BROADCOM SOFTWARE

Aerospace-grade Software QA in Broadcom PTC

The journey, the how and the results

About the speaker

Ondrej Tulach

- Former programmer (IBM PL/I, C#.NET)
- Former Windows Sysadmin
- Senior QA Engineer, 12 years experience in CA/AVGO
- Self-learning advanced Aerospace/Mil-grade QA + SSRE
- Thinks in categories of systems, taxonomies, data trees

BROADCOM SOFTWARE

When „basic testing“ and Automation isn‘t enough

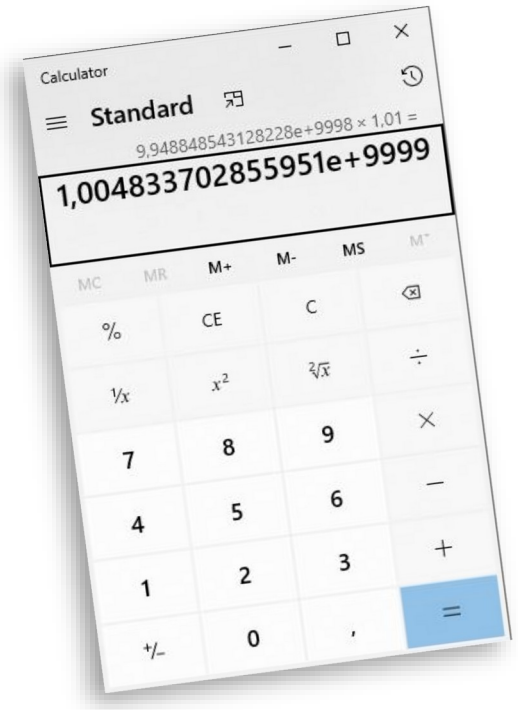
- Upon joining CA in 2011, I got top-end QA training from RCBS/ISTQB
  - This training and associated skills has been absent for years
  - PTC has designed internal Basic and Advanced QA courses to fill the gap
- As a former programmer, I enthusiastically pushed for Test Automation...
- Yet automated testing failed to discover defects!

- What was missing? **Test coverage** beyond Basic Positive Testing

BROADCOM SOFTWARE

Basic Positive Testing vs. Real world

- Theoretical quality:
  - We have tested 100 % Code Coverage
  - We have tested 100 % of functionality
  - We have tested 100 % Boundary Values



- Real world:
  - Unexpected user input
  - Crashed 1st minute during 1st attempt
  - One more angry customer

