan to scale up by an order of magnitude?
  - How will you characterize the tradeoffs between fidelity of your modeling abstractions and scalability of the verification approach.
- Reduce overhead of operation-time assurance
  - What is the baseline overhead of the operation-time assurance monitoring techniques?
  - How do you plan to minimize it to be below 10% of the nominal system resource utilization?
- Scale up dynamic assurance
  - What is the size and scale of dynamic assurance case that can be developed and dynamically evaluated with your tools?
- Reduce trials to assurance
  - How will you approach quantifiably reduce the need for statistical testing?

Scalability

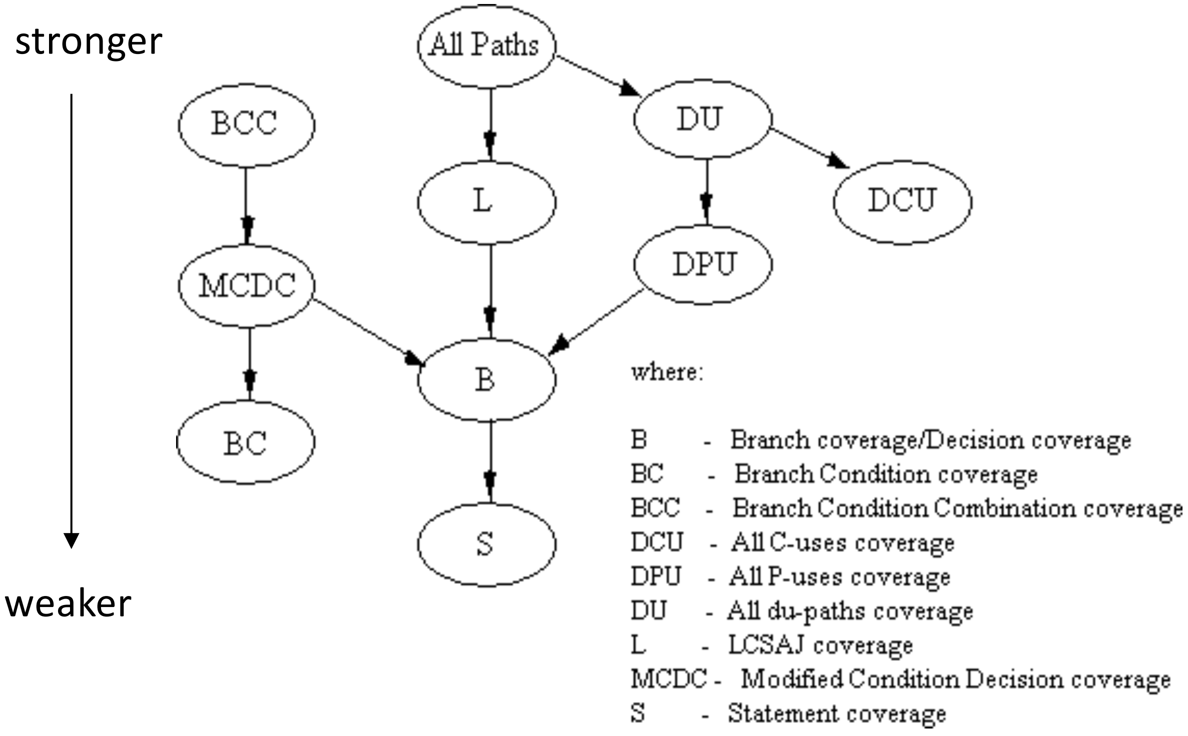
Cost

Resources

Time

National Institute of Standards and Technology

Code coverage works well - for conventional software



Subsumption relationships of structural coverage criteria

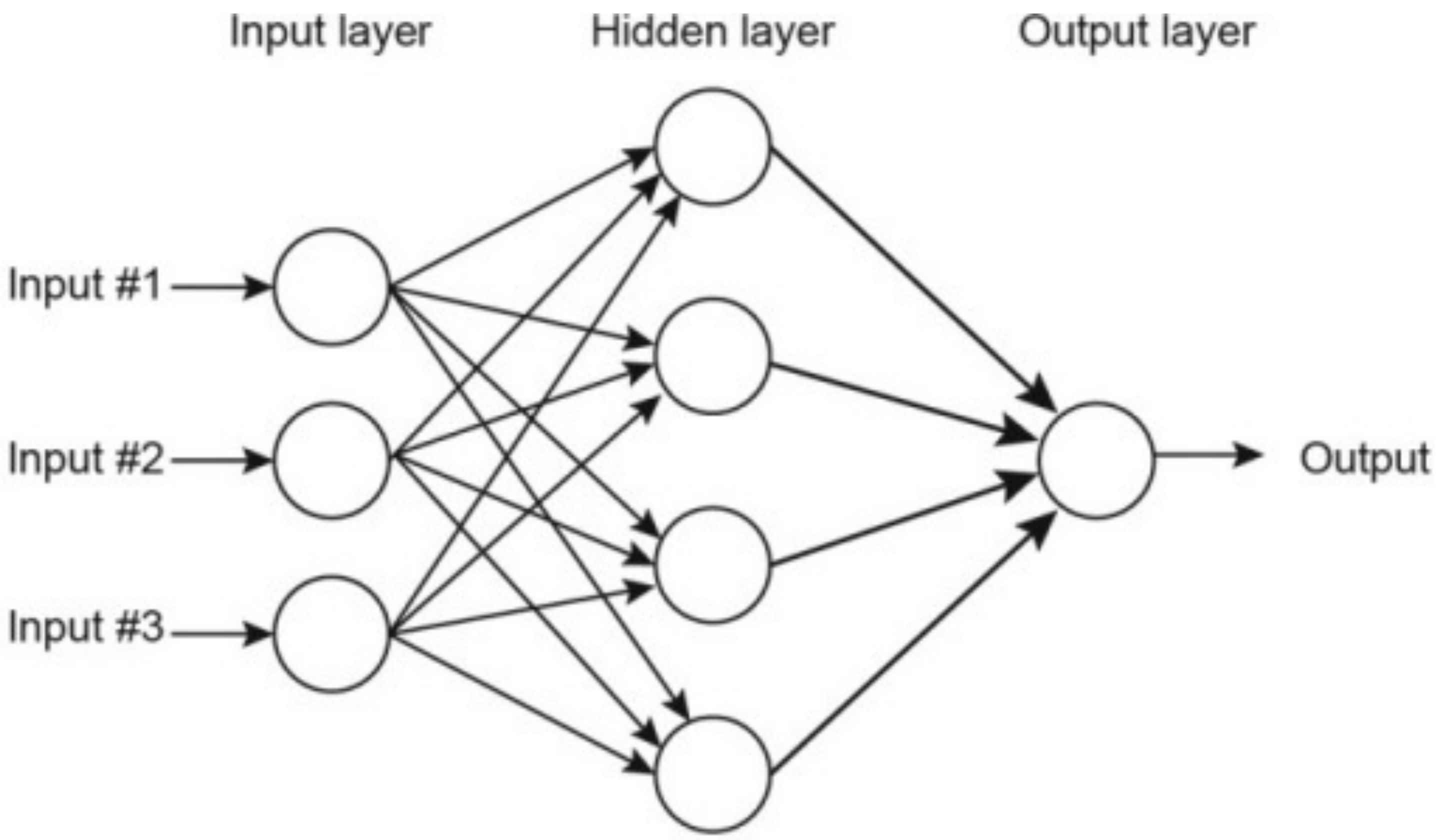
- Test coverage has traditionally been defined using graph-based structural coverage criteria:
  - statement (weak)
  - branch (better)
  - etc.
- Based on paths through the code

- We may have perfect structural coverage of code, but what does that tell us about response to rare inputs?

- What if the code is always the same, and only the inputs matter?

Can we use code coverage for machine learning?

- Much of AI/ML depends on various neural nets
- Algorithm and code stays the same
- Connections and weights vary
- Behavior changes depending on inputs used in training



NIST National Institute of Standards and Technology

Input space coverage is needed

- Gold standard of assurance and verification of life-critical software is not suitable for much of new life-critical autonomy software

- We can measure "neuron coverage", but indirect measure and not clear how closely related to accuracy and ability to correctly process all of the input space

- Measure the input space directly

- Then see if the AI system handles all of it correctly

Outline

- Why current safety-critical assurance isn't suitable
- Assurance based on input space coverage
- Explainable AI as part of validation, and
- Transfer learning

Major DoD investment in assured autonomy

"The notion that autonomous systems can be fully tested is becoming increasingly infeasible as higher levels of self governing systems become a reality…*the standard practice of testing all possible states and all ranges of inputs to the system becomes an unachievable goal*. Existing TEVV methods are, by themselves, insufficient for TEVV of autonomous systems; therefore *a fundamental change is needed in how we validate and verify these systems*."
- OSD TEV&V Strategy Report, May 2015

(Note that "testing all possible states and all ranges of inputs" was already unachievable, but the point holds.)

