

# Tools, Evidence, Arguments & Standards

Mark Lawford

Associate Director

McMaster Centre for Software Certification

Professor

Department of Computing and Software

McMaster University

Hamilton, ON

Canada



McSCert

# Outline

- Goals & Motivation
- Some “strawman tools” as examples to think about
- Do you need to qualify the tool(s)?
- Overview of tool qualification in the standards
- Uncovering the arguments and assumptions
- Going forward

# Goals

- Goal: Try to understand different software *tool* qualification requirements in *standards*:
  1. DO-178C Software Considerations in Airborne Systems and Equipment Certification + DO-330 Software Tool Qualification Considerations
  2. IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements & Part 4 Defs & Abbrev
  3. ISO 26262 Road vehicles — Functional safety —Part 8: Supporting processes

# Goals & Motivation

- In particular, for each standard:
  1. What *evidence* is required to qualify a tool?
  2. What is the (typically implicit) *argument* behind the tool qualification requirements?
- Compare and contrast standards
- Ease adoption of Formal Methods (FM) tools by:
  1. Helping academic tool developers understand what they need to do to have their tools used
  2. See that it might not be that hard to qualify an FM tool

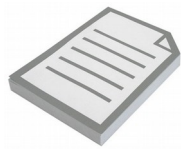
# Inspiration & Disclaimer

- Idea to do this came from Dagstuhl Seminar:
  - 15182 Qualification of Formal Methods Tools  
April 26-29, 2015
- This is preliminary work. Any errors or misunderstandings of the standards are attributable to me, not the other seminar participants. Any good ideas accidentally contained here are probably from the other participants. I bit off way more than I can chew when I told Alan I could do this. The standards seem to have arbitrary difference, though a lot in common. Are you still reading this? Blame Alan.....

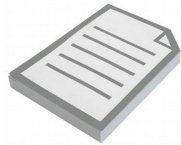
# Some Tools to Consider

- To make things concrete, keep the following 2 tools in mind for the presentation
- Think how you would classify them according to each standard

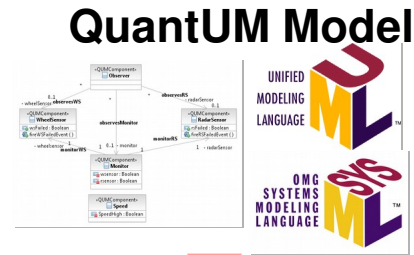
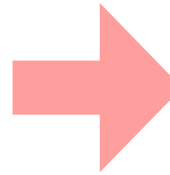
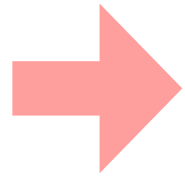
# The QuantUM Tool



Failure Mode Specification



(Safety-) Requirements



automatic

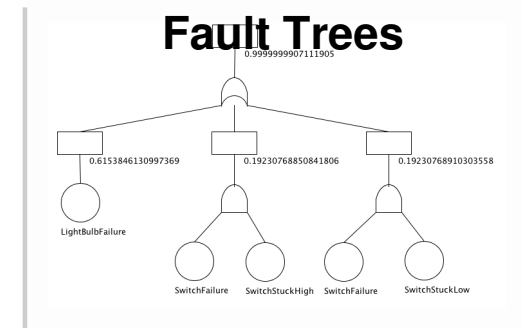
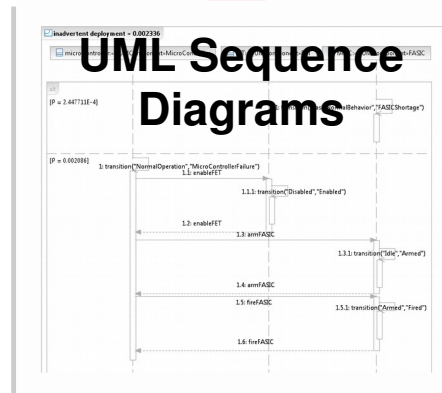
# QuantUM

Quantitative Analysis of UML Models

automatic

automatic

- Tool offers automated model-based functional safety analysis (Fault Tree Analysis, FMEA). It can directly process architecture models.