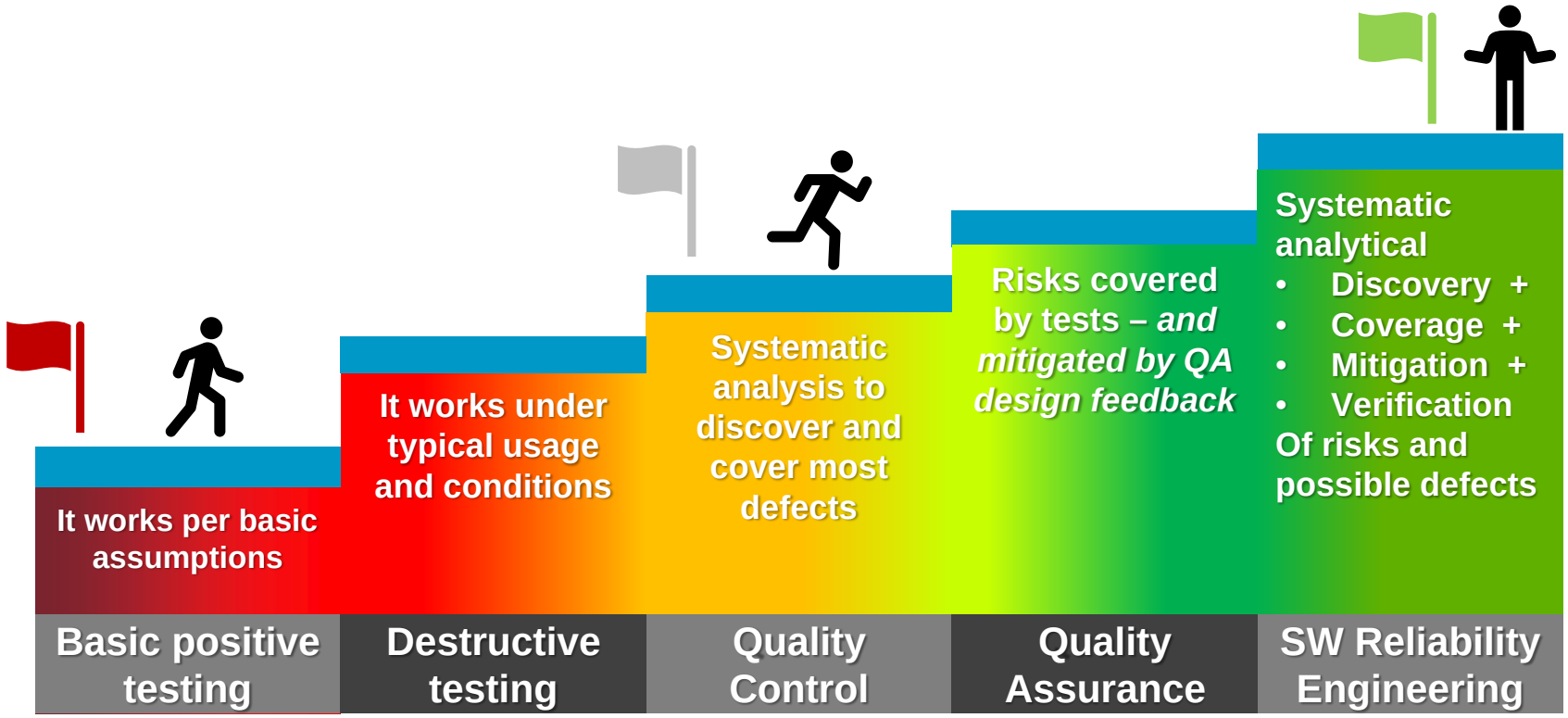
BROADCOM SOFTWARE

Basic Positive Testing flaws and solution

- Basic Positive Testing – test that the code does what DEVs wanted
  - Not that it would handle what the *users* wanted
  - Not that it would handle real-world host-systém influence and interference
- Next level: Destructive testing
  - HOET - what the users could/would do (see Steve Krug)
  - Challenging assumptions, advanced scenarios, inputs, combinations, workload
  - Failure Mode testing (*how it fails when it fails* – error messages...)

Still not enough!

The journey: stepping up the Quality



BROADCOM SOFTWARE

How it unfolded

- 2011 – ISTQB and boundary values, destructive testing
- 2015 – Technical Test Analysis
- 2016 – Test Classes Checklist/Cheatsheet → the largest leap in Test Coverage
- 2017 – BS7925 Component testing. Top-of-the-line QC.

Is that it? Cannot QA get better, discover more defects, lower risks more?

- 2019 – System Safety/Reliability Engineering → proper QA feedback loops
- 2019 – Applied SSRE Fault Tree Analysis to significantly improve „pain code“
- 2021 – Discovered „the safe systems development Bible“ – ARP4761
- 2021 – mapped everything (FnL, TCl, FMEA, Risks) into one integrated workflow

BROADCOM SOFTWARE

Cherry-picking the Aerospace SotA *systematic* QA/SSRE

- Extremely robust, systematic, analytical

- But also too formal, documentation-heavy, „certifications“
- For our purposes: cherry-pick and „LEAN-ize“ ideas, concepts

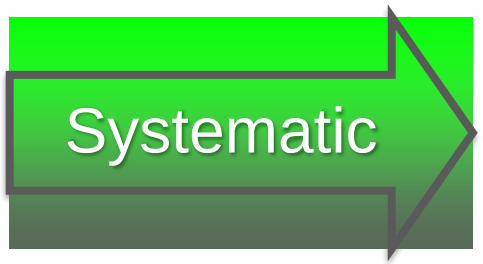
BROADCOM SOFTWARE

Taxonomy of fundamental QC/QA approaches

- Exploratory testing, Error guessing
- Typical Unit testing
- Manual subset/crowd Regression

- Defect discovery ~ **personal experience**
- **Unknown** test coverage, unknown quality
- „*Should we test more? And what?*“