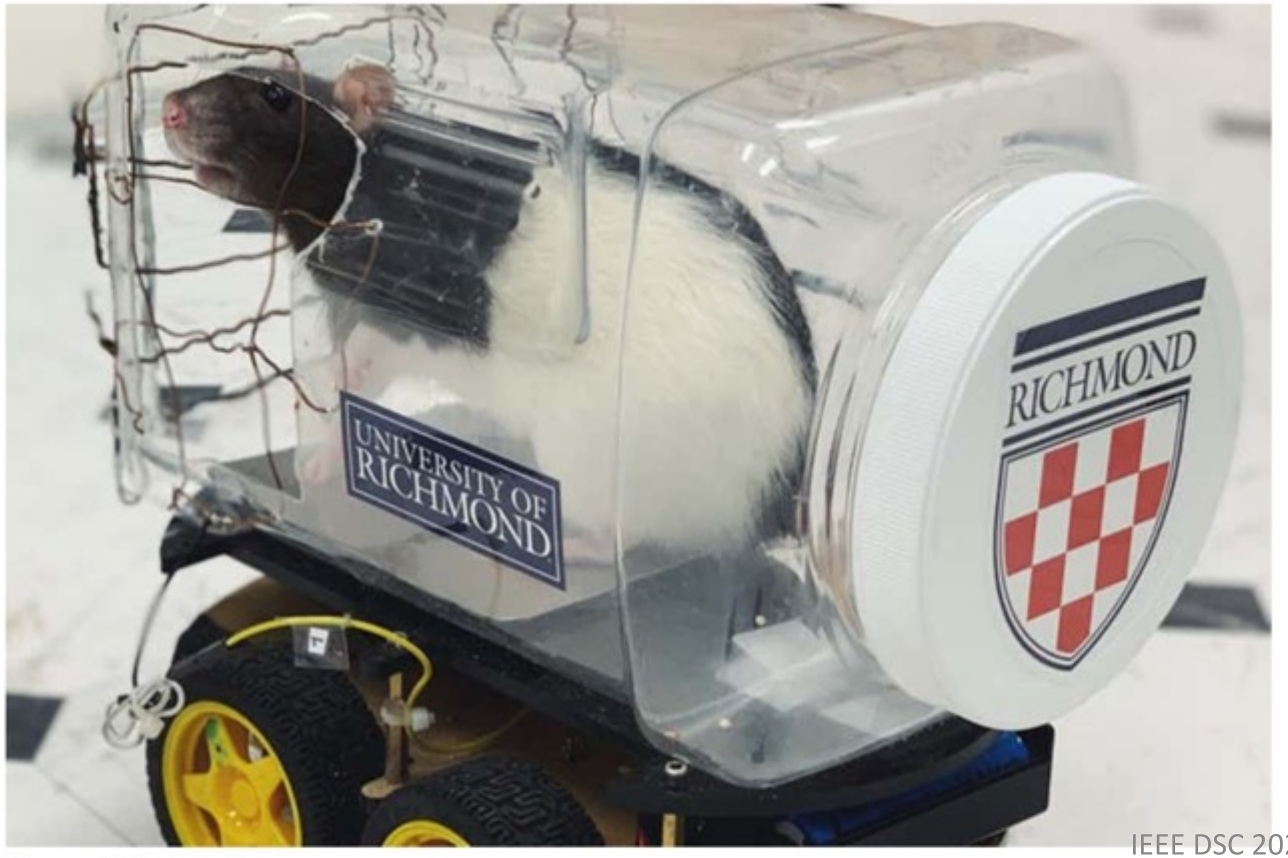
National Institute of Standards and Technology

NewScientist

Scientists have trained rats to drive tiny cars to collect food

LIFE 22 October 2019

By Alice Klein



It doesn't take much intelligence to drive a car. Even rats can do it!

But can they do it under all kinds of conditions ?

The problem is harder outside of a constrained environment

NIST
National Institute of Standards and Technology

Things get tricky as the scene becomes complex

- Multiple conditions involved in accidents
  - "The camera failed to recognize the white truck against a bright sky" (2 factors)
  - "The sensors failed to pick up street signs, lane markings, and even pedestrians due to the angle of the car shifting in rain and the direction of the sun" (3 factors)
- We need to understand what combinations of conditions are included in testing

NIST
National Institute of Standards and Technology

How can we measure interaction fault detection capability?

rows of input

a	b	c	d
0	0	0	0
0	1	1	0
1	0	0	1
0	1	1	1

19 combinations included in test set

Vars	Combination values	Coverage
a b	00, 01, 10	.75
a c	00, 01, 10	.75
a d	00, 01, 11	.75
b c	00, 11	.50
b d	00, 01, 10, 11	1.0
c d	00, 01, 10, 11	1.0

100% coverage of 33% of combinations
75% coverage of half of combinations
50% coverage of 16% of combinations

Kuhn, D. R., Mendoza, I. D., Kacker, R. N., & Lei, Y. (2013). Combinatorial coverage measurement concepts and applications. *2013 IEEE Sixth Intl Conference on Software Testing, Verification and Validation Workshops*

National Institute of Standards and Technology

Vars	Combination values	Coverage
a b	00, 01, 10	.75
a c	00, 01, 10	.75
a d	00, 01, 11	.75
b c	00, 11	.50
b d	00, 01, 10, 11	1.0
c d	00, 01, 10, 11	1.0

Total possible 2-way combinations $= 2^2 \binom{4}{2} = 24$

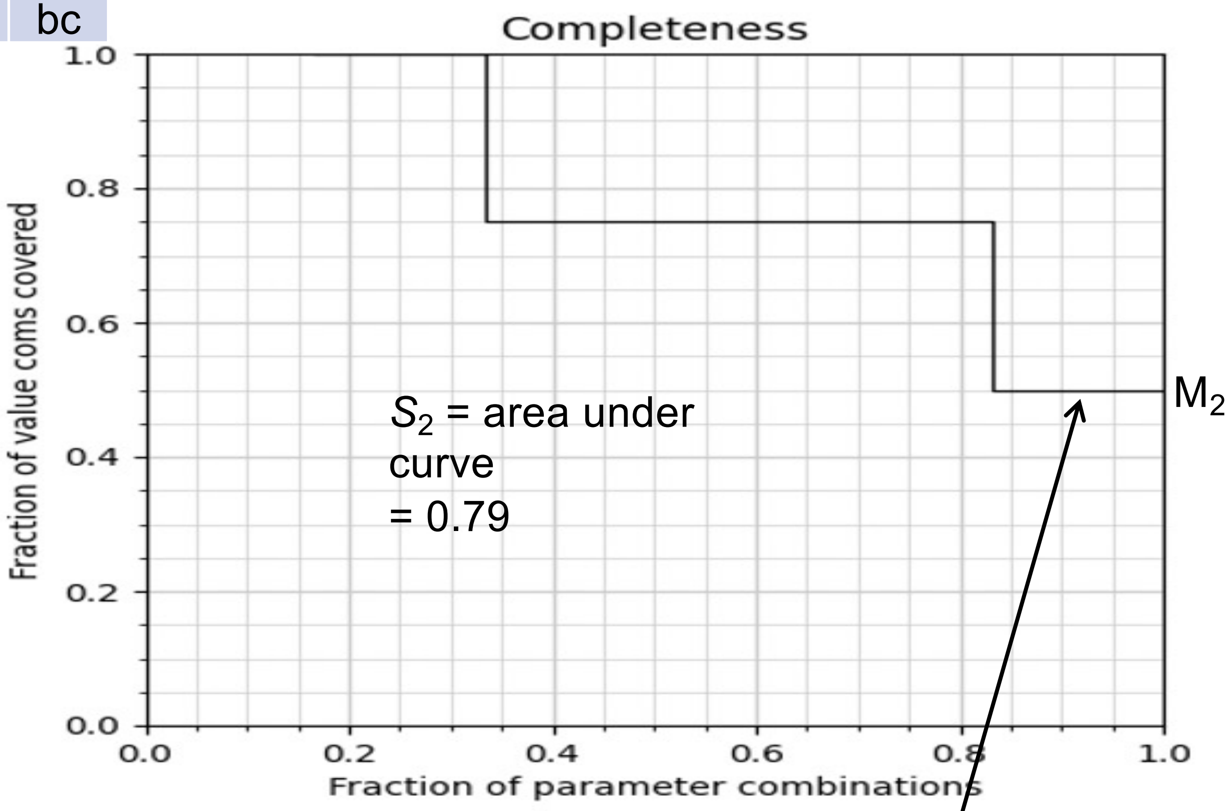
S_2 = fraction of 2-way combinations covered = 19/24 = 0.79

Rearranging the table:

1.00	00	00				
.75	01	01	00	00	00	
.50	10	10	01	01	01	00
.25	11	11	10	10	11	11
	bd	cd	ab	ac	ad	bc

Graphing Coverage Measurement

1.00	00	00				
.75	01	01	00	00	00	
.50	10	10	01	01	01	00
.25	11	11	10	10	11	11
	bd	cd	ab	ac	ad	bc



100% coverage of .33 of combinations
75% coverage of .50 of combinations
50% coverage of .16 of combinations