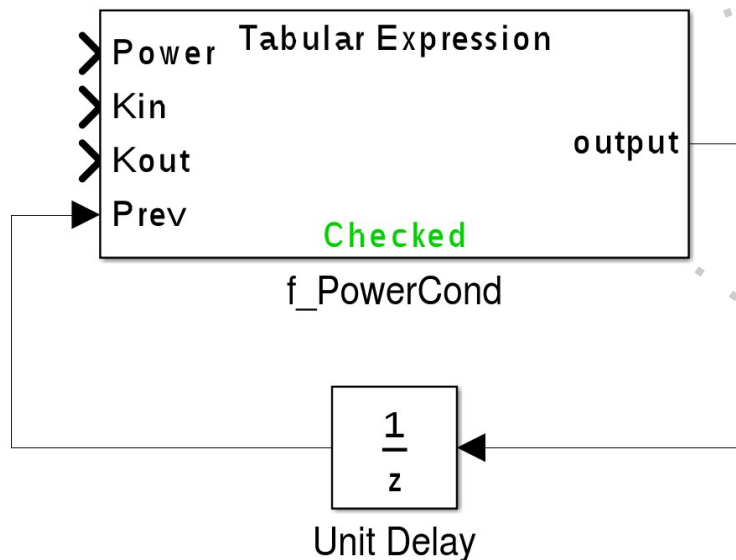
# QuantUM Tool

- Developed by Adrian Beer, Florian Leitner-Fischer, Stefan Leue at University of Konstanz
- Envisioned use: Support safety cases development in 26262
- Helps to determine hazardous states using two different model checkers for different things (not redundant checks)
- Also supports quantitative analysis using PRISIM modelchecker
  - DO-178C does not apply if you use probabilities so we assume we are not qualifying the tool to use this feature
  - Qualification for use of feature subset is explicitly considered in all the standards



# Tabular Expression Toolbox (TET)

- Create tabular expressions in Matlab/Simulink
- Verify completeness (input coverage) & disjointness (determinism) using FM tools
- Can also generate Simulink blocks or .m code for tabular expression that Matlab Coder can convert to C code