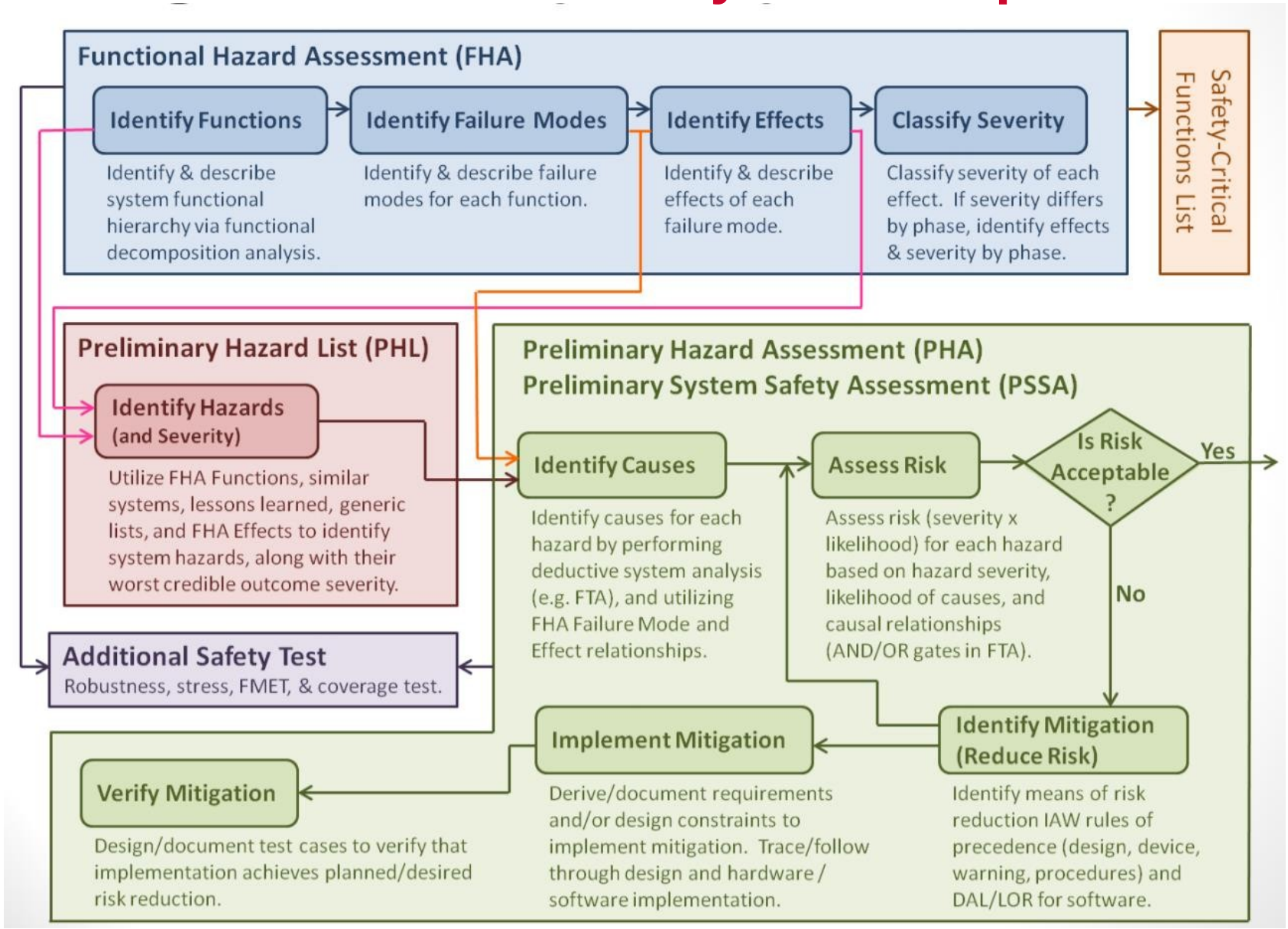
- Generic test lists, Function lists
- FHA,PHA-enabled Unit testing
- Function-based Regression tests



- Defect discovery is **data-based**
- **Known** test coverage, assured quality
- „*Optimize risk-benefit ratio of tests*“

BROADCOM SOFTWARE

The Pillar of aviation safety & development: SAE ARP4761



How to cherry-pick?

- FHA phase
  - Function list
  - FMEA
  - Risk analysis
- PHL phase
  - Generic lists = Test Classes
  - Lessons learned = Defect RCA (in TCI)
- PHA phase
  - QA mitigation suggestions