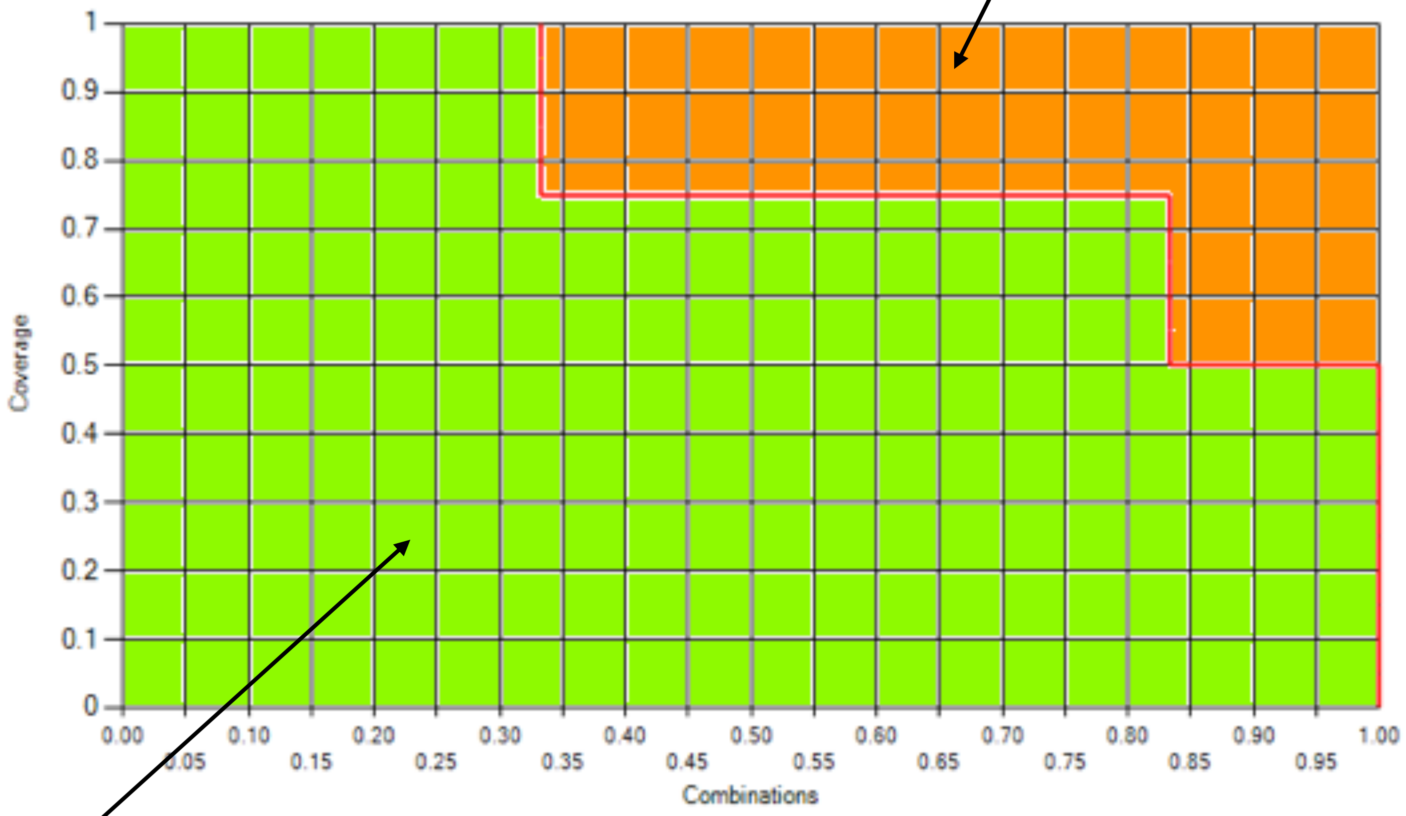
Bottom line:
All combinations covered to at least .50

National Institute of Standards and Technology

What else does this chart show?

$1 - S_t$ = **Untested combinations**

(look for problems here)



S_t = Tested combinations => code works for these

How is input combination coverage related to structural coverage?

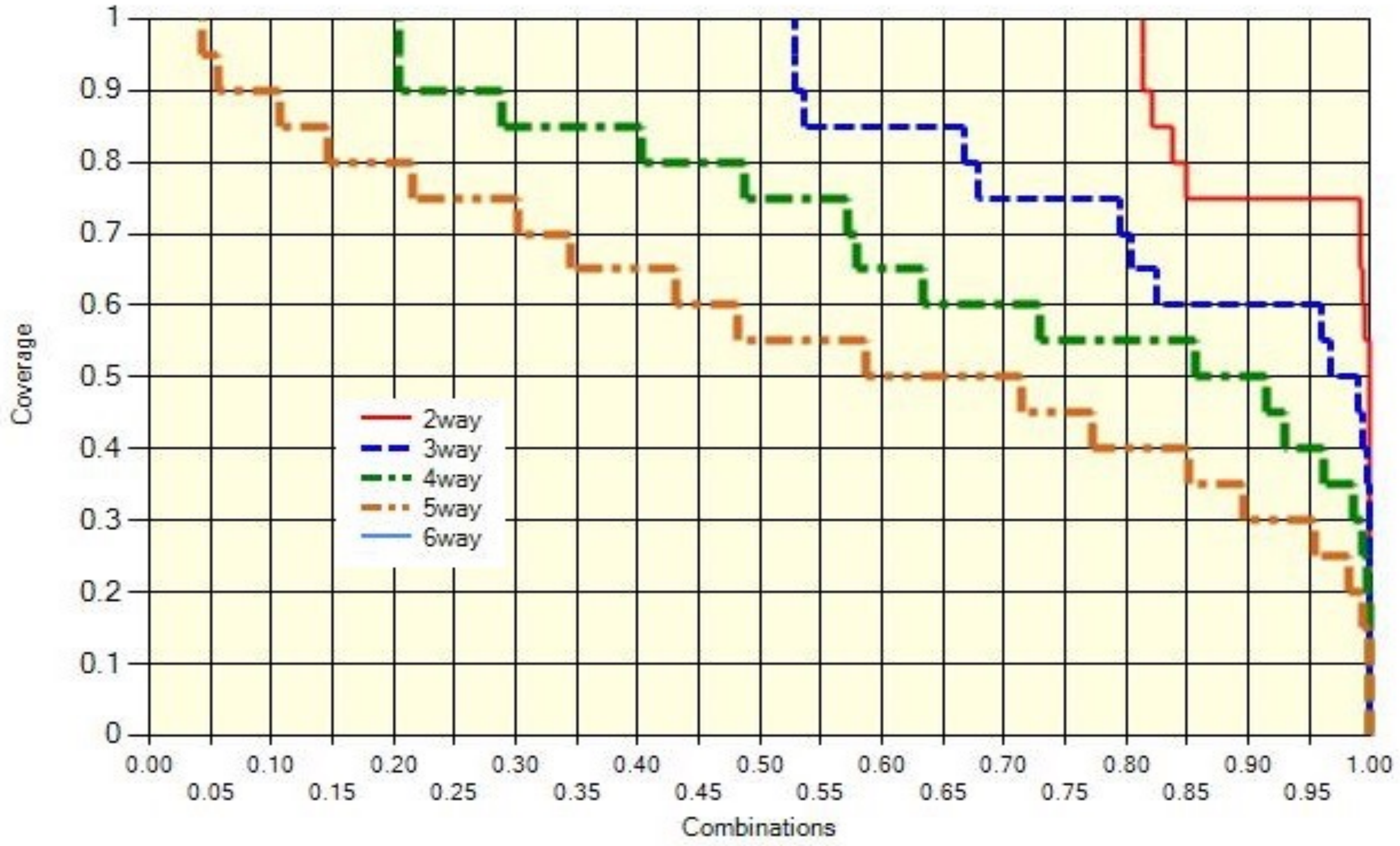
- *Branch coverage condition theorem*
- Where M_t is the proportion of input combinations covered, and
- B_t is the minimum proportion of input combinations triggering a code branch,
- then 100% branch coverage will be achieved if

$$M_t + B_t > 1$$

- (Recall that branch coverage subsumes statement coverage)

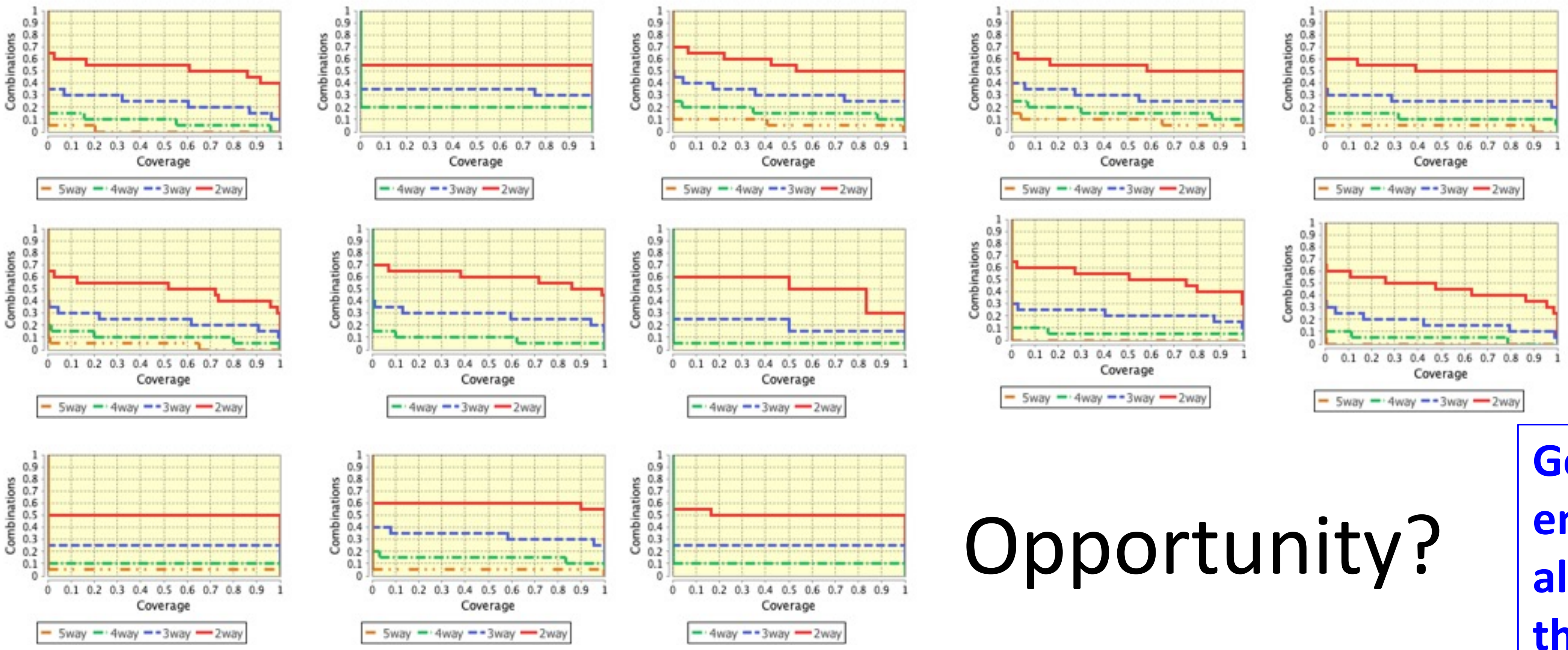
How much combinatorial coverage is achieved with conventional tests?