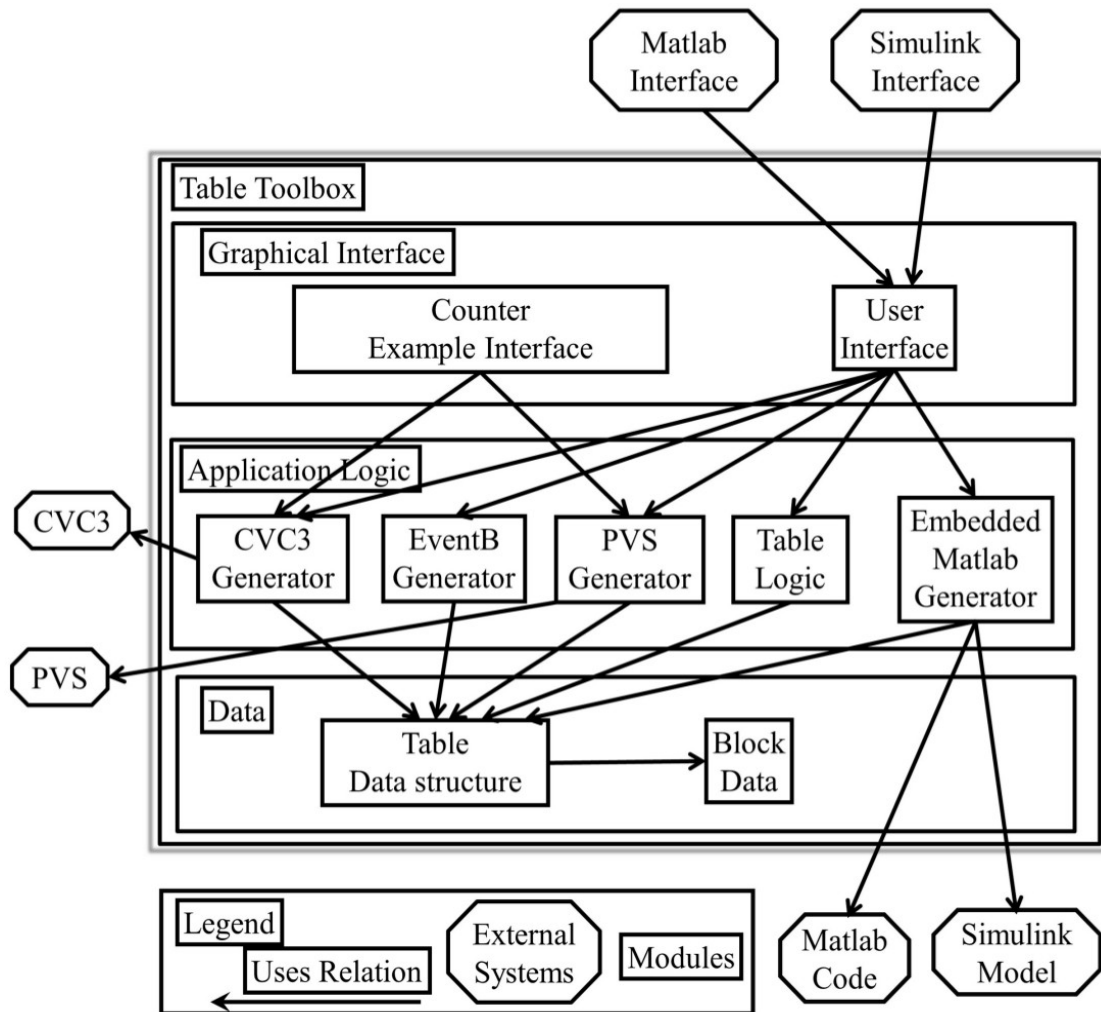


The screenshot shows the TET software interface. The main window is titled "f\_PowerCond" and contains a tabular expression editor. The inputs are defined as "Power, Kin, Kout:real, Prev:boo". The expression name is "f\_PowerCond". The table below shows the tabular expression:

Condition	Value
Power < Kout	1
Kout <= Power && Power < Kin	Prev
Kin <= Power	true

Below the table, there are three dropdown menus for the conditions, with values "false", "Prev", and "true" respectively. The "CVC3 Report" window is open, showing a "Typecheck Summary" for the expression. The summary includes the TCC Name "f\_PowerCond.cvc\_1.DIS", the sequent "QUERY (NOT ((Power < Kout) AND (Kout <= Power AND Power < Kin)) AND NOT ((Power < Kout) AND (Kin <= Power)) AND NOT ((Kout <= Power AND Power < Kin) AND (Kin <= Power)))", and a counter example: "Power = 0; Kin = 0; Kout = 0.25;".

# TET Architecture



- Completeness & disjointness checks generated 2 ways:
  1. TET->PVS Table
  2. TET->CVC3 queries
- Embedded Matlab generator only path to code
- Table Data structure is also single point failure for everything

# Do I Qualify? 1<sup>st</sup> understand “verified” in DO-178C

“Verification is a technical assessment of the outputs of the software planning process, software development processes, and the software verification process.” - DO-178C sec 6

Verification is not simply testing. Testing, in general, cannot show the absence of errors. As a result, the following sections use the term "verify" instead of "test" to discuss the software verification process activities, which are typically a combination of reviews, analyses, and tests. - DO178C s.6

# DO-178C/DO-330

An assessment on all the tools used in the framework of the tool life cycle processes should be conducted in order to identify the need for qualification of these tools. Qualification of these tools is needed when processes of this document are eliminated, reduced, or automated by the use of a tool without its output verified as specified in section 6.

DO-330 S. 4.4(e)

Tool Planning Process Activities

# DO-178C/DO-330

An assessment on all the tools used in the framework of the tool life cycle processes should be conducted in order to identify the need for qualification of these tools. Qualification of these tools is needed when processes of this document are eliminated, reduced, or automated by the use of a tool **without its output verified** as specified in section 6.

DO-330 S. 4.4(e)

Tool Planning Process Activities

# ISO 26262

## 11.4 Requirements and recommendations

### 11.4.1 General requirement

11.4.1.1 If the safety lifecycle incorporates the use of a software tool for the development of a system, or its hardware or software elements, such that activities or tasks required by ISO 26262 rely on the correct functioning of a software tool, and where the relevant outputs of that tool are not examined or verified for the applicable process step(s), such software tools shall comply with the requirements of this clause.