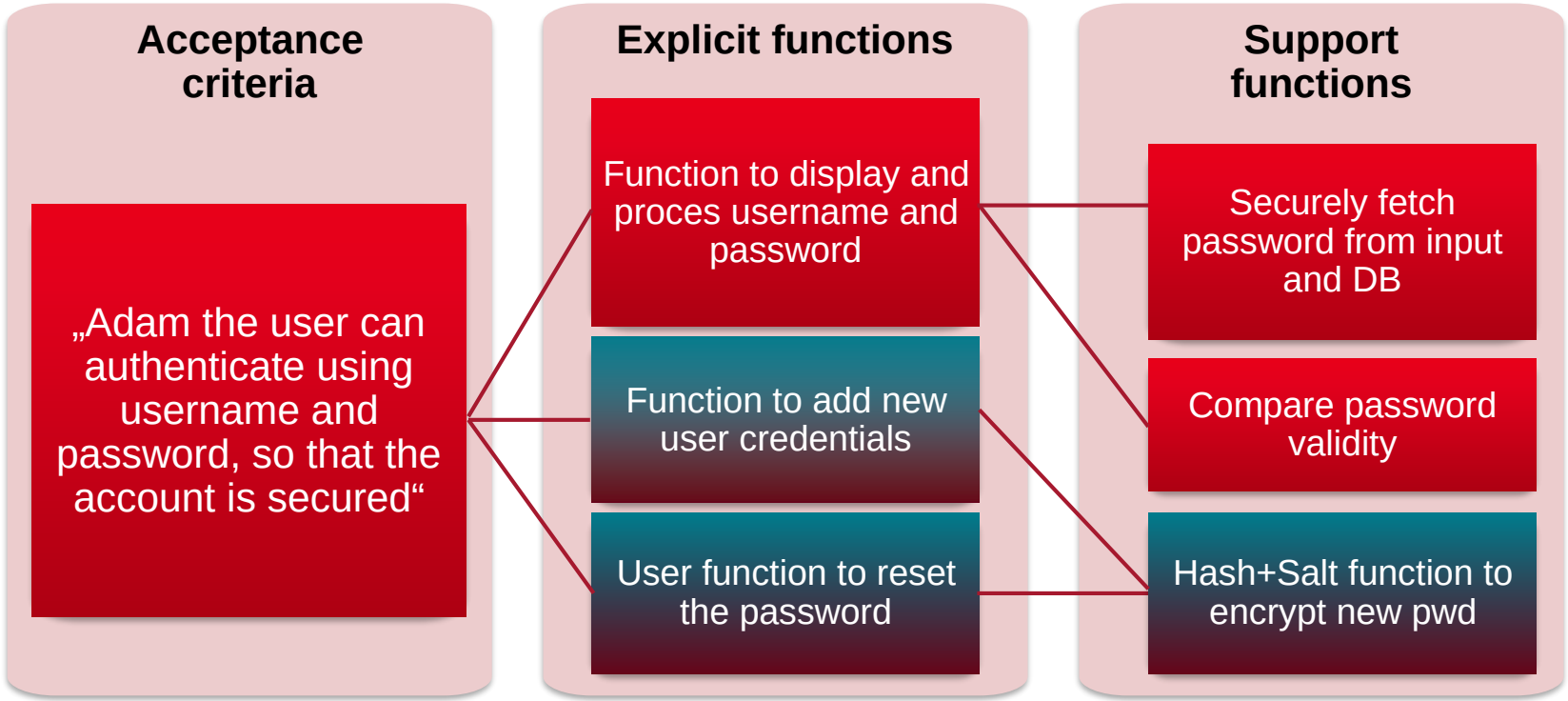
Cherrypicking the Fault Hazard Analysis

Function list - why

FHA phase

- Grooming, Personas, Acceptance criteria: few „explicit“ primary functions
- Real product: needs additional „fill the blanks“ primary functions
- Back-end implementation: needs many support functions

- Example:



BROADCOM SOFTWARE

Function list - how

FHA phase

- Ad-hoc discovery of Secondary + Support functions during coding?
  - Missing objects and support = frequent re-architecture to add missing needed
    - Any re-architecture decreases quality
- Instead:
- Function lists allow **systematic** discovery of **all*** functions *before* coding!
- DEV and QA sit together and
  1. Write down all known Primary functions (from Acceptance criteria etc.)
  2. Identify needed Secondary functions for Primary functions
  3. Identify needed Support functions for all Primary and Secondary Functions
- Function list then productively used as:
  1. Unit tests goals for DEV
  2. Input for FMEA and Risk analysis by QA (and possibly PO)
  3. Functional test design goals for QA

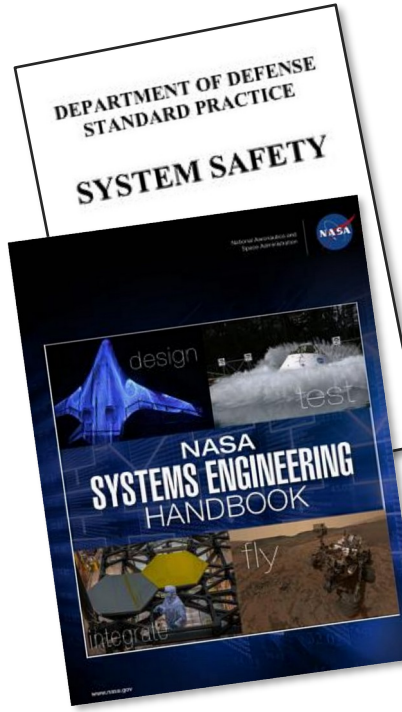
*except late design changes

BROADCOM SOFTWARE

Simplified FMEA

FHA phase

- Method for discovering functional risks
- For high-integrity software, quite complex
- Simplified FMEA cuts to the bones:
  1. For every function in Fn list,
  2. Identify how it could fail
  3. Identify how severe would such fail be (using helpful SAE J-1739 10-point scale)



Fn	Failure mode	Failure effect	Sev
1.1	Failure to correctly separate parameter from value	[redacted] not encrypted [redacted]	8
	Failure to read and pass the entire value correctly		
	Failure to process the parameter (i.e. ignoring or rejecting it)		
	Failure to handle atypical yet allowed [redacted] (e.g. "KEY.")		
	Failure to capitalize lowercase [redacted]		
1.3	[redacted] not shown although coded	Confusion and confidence loss	6
	[redacted] shown although not coded	False assurance – security issue	10