<u>Spacecraft software example</u>
- 82 variables,
- 7,489 tests,
- conventional test design



What levels of input space coverage are seen in practical machine learning data sets?

Examples from WEKA data mining demo set



Opportunity?

Goal: enumerating all conditions that matter

Research questions

- Practical ML examples <u>don't seem to have very high input space coverage</u> (previous slide)
- Can we improve results with better input space coverage?
- Empirical data show that small numbers of factors are involved in system failures (generally 1 to 6).
- Is this also true of autonomous systems?
- How are input space coverage and classification/prediction accuracy related?
- Can we apply some of these methods to temporal aspects? (sequence covering arrays)

Outline

- Why current safety-critical testing isn't suitable
- Assurance based on input space coverage
- Explainable AI as part of validation, and
- Transfer learning

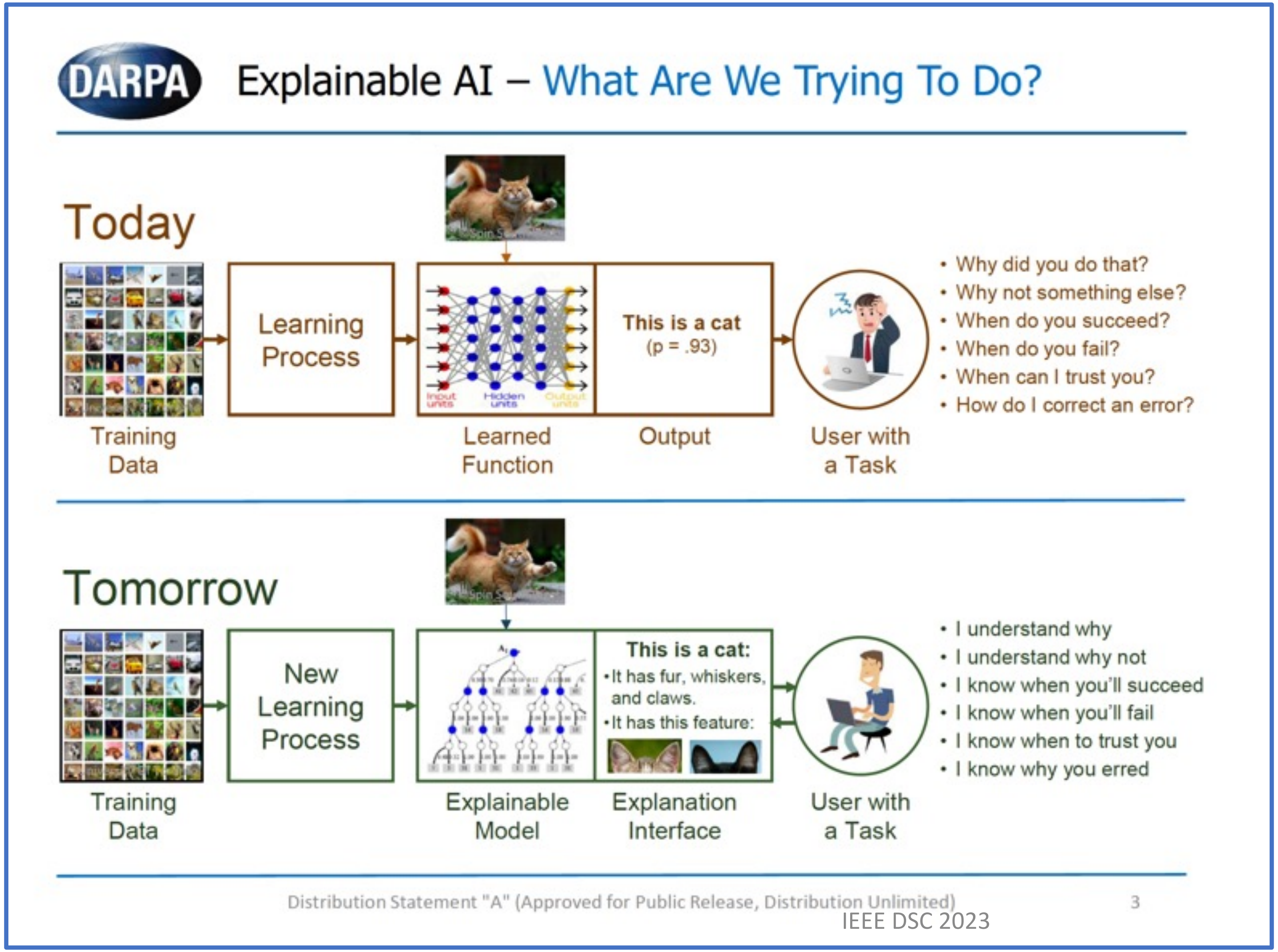
What is the explainability problem?

- AI systems are good, but sometimes make mistakes, and human users will not trust their decisions without explanation or justification
  → <u>assurance and explainability are closely tied</u>

- There is a tradeoff between AI accuracy and explainability: the most accurate methods, such as convolutional neural nets (CNNs), provide no explanations; understandable methods, such as rule-based, tend to be less accurate

- The black-box nature of these systems that makes assurance and testing difficult also makes explanation even harder

Explainability – what's current state of the art?



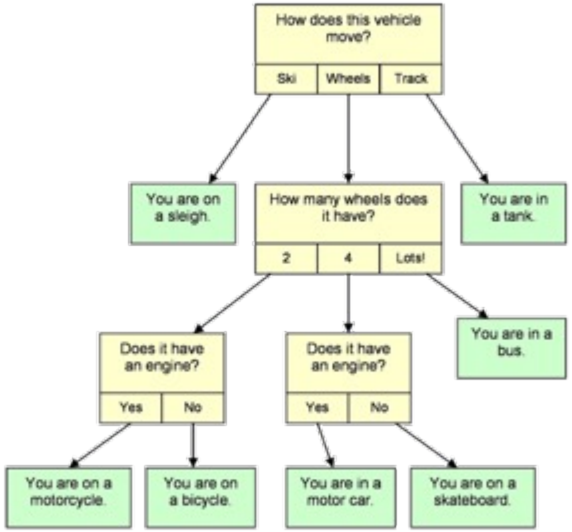
IEEE DSC 2023

Black-box statistical predictions are inadequate

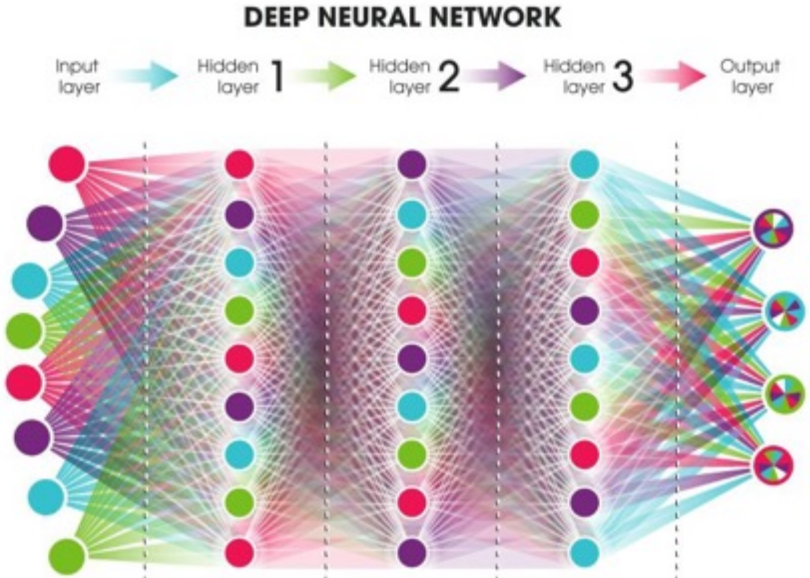
Explanations must be understandable to non-specialist

National Institute of Standards and Technology

Tradeoff:



- OR -



neuralnetworksanddeeplearning.com - Michael Nielsen, Yoshua Bengio, Ian Goodfellow, and Aaron Courville, 2016.

Expert system:
Good for explanations,
not so good for accuracy

Neural nets:
Good for accuracy,
not so good for explanations

Can we get the best of both worlds?

NIST
National Institute of Standards and Technology

What has been tried?

