AESSCS Workshop, Newcastle upon Tyne UK, 13 May 2014

Evaluating
The Assessment of Software Fault-Freeness

John Rushby

SRI International, Menlo Park CA USA