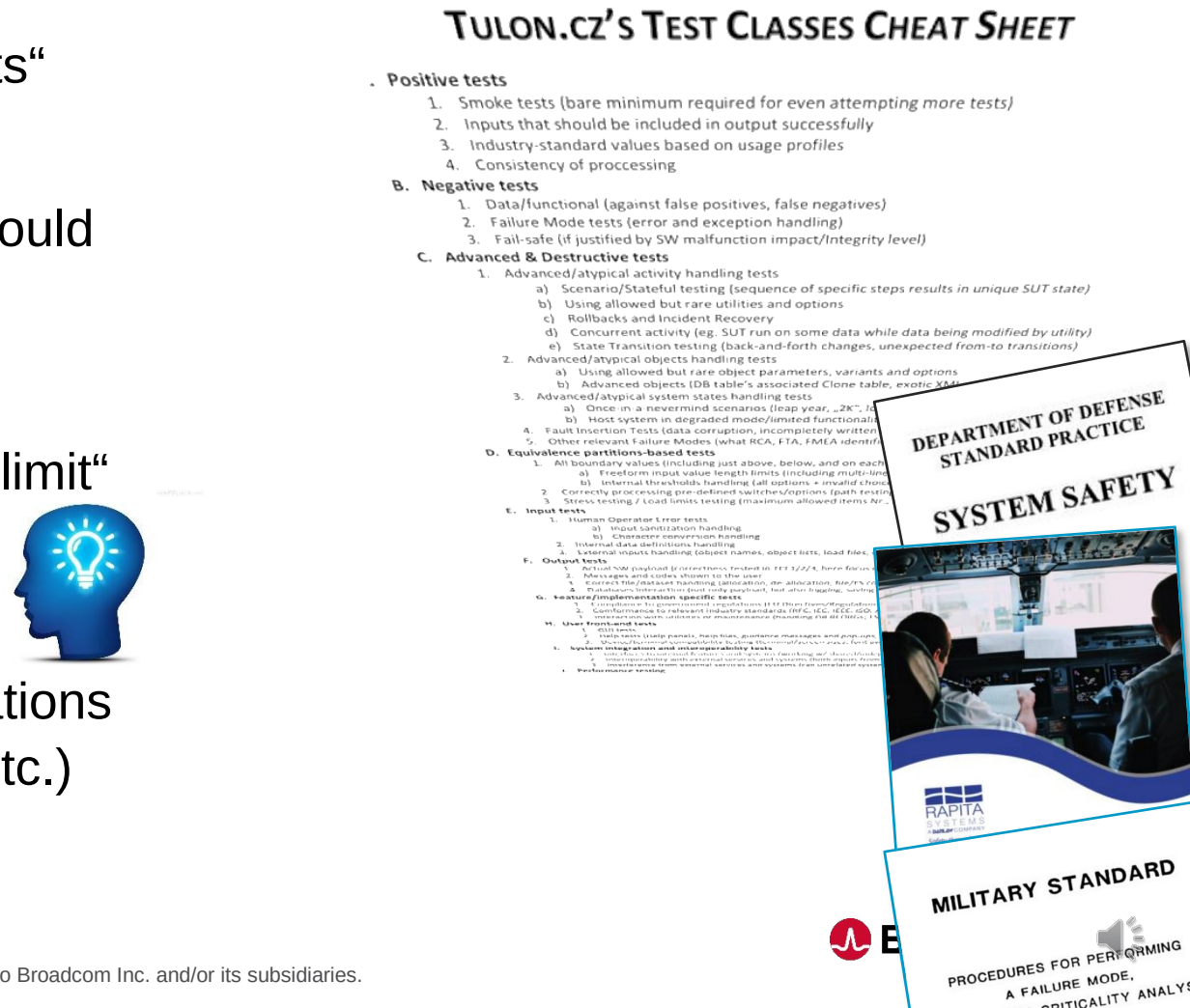
BROADCOM SOFTWARE

Cherrypicking the Hazard and Generic Lists

Generic testing list – the Test Classes Cheat Sheet

PHL phase

- Hidden treasure in IEEE 829-2008 Test plans
  - „A designated grouping of test cases“
  - „Summarizes the unique nature of particular tests“
- IEEE gave few examples:
  - „Positive (or valid) testing of input values that should be processed successfully“
  - „Negative (or invalid) values that should NOT process but do provide an appropriate error
  - „All boundary values on, above and below each limit“
- Idea: develop „**Test design Cheat-sheet**“
  - No more guessing „*what else to test*“ !
  - **Systematic** coverage of all test areas considerations
  - Include **feedback** from Lessons learned (RCA etc.)
  - Checklist: **data-based** Test areas coverage



10 basic TCCS items → coverage increase by 400 %

PHL phase

1. Positive testing
2. Negative testing (not Destructive)
3. Destructive and advanced testing
4. Boundary values testing
5. Input testing
6. Output testing
7. Front-end testing
8. Compliance testing
9. Integration and interoperability testing
10. Performance testing

- Pilot program in DB2B&R PLA
  - Test analysis + Test classes **quadrupled** typical test cases № and coverage
- Consistently used by PLA, XCOM, maybe others?
- Generic TCCS → Product-specific TCCS
- Training provided to 15+ teams

BROADCOM SOFTWARE

TCCS incorporates „Lessons learned“

PHL phase

- Add more specific Test (sub)Classes:
  - When a customer defect is discovered, yet no tests cover that category
  - When a code path ignored by current test categories is found
  - When exploratory testing finds new test categories to cover
- Very fast feedback and quality improvement
  - Each defect caught and „learned back“ into TCCS should prevent same-class future defects
  - Lessons learned from testing feature X translate into testing *all* features