- Interpretable models – e.g. rule-based expert systems: “if patient has symptoms A and B, or has B with C and D, then illness is X”
  - best for explanations
  - hard to find rules
  - less accurate than other approaches
- Modify neural nets etc. to add explanations
  - reduces accuracy, complicates the system
  - explanations still not very understandable
- Model induction - infer explainable model from black-box
  - flexible for application, good explanations using only input, output
  - hard to produce the explainable model
- Our approach – derive rule predicates from inputs and outputs to CNNs and other black-box functions

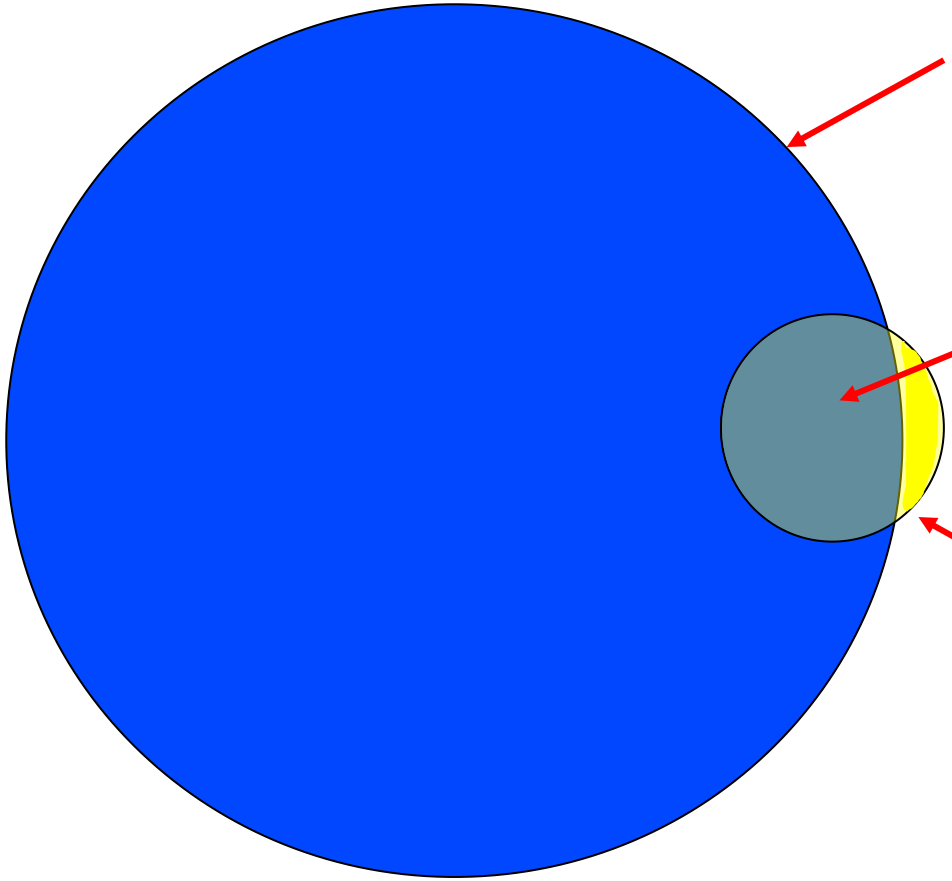
NIST National Institute of Standards and Technology

Fault location – identify fault-triggering input

Given: a set of tests that the SUT fails, which combinations of variables/values triggered the failure?

Blue = combinations in passing tests

Yellow = combinations in passing tests

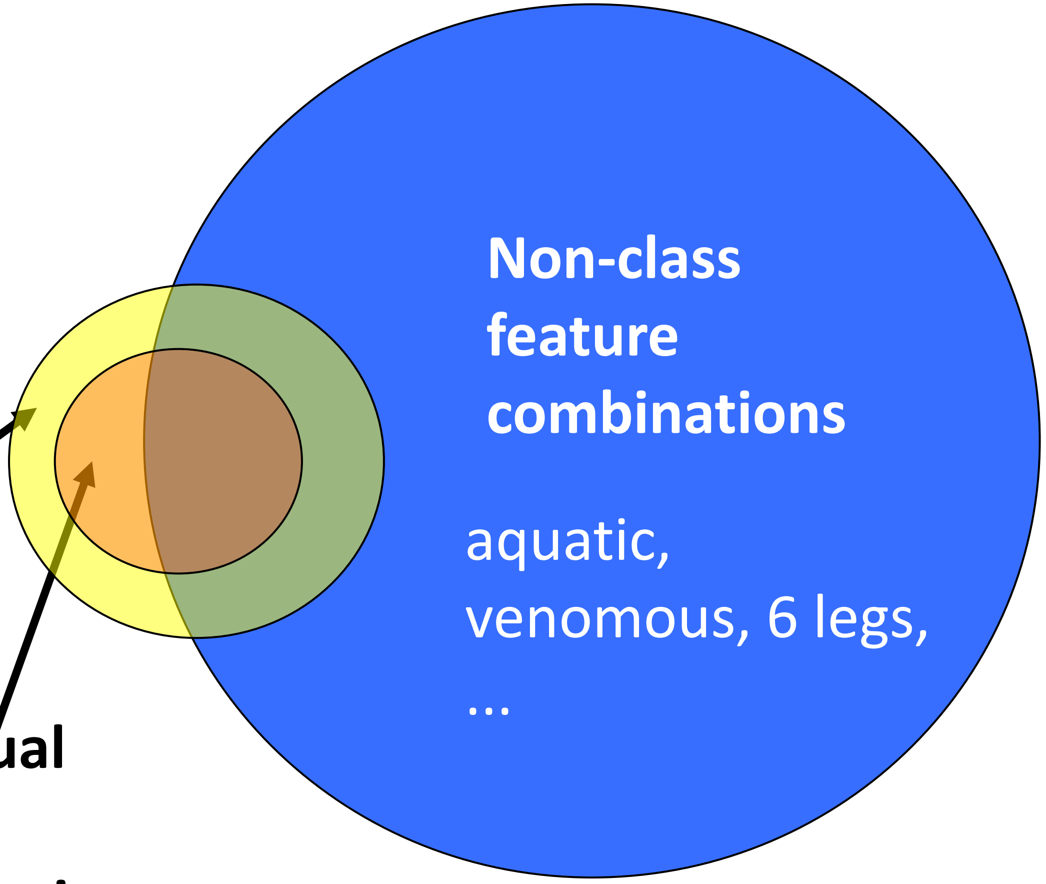


variable/value combinations in passing tests

variable/value combinations in failing tests

Combinations in failing but not in passing tests
<u>These are the ones we want</u>

Relevance to explainable AI



Class feature combinations - brown & furry, black & furry, whiskers, claws, **...**not aquatic, not venomous, not 6 legs,

Individual feature combinations – brown & furry, whiskers, claws, not aquatic, not venomous, not 6 legs, **...**

Animal shares features with cat class

Animal does not share features with non-cat classes

Kuhn, D. R., Kacker, R. N., Lei, Y., & Simos, D. E. (2020). Combinatorial methods for explainable AI. In *2020 IEEE Intl Conference on Software Testing, Verification and Validation Workshops (ICSTW)*

Class File:	Class file rep1.csv; rows=1; cols=16
Nominal File:	Nominal file notreptile.csv; rows=96; cols=16 || 2-way: 120 3-way: 560 4-way: 1,820 5-way: 4,368 6-way: 8,008

Input configuration $2^{15}6^{1}$

Class File Contents:

hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	nlegs	tail	domestic	catsize
0	0	1	0	0	0	0	0	1	1	0	0	4	1	0	1

Is this creature a reptile?

Consider rare combinations

```
---------------
0053 occurrences = 0.552 of cases, hair = 0
0076 occurrences = 0.792 of cases, feathers = 0
0055 occurrences = 0.573 of cases, eggs = 1
0055 occurrences = 0.573 of cases, milk = 0
0072 occurrences = 0.750 of cases, airborne = 0
0061 occurrences = 0.635 of cases, aquatic = 0
0044 occurrences = 0.458 of cases, predator = 0
0039 occurrences = 0.406 of cases, toothed = 0
0078 occurrences = 0.813 of cases, backbone = 1
0076 occurrences = 0.792 of cases, breathes = 1
0090 occurrences = 0.938 of cases, venomous = 0
0079 occurrences = 0.823 of cases, fins = 0
0036 occurrences = 0.375 of cases, nlegs = 4
0070 occurrences = 0.729 of cases, tail = 1
0083 occurrences = 0.865 of cases, domestic = 0
0043 occurrences = 0.448 of cases, catsize = 1
```

No single feature is sufficient explanation – shares features with non-reptiles

No pair of features sufficient – shares 2-way combinations w/ non-reptiles

```
0002 occurrences = 0.021 of cases, toothed,nlegs = 0,4
0005 occurrences = 0.052 of cases, hair,nlegs = 0,4
0005 occurrences = 0.052 of cases, milk,nlegs = 0,4
0006 occurrences = 0.063 of cases, eggs,nlegs = 1,4
0008 occurrences = 0.083 of cases, toothed,catsize = 0,1
0011 occurrences = 0.115 of cases, milk,catsize = 0,1
0012 occurrences = 0.125 of cases, eggs,catsize = 1,1
0013 occurrences = 0.135 of cases, hair,catsize = 0,1
0015 occurrences = 0.156 of cases, predator,catsize = 0,1
```