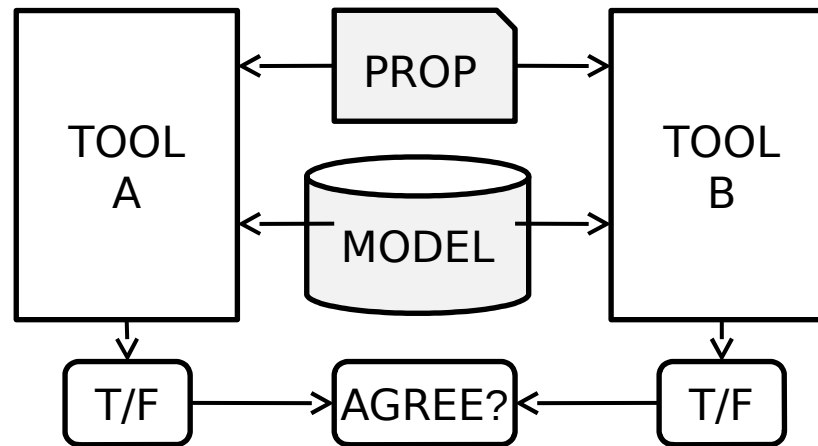
# ISO 26262

## 11.4 Requirements and recommendations

### 11.4.1 General requirement

11.4.1.1 If the safety lifecycle incorporates the use of a software tool for the development of a system, or its hardware or software elements, such that activities or tasks required by ISO 26262 rely on the correct functioning of a software tool, **and where the relevant outputs of that tool are not examined or verified for the applicable process step(s)**, such software tools shall comply with the requirements of this clause.

# No qualification required?



- Literal interpretations of ISO 26262 and DO-330 seem to indicate that no qualification is required for either tool
- Neither 26262 nor DO-178C seem to consider diversity of tools check in this case (though they do elsewhere)
- IEC 61508 still requires qualification???



# Do our tools need to be qualified?

- QuantUM
  - Yes. Checking is not redundant. Model checkers are used collaboratively not redundantly
- TET
  - Use 1: Demonstrate completeness determinism of e.g. requirements tables
    - CVC3 and PVS provide redundant checks of each other's outputs table
    - Still need to qualify “the box” - TET- but not CVC3 or PVS
  - Use 2: Code generation from tables
    - Definitely (I think)

# DO-178C

DO-178C added new **criteria** to determine the required tool qualification level (unique to aviation domain)

## Criteria

1. A tool that automates development processes (output is part of the airborne software) and thus could insert an error
2. A tool that automates verification processes and thus could fail to detect an error, **and**

whose output is used to justify the elimination or reduction of verification process(es) other than that automated by the tool, **or**

development process(es) which could have an impact on the airborne software.