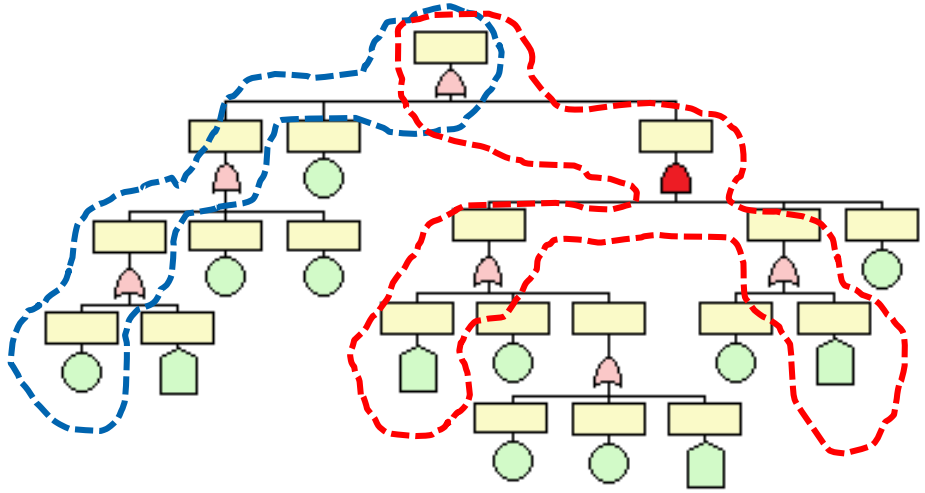
BROADCOM SOFTWARE

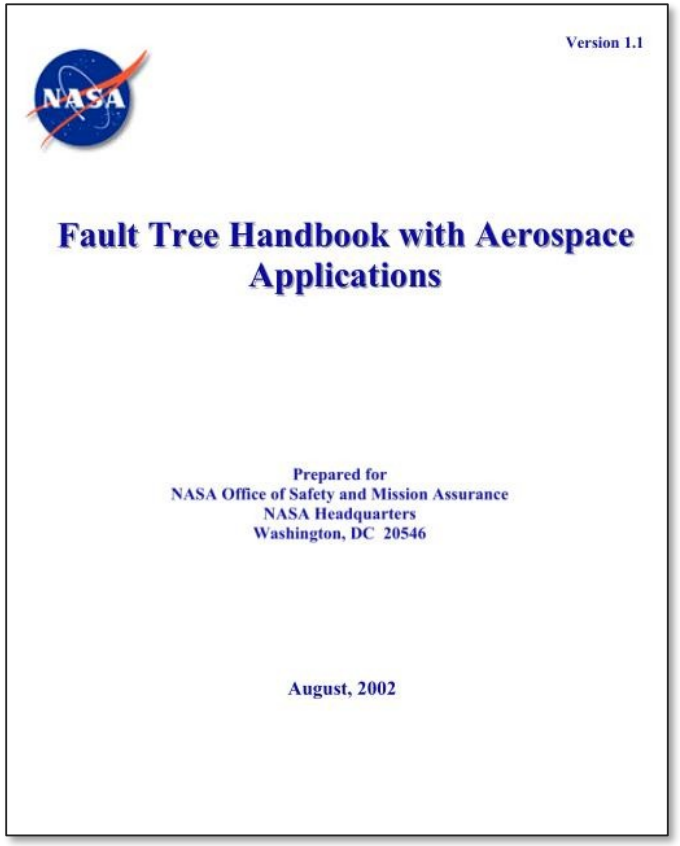
System Safety/Reliability Analysis tools

Fault Tree Analysis – from NASA handbook

PHA/PSSA phase

- Systems modelling technique
- Allows discovery of *all possible* Root Causes of Defects
- Detects error propagation within system
- Uses logical gates (AND, OR...) for conditioned errors