3-way combinations produce rules to explain recognition of a reptile

```
00000 occurrences = 0.000 of cases, aquatic,toothed,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, eggs,aquatic,nlegs = 1,0,4
00000 occurrences = 0.000 of cases, hair,aquatic,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, hair,nlegs,catsize = 0,4,1
00000 occurrences = 0.000 of cases, milk,aquatic,nlegs = 0,0,4
00000 occurrences = 0.000 of cases, milk,nlegs,catsize = 0,4,1
00000 occurrences = 0.000 of cases, predator,toothed,nlegs = 0,0,4
00001 occurrences = 0.010 of cases, eggs,nlegs,catsize = 1,4,1
00001 occurrences = 0.010 of cases, eggs,predator,nlegs = 1,0,4
00001 occurrences = 0.010 of cases, feathers,toothed,backbone = 0,0,1
```

Non-reptiles in the database do not have these 3-way combinations

Only reptiles have these combinations of features:

not aquatic AND not toothed AND four legs

egg-laying AND not aquatic AND four legs

not hairy AND four legs AND cat size

not milk-producing AND not aquatic AND four legs

not milk-producing AND four legs AND cat size

not predator AND not toothed AND four legs

Mapping combinations to expressions

- Identify *t*-way combinations that distinguish the predicted class from others
- Combinations can be mapped to expressions to produce a rule-based type of explanation

if (not aquatic AND not toothed AND four legs)
OR (egg-laying AND not aquatic AND four legs)
OR (not hairy AND four legs AND cat size)
OR (not milk-producing AND not aquatic AND four legs)
OR (not milk-producing AND four legs AND cat size)
OR (not predator AND not toothed AND four legs)
then reptile;
else not reptile;

As noted, none of the single factors above is sufficient for explanation

Example: empty vs. occupied rooms, using sensor data

File Information

Class File:	Class file o1.csv; rows=1; cols=5
Nominal File:	Nominal file empty.csv; rows=7703; cols=5 || 2-way: 10 3-way: 10 4-way: 5 5-way: 1 6-way: 0
Class File Contents:	Temperature Humidity Light CO2 HumidityRatio B3 B3 B2 B2 B4

2-Way | 3-Way | 4-Way | 5-Way | 6-Way

☑ Enabled

```
Combinations = 10, Settings = 210
0016 occurrences = 0.002 of cases, Humidity,Light = B3,B2
0016 occurrences = 0.002 of cases, Light,CO2 = B2,B2
0036 occurrences = 0.005 of cases, Temperature,Light = B3,B2
0040 occurrences = 0.005 of cases, CO2,HumidityRatio = B2,B4
0043 occurrences = 0.006 of cases, Light,HumidityRatio = B2,B4
0054 occurrences = 0.007 of cases, Temperature,CO2 = B3,B2
0078 occurrences = 0.010 of cases, Humidity,CO2 = B3,B2
0205 occurrences = 0.027 of cases, Temperature,HumidityRatio = B3,B4
0247 occurrences = 0.032 of cases, Temperature,Humidity = B3,B3
0495 occurrences = 0.064 of cases, Humidity,HumidityRatio = B3,B4
---------------
0523 occurrences = 0.068 of cases, Temperature = B3
2415 occurrences = 0.314 of cases, Humidity = B3
0085 occurrences = 0.011 of cases, Light = B2
0534 occurrences = 0.069 of cases, CO2 = B2
2190 occurrences = 0.284 of cases, HumidityRatio = B4
```

Why do we conclude this room is occupied?

These levels of humidity and lighting are strong indication

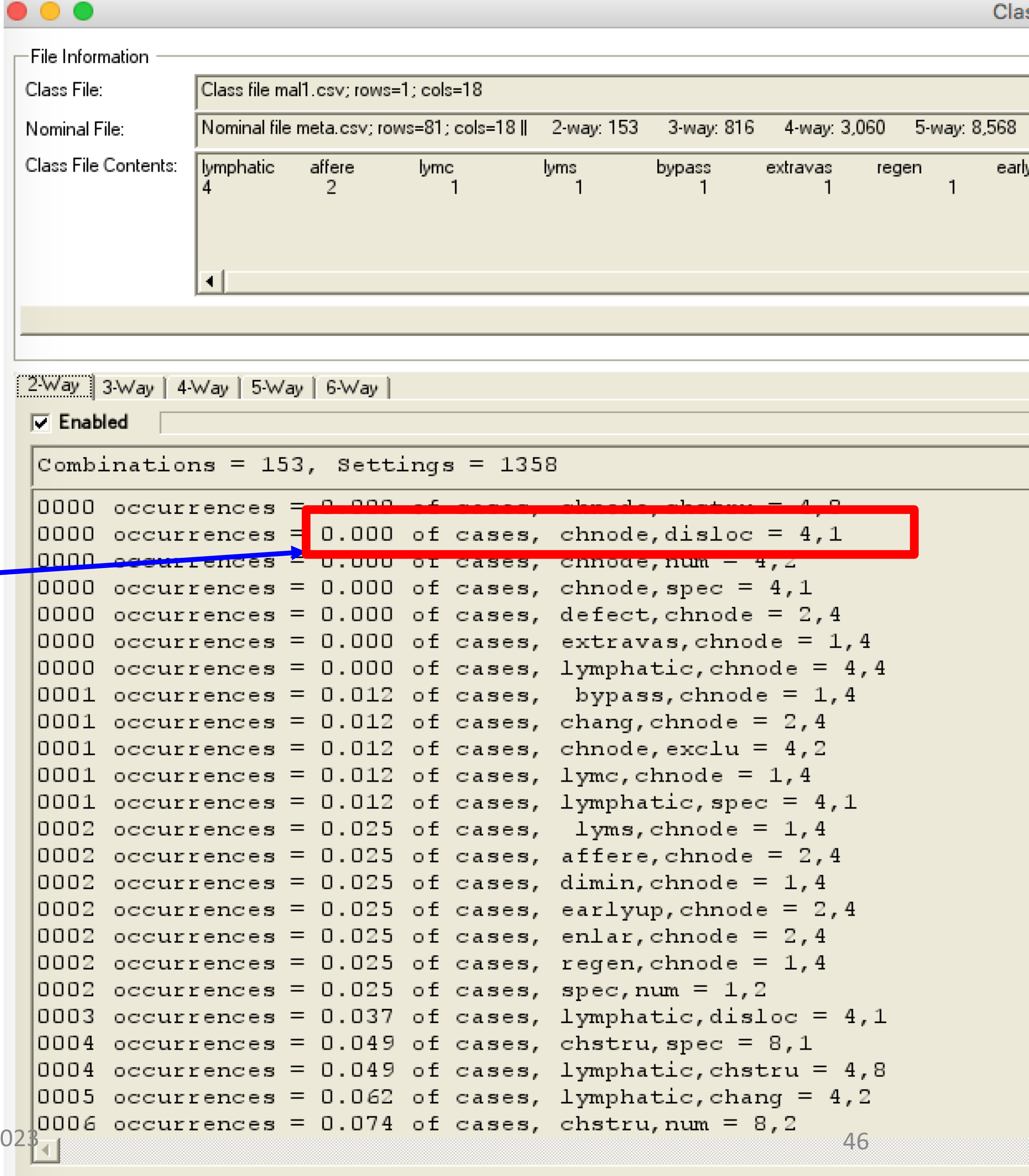
Considering levels of lighting, CO2, and humidity ratio provide even stronger evidence:

Empty rooms don't have these levels

```
00003 occurrences = 0.000 of cases, Light,CO2,HumidityRatio = B2,B2,B4
00005 occurrences = 0.001 of cases, Humidity,Light,CO2 = B3,B2,B2
00008 occurrences = 0.001 of cases, Temperature,Light,CO2 = B3,B2,B2
00011 occurrences = 0.001 of cases, Humidity,Light,HumidityRatio = B3,B2,B4
```

A different example: lymph node pathology – why is this classified as malignant not metastatic?

- These combinations are characteristic of lymphoma that arises in lymph node instead of metastatic that spread to node from somewhere else



Summary - explainable AI

- Combinatorial methods can provide explainable AI
- We have prototype that applies this approach
  - Determine combinations of variable values that differentiate an example from other possible conclusions
    - ➔ Feature combinations present shared with class
    - ➔ Feature combinations not shared with class not present
- Method can be applied to black-box functions such as CNNs
- Present explanation in the preferred form of rules, "if A & B, or C with D & E, then conclusion is X"

NIST National Institute of Standards and Technology

Outline

- Why current safety-critical testing isn't suitable
- Assurance based on input space coverage
- Explainable AI as part of validation, and
- Transfer learning – example application

Transfer learning – what is the problem?

- Differences inevitably exist between training data sets, test data sets, and later real-world data
- Further differences exist between data from two or more different environments

- How do we predict performance of a model trained on one data set when applied to another?
    - New environment
    - Changed environment
    - Additional possible values, etc.