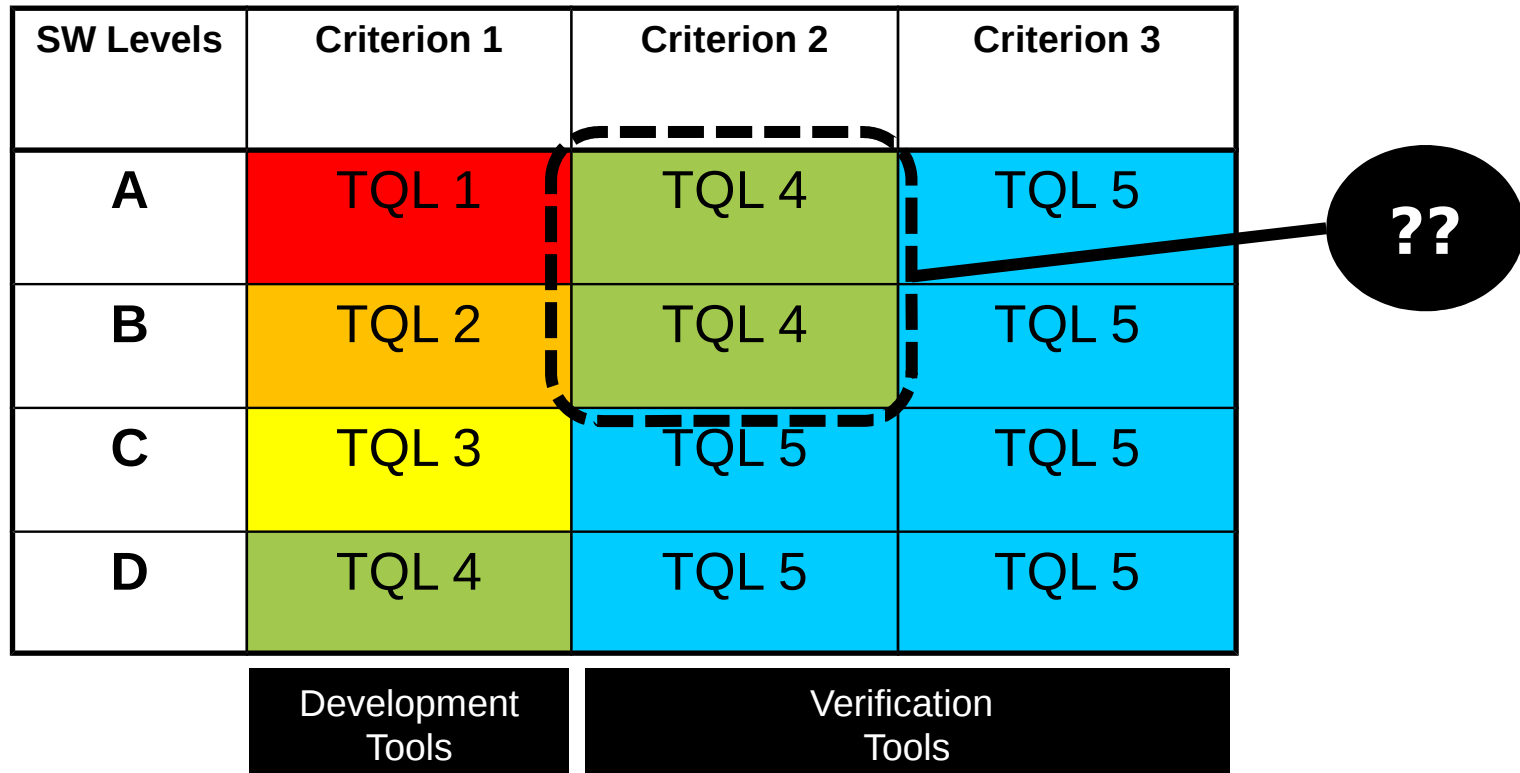
3. A tool that automates verification processes and thus could fail to detect an error

# Tool Qualification Level (TQL)

(from Darren Cofer, Rockwell-Colins)

SW Levels	Criterion 1	Criterion 2	Criterion 3
A	TQL 1	TQL 4	TQL 5
B	TQL 2	TQL 4	TQL 5
C	TQL 3	TQL 5	TQL 5
D	TQL 4	TQL 5	TQL 5

Development Tools      Verification Tools



“The problem arises when, based on the confidence of a given verification activity, some alleviation is claimed for other objectives or activities that are not the direct purpose of that verification activity.”

My example (not Darren's): Doing formal proof that code conforms to low level requirements then saying that as a result you don't need to do unit test.

# DO-178C/DO-330 TQL4 vs TQL5

(From Nick Tudor, D-RisQ)

- TQL5 has only 14 Objectives
  - 2 require QA independence
  - 4 relate to certifier liaison
  - 6 relate to Tool Operational Requirements and installation of the tool
  - Remaining 2 are configuration management
- TQL4 has the above 14 plus a further 24
  - Most extra Objectives relate to the need to have Requirements derived from Tool Operational Requirements
  - This then drives a number of other Objectives
  - Also drives the need to have extra documentation such as Requirements Standards and the reviews associated with them
- Subsequent levels pile on more Objectives

# IEC 61508-4 Tool Classification

## 3.2.11

### **software off-line support tool**

software tool that supports a phase of the software development lifecycle and that cannot directly influence the safety-related system during its run time. Software off-line tools may be divided into the following classes:

#### **– T1**

generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety related system;

NOTE 1 T1 examples include: a text editor or a requirements or design support tool with no automatic code generation capabilities; configuration control tools.

#### **– T2**

supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software;

NOTE 2 T2 examples include: a test harness generator; a test coverage measurement tool; a static analysis tool.

#### **– T3**

generates outputs which can directly or indirectly contribute to the executable code of the safety related system.

NOTE 3 T3 examples include: an optimising compiler where the relationship between the source code program and the generated object code is not obvious; a compiler that incorporates an executable run-time package into the executable code.