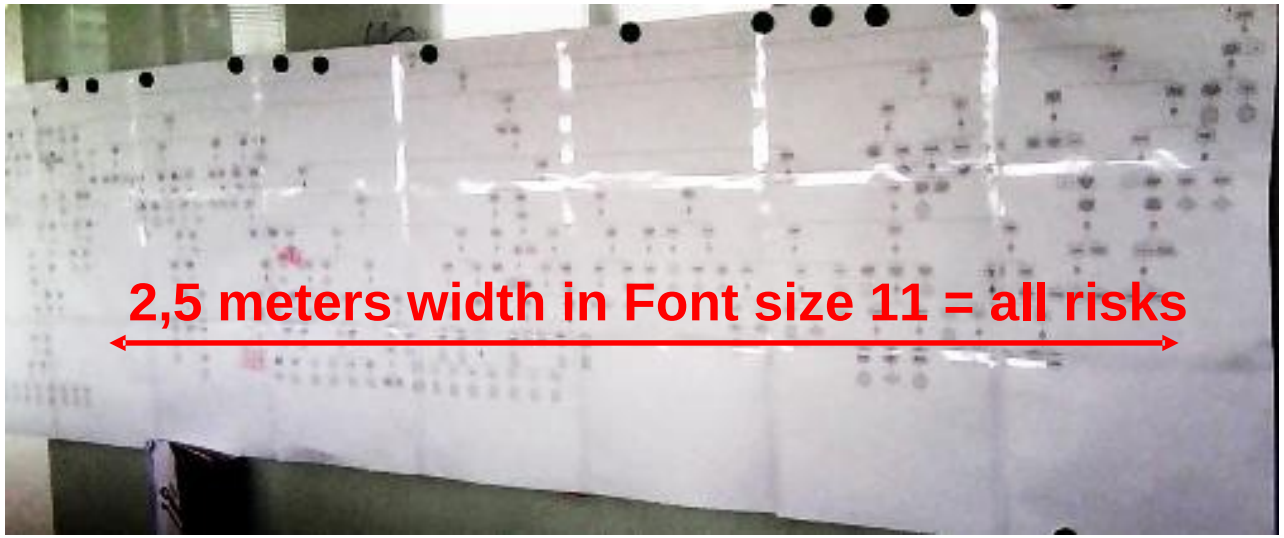
- Extremely powerful for defect-proofing large/legacy code modules

BROADCOM SOFTWARE

NASA FTA applied in Broadcom

PHA/PSSA phase

- Pareto principle: 80% of defects in 20% of code
- Pain-point of customer defects was „Huffmann Compression Dictionary“ support for IBM Db2
- New feature required expansion of this code!
- Engineering Excellence PI → use FTA to fix it
- FTA Top-Level-Event: failure to decompress



1. Review all customer defects 2004~2018
2. Review all IBM HCD architecture docs
3. Review our HCD code
4. Map all functions and possible failures on FTA

↓

Cca. 200 test cases

↓

Most feature defects fixed

BROADCOM SOFTWARE

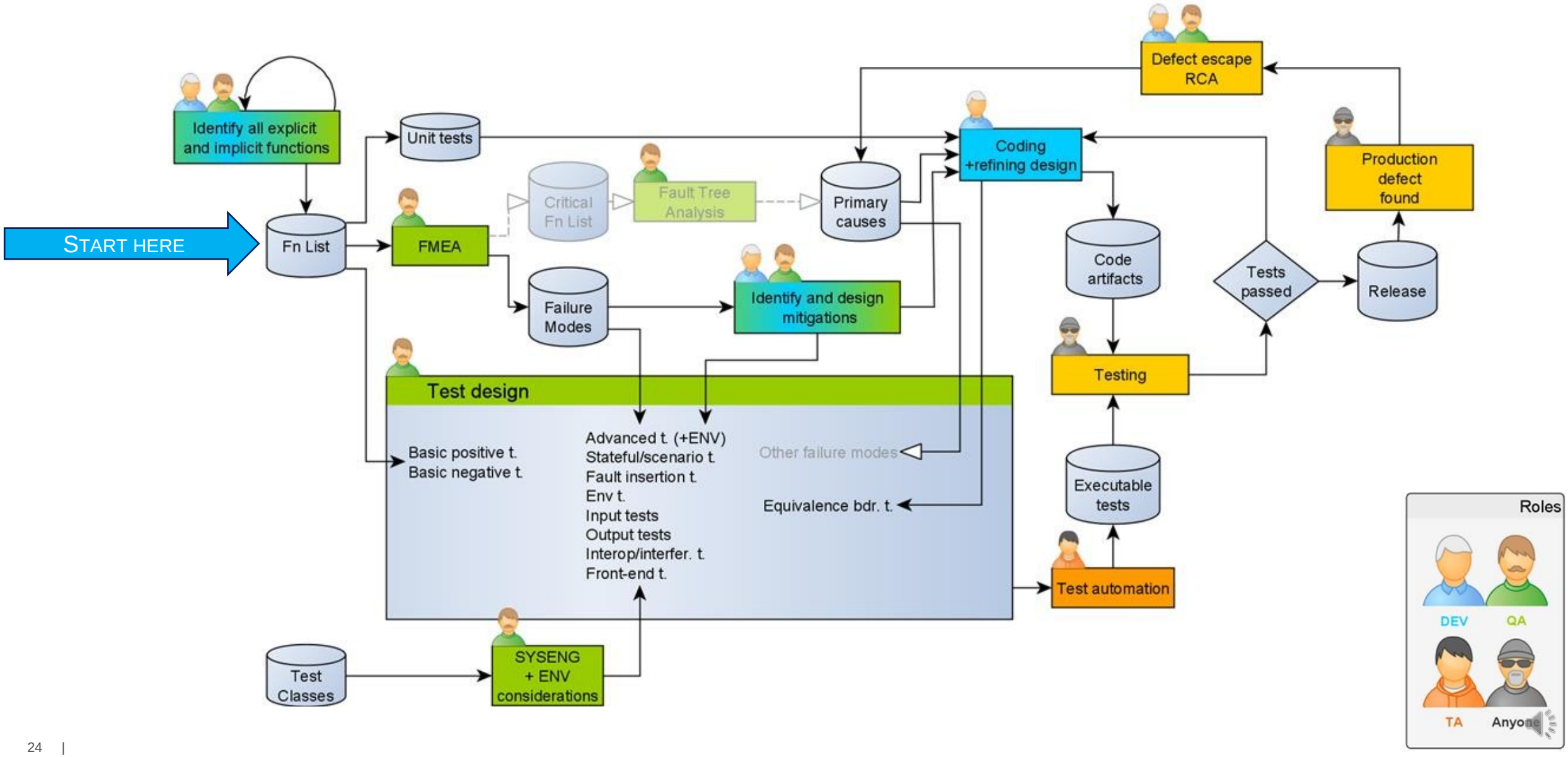
Cherrypicking Aerospace-inspired QA workflow

Agile, no-waste, high-efficiency QA approach

- QC is reactive (after code is written), QA is preventive (before coding)
- Aerospace-inspired FHA enables better code, better quality
  - Function Lists decrease refactoring needed, increase functions reuse
  - FMEA identifies risky functions before coding, allows redesign for more robust code
- Very lightweight, Lean
  - Completely unlike Functional Specifications – FnList takes page or two
  - FnList is more akin a C-language header file
- No waste, high efficiency
  - FnList is used 3× - for Unit tests, for FMEA Risk analysis, for Functional test design
  - FMEA is used 2× - for Risk analysis, for Functional FMET tests design
  - Test Classes are efficient 3× - Test design cheatsheet, Test coverage checklist, anti-redundancy
- Follows Agile Manifesto – Working software over comprehensive documentation
  - Not „doc for doc sake“, but practical help for producing working software

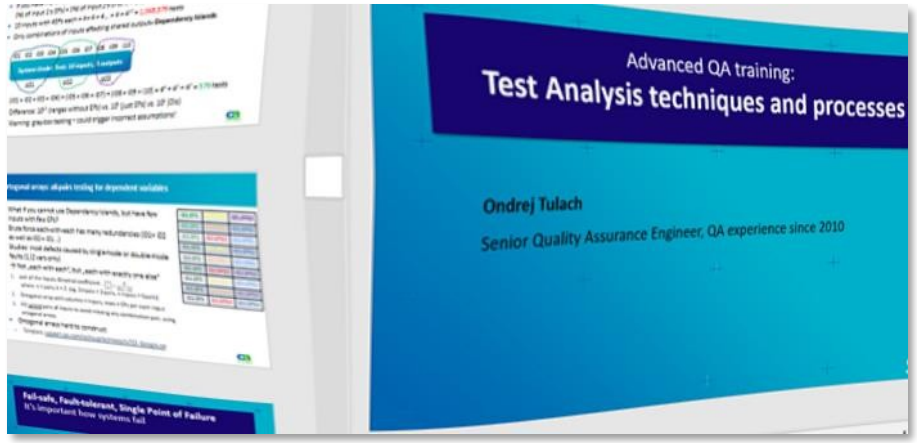
BROADCOM SOFTWARE

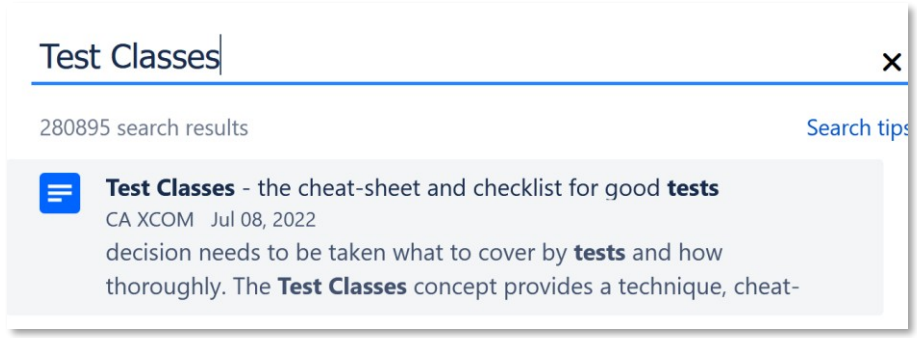
Lean Aerospace-inspired workflow (QA+DEV+Automation)



Broadcom Advanced QA training

- Happy to share what I‘ve learned
- AQA:ATAT –intensive 9-hour QA/SSRE course
  - Recording in English available: https://broadcom.box.com/v/AVGO-MFD-QA-ATAT-2022
  - Live session for your team? Ask Zuzana Pruchova
- Teached QA to Czech Technical University students via Broadcom-EDU partnership (2019)
  - Slides in Czech available, covering basics to advanced
- Test Classes lecture also transcribed to CA Wiki
  - just search Confluence for „Test Classes“
- Custom/Ad-hoc knowledge transfers possible





BROADCOM SOFTWARE

Q & A

BROADCOM SOFTWARE