Lanus, E., Freeman, L. J., Kuhn, D. R., & Kacker, R. N. (2021, April). Combinatorial Testing Metrics for Machine Learning. In *2021 IEEE Intl Conference on Software Testing, Verification and Validation Workshops (ICSTW)*

Transfer learning – conventional practice

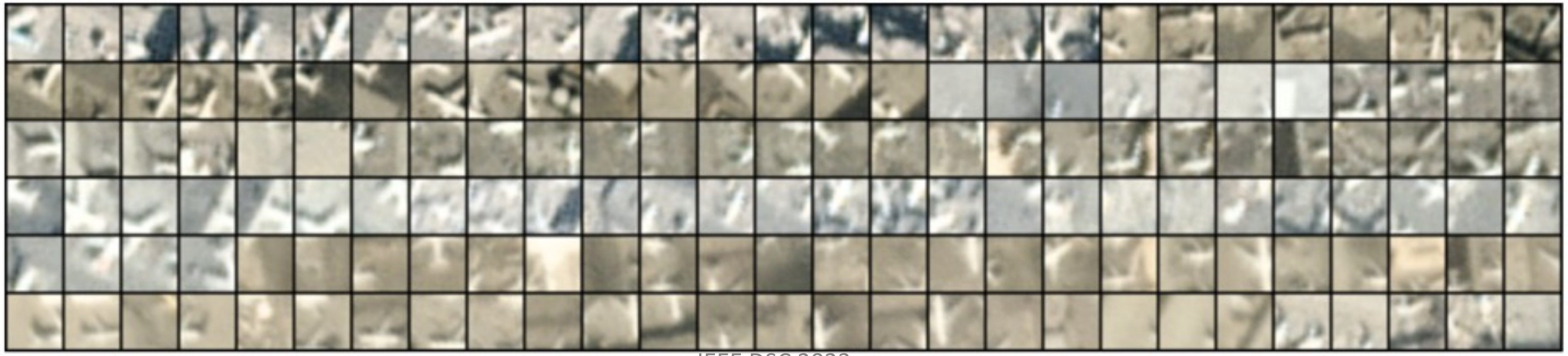
Randomized selection – but how much random data will be sufficient, especially with smaller data sets?

Ensure at least one of each object type – but this may not be representative of object attribute distributions

Interactions are critical to consider in most ML problems, especially for safety, but conventional practice does little to ensure data sets are adequately representative of interactions

Example – image analysis

- Planes in satellite imagery – Kaggle ML data set – determine if image contains or does not contain an airplane
- Two data sets – Southern California (SoCal, 21,151 images) or Northern California (NorCal, 10,849 images)
- 12 features, each discretized into 3 equal range bins



NIST National Institute of Standards and Technology

Transfer learning problem

- Train model on one set, apply to the other set
- Problem –
    - Model <u>trained on larger</u>, SoCal data applied to smaller, NorCal data → <u>performance drop</u>
    - Model <u>trained on smaller</u>, NorCal data applied to larger, SoCal data → <u>NO performance drop</u>
- This seems backwards!
- Isn't it better to have more data?
- Can we measure, explain and predict it next time?

Density of combinations in one versus the other data set, 2-way

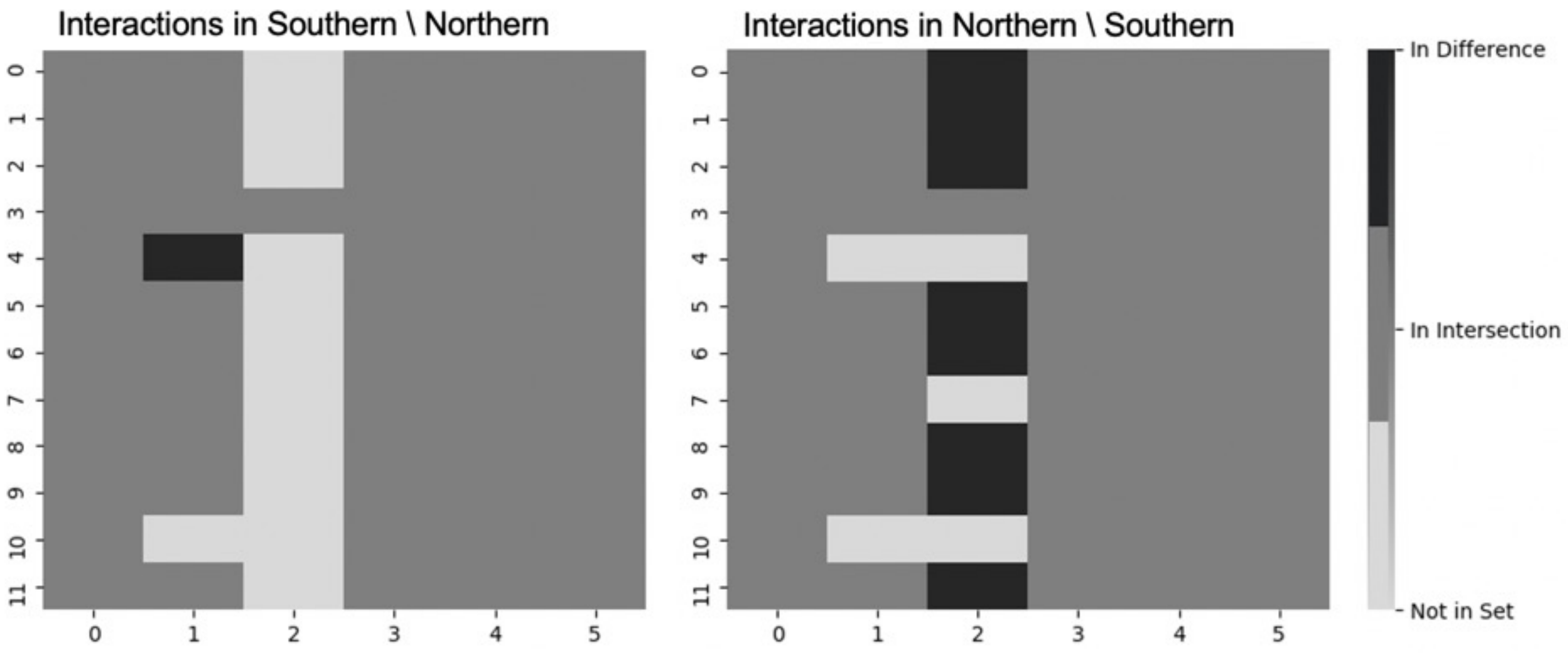


Image from Combinatorial Testing Metrics for Machine Learning, Lanus, Freeman, Kuhn, Kacker, IWCT 2021

For C = SoCal, N = NorCal,
|C\N| / |C| = 0.02
|N\C| / |N| = 0.12

The NorCal data set has fewer “never seen” combinations, even with half as many observations

National Institute of Standards and Technology

Summary – Transfer learning

- Current approaches to estimating success for transfer learning are largely ad-hoc and not highly effective
- Combinatorial methods show promise for improvements – measurable quantities directly related to determining if one data set is representative of the field of application
- Much additional work is needed to evaluate this idea, and to understand the link between combinatorial difference values and prediction accuracy
- Empirical studies planned

Assured autonomy – more questions than answers

- Interactions of learning components with programmed components – especially replacing humans

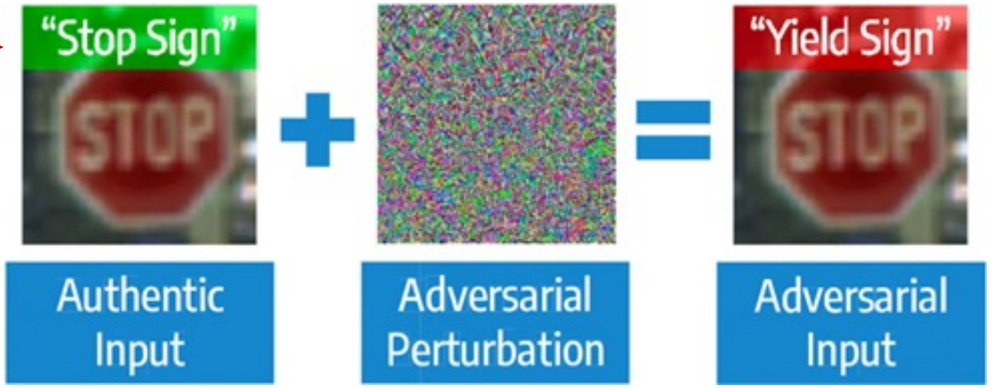
- Changes the nature of system failures
- More like failures involving human factors issues? ☺ Turing test for bugs! Distinguish between human-triggered and AI-triggered system failures?

Assured autonomy – key points & current state

- For capability and cost reasons, <u>autonomous components are becoming routine</u> in software engineering
- Many, or most, <u>methods used in high assurance conventional systems are not sufficient</u> for many autonomous components
  - Structural coverage – not for neural nets, and others
  - Formal proofs – for some parts but limited
- How to deal with learning, dynamic changes in system?
- Understanding and measuring interaction coverage is necessary

Where are we going?

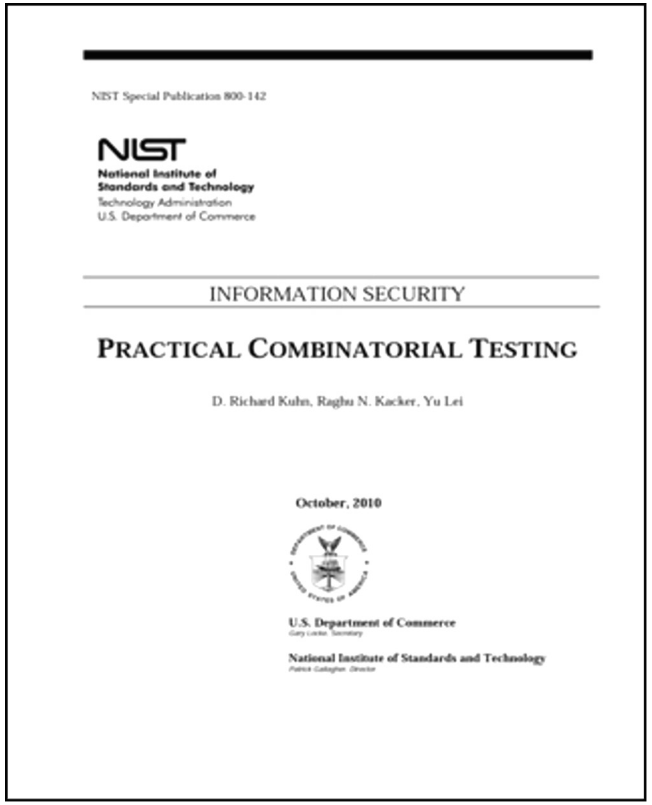
- Need new approaches in:
  - Design
  - Simulation
  - Validation
  - Formal verification
  - Testing
  - <u>Explainability</u>
- *Security – much bigger problem than safety assurance – solvable?*
  - *All the old vulnerabilities apply – with greater consequences*
  - *And new vulnerabilities*