# ISO 26262-8 Tool Impact (TI)

11.4.5.2 The intended usage of the software tool shall be analysed and evaluated to determine:

a) the possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed. This is expressed by the classes of *Tool Impact* (TI):

- 1) TI1 shall be selected when there is an argument that there is no such possibility;
- 2) TI2 shall be selected in all other cases;

# ISO 26262-8 Tool error Detection (TD)

b) the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output. This is expressed by the classes of Tool error Detection (TD):

# ISO 26262-8 Tool error Detection (TD)

- 1) TD1 shall be selected if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected;
- 2) TD2 shall be selected if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected;
- 3) TD3 shall be selected in all other cases.



# ISO 26262 Tool Confidence Level (TCL)

**Table 3 — Determination of the tool confidence level (TCL)**

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

## 11.4.6 Qualification of a software tool

11.4.6.1 For the qualification of software tools classified at TCL3, the methods listed in Table 4 shall be applied. For the qualification of software tools classified at TCL2, the methods listed in Table 5 shall be applied. A software tool classified at TCL1 needs no qualification methods.

# ISO 26262 TCL2

Table 5 — Qualification of software tools classified TCL2

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use in accordance with 11.4.7	++	++	++	+
1b	Evaluation of the tool development process in accordance with 11.4.8	++	++	++	+
1c	Validation of the software tool in accordance with 11.4.9	+	+	+	++
1d	Development in accordance with a safety standard <sup>a</sup>	+	+	+	++

<sup>a</sup> No safety standard is fully applicable to the development of software tools. Instead, a relevant subset of requirements of the safety standard can be selected.

EXAMPLE Development of the software tool in accordance with ISO 26262, IEC 61508 or RTCA DO-178.

Similar to DO-178C, the application criticality of application (ASIL) determines qualification requirements, but 26262 TCL stays across ASIL same whereas DO-178C TQL changes with SW level

# Approximate relationship

DO-178C	IEC 61508	ISO 26262
Criterion 1	Tool class T3	TCL ?
Criterion 2	Tool class T2	TCL ?
Criterion 3	Tool class T2	TCL ?
	Tool class T1	TCL ?

QuantUM – Criteria 3, T2, TCL 3 (TI 2, TD3)

TET Use1 – Criteria 3, T2, TCL 2 (TI 2, TD2?)

TET Use2 – Criteria 1, T3, TCL 1 (TI 2, TD1)

“Easy”

“Medium”

“Hard”

# Classifying our tools

Assuming on this slide and previous:

1. “Typical” SW dev process for the standard
2. SW Level 1, SIL4, ASIL D most critical:

QuantUM – TQL 5, T2, TCL 3 (TI 2, TD3)

TET Use1 – TQL 5, T2, TCL 2 (TI 2, TD2?)

TET Use2 – TLQ 1, T3, TCL 1 (TI 2, TD1)