- ***Leading to … AI vs. AI?***

Learning and Applying Combinatorial Methods

- Self-contained tutorial on using combinatorial testing for real-world software
- Key concepts and methods, explains use of software tools for combinatorial testing
- Advanced topics such as the use of formal models and test oracle generation
- Costs and practical considerations
- Designed for testers or undergraduate students of computer science or engineering


NIST Special Publication 800-142

NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

INFORMATION SECURITY

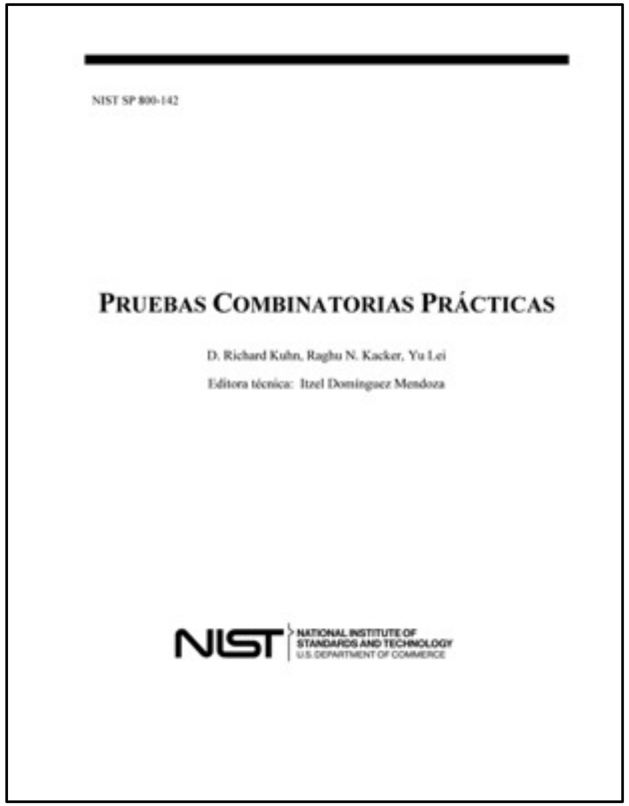
PRACTICAL COMBINATORIAL TESTING

D. Richard Kuhn, Raghu N. Kacker, Yu Lei

October, 2010

U.S. Department of Commerce

National Institute of Standards and Technology


NIST SP 800-142

PRUEBAS COMBINATORIAS PRÁCTICAS

D. Richard Kuhn, Raghu N. Kacker, Yu Lei

Editora técnica: Itzel Dominguez Mendoza

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY U.S. DEPARTMENT OF COMMERCE

NIST
National Institute of Standards and Technology

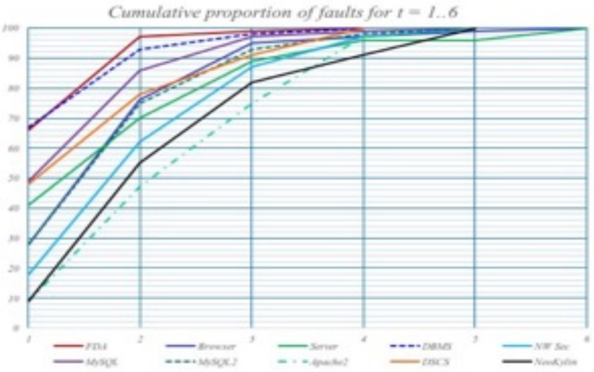
http://csrc.nist.gov/acts

Automated Combinatorial Testing for Software ACTS

Overview

Combinatorial methods can reduce costs for software testing, and have significant applications in software engineering:

- **Combinatorial or *t*-way testing** is a proven method for more effective testing at lower cost. The key insight underlying its effectiveness resulted from a series of studies by NIST from 1999 to 2004. NIST research showed that most software bugs and failures are caused by one or two parameters, with progressively fewer by three or more, which means that combinatorial testing can provide more efficient fault detection than conventional methods. Multiple studies have shown fault detection equal to exhaustive testing with a 20X to 700X reduction in test set size. New algorithms compressing combinations into a small number of tests have made this method practical for industrial use, providing better testing at lower cost. See articles on high assurance software testing or security and reliability.



- **Autonomous systems assurance:** Input space coverage measurements are needed in life-critical assurance and verification of autonomous systems, because current methods for assurance of safety critical systems rely on measures of structural coverage, which do not apply to many autonomous systems. Combinatorial methods, including a theorem relating measures of input space coverage, offer a better approach for autonomous system verification.
- **Metrology* for software engineering.** Sound engineering requires adequate measurement and analysis. Structural coverage enables formally defined criteria for test completeness, but even full coverage may miss faults related to rare inputs. Combinatorial methods open new possibilities for metrology in software engineering, providing a more scientific approach to assurance and verification.
  *Metrology is the science of measurement (NIST is the US national metrology institute).

NEW: Combinatorial Coverage Difference Measurement for assurance of autonomous systems and

PROJECT LINKS

Overview
FAQs

ADDITIONAL PAGES

Quick start
Downloadable Tools
Combinatorial Methods in Testing
- Why do Combinatorial Testing?
- Event Sequence Testing
- Oracle-free Testing and Test Automation
- Case Studies

Input Space Measurement
- Why Measure Input Space?
- Case studies

Assured autonomy
- Explainable AI, Verification, and Validation
- Rule-based Expert Systems and Formal Methods
- Case studies

Cybersecurity Testing
- Combinatorial approach
- Magic mirror vulnerability testing tool
- Case studies

Software Testing Methodology
- NIST Testing Process
- DOs and DON'Ts of testing

ACTS Library
- Fundamental background papers
- Papers on combinatorial test methods
- Covering Array Library

NIST
National Institute of Standards and Technology

http://www.afit.edu/STAT/

AFIT

AIR FORCE INSTITUTE OF TECHNOLOGY

HOME ABOUT AFIT GRADUATE EDUCATION CONTINUING EDUCATION RESEARCH CONSULTING STUDENTS LIBRARY ALUMNI CAREERS SEARCH QUICK LINKS

STAT CENTER OF EXCELLENCE

CENTER HOME OUR PEOPLE BEST PRACTICES AND TEST PLANNING GUIDES TOOLS ABOUT PROGRAMS STAT NEWSLETTER ASK A STAT QUESTION SHORT COURSES

Welcome To The STAT Center Of Excellence

DASD (DT&E), in collaboration with the Commander Air Education and Training Command, established the STAT Center of Excellence (COE) in April 2012 under the stewardship of the Air Force Institute of Technology (AFIT). The COE attained Full Operational Capability in July 2012.

During development of the Test & Evaluation Master Plan (TEMP), the COE works with acquisition program managers and the program's Chief Developmental Tester to improve test effectiveness and ensure efficient use of scarce resources. Utilizing a combination of rigorous scientific methods and lessons learned, the COE determines where test designs can be improved and efficiencies gained, and then applies this knowledge to the program's T&E strategy development.

In order to achieve more defensible test results, the STAT Center initially partnered with 20 major acquisition programs. This partnership has grown to support more than 59 major acquisition programs since 2012. As a condition for effective partnering, these programs are early enough in their test strategy planning to allow the implementation of STAT to allow the better informing of the program leadership. The use of STAT does not ensure the success of a program, but rather allows programs to make better use of

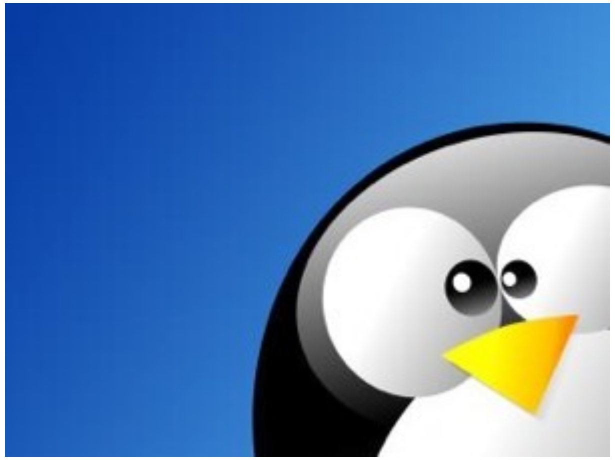
NIST National Institute of Standards and Technology

Freely Available Tools

- **Covering array generator** – basic tool for test input or configurations;
- **Combinatorial coverage measurement** – detailed analysis of combination coverage; automated generation of supplemental tests; helpful for integrating c/t with existing test methods
- **Sequence covering array generator** – new concept; applies combinatorial methods to event sequence testing
- **Input modeling tool** – design inputs to covering array generator using classification tree editor; useful for partitioning input variable values
- **Fault location tool** – identify combinations and sections of code likely to cause problem

NIST
National Institute of Standards and Technology

Please contact us
if you're interested!



Rick Kuhn, Raghu Kacker, M.S. Raunak
{kuhn, raghu.kacker, raunak}@nist.gov

http://csrc.nist.gov/acts