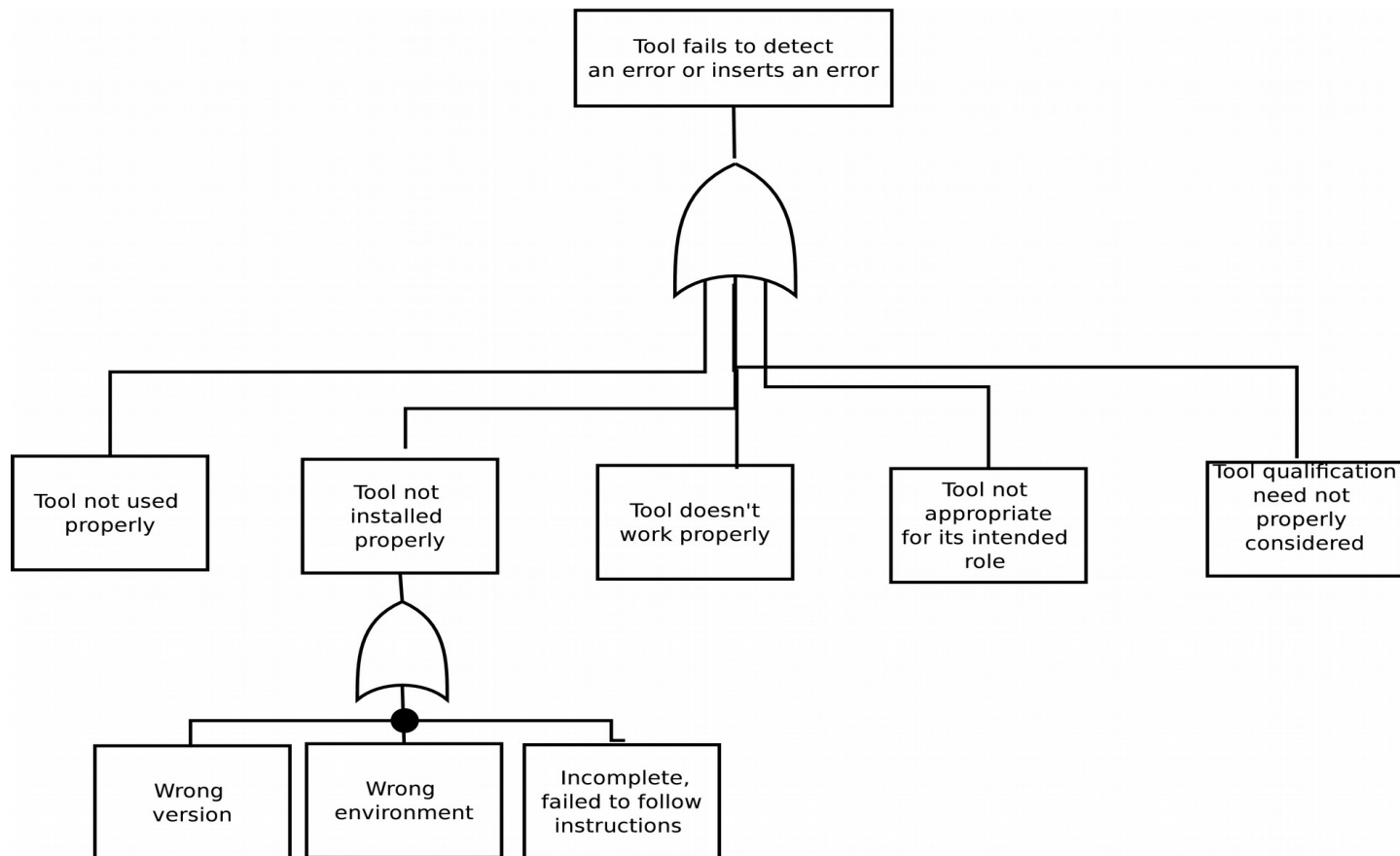
“Easy”

”Medium”

“Hard”

# Tool Hazard Analysis Template?

- Would be useful to understand how hazards they are trying to mitigate were arrived at, e.g. DO-330 might look something like:



# DO-330 TQL-5 Assurance Case

- Need for qualification at appropriate level has been done correctly
  - Tool specific info Plan for Software Aspects of Certification (PSAC) = Tool put through appropriate QA for its use
- Tool operational requirements are defined
  - Tool Operational Requirements (TOR) = Say how tool will be used and what it will produce
- Tool installed properly
  - Tool executable & Tool installation report

# DO-330 TQL-5 Assurance Case

- Tool operation complies with TOR = Tool works as specified when used as specified
  - Tool operational V&V Cases and Procedures
  - Tool operational V&V Results
- TOR Sufficient & correct
  - Spec of how will be tool used & what it must do is correct and sufficient
- Software Lifecycle process needs met by tool=Tool doing what is required of it by SW process
  - Tool operational V&V Cases and Procedures
  - Tool operational V&V Results

# DO-330 TQL-5 Assurance Case

- Tool Config identified
  - Tool config management records
- Tool config properly done
  - Tool config management records



# DO-330 TQL-5 Assurance Case

- You assure Tool processes comply with approved plans= Your tool processes correspond to that was planned
  - Tool Quality Assurance Record
- Tool conformity review is done= You check that the tool is being used correctly & producing what you need
  - Tool Quality Assurance record

# Connections to talks so far

- John Goodenough – Hazards to evidence
  - Seems to correspond somewhat to 26262-8 TD
  - IEC 61508-3 7.4.4.5:

An assessment shall be carried out for offline support tools in classes T2 and T3 to determine the level of reliance placed on the tools, and the potential failure mechanisms of the tools that may affect the executable software. Where such failure mechanisms are identified, appropriate mitigation measures shall be taken.

Questions?

# Questions for Rick

- Why do regulatory industry reps hate it?
- Why do engineers like it?
- Has assurance case related infusion pump guidance helped make reviews easier?
- Robert Wachter, The digital doctor
- 21<sup>st</sup> century cures legislation
- 70,000 adverse event reports involving infusion pumps in the last 5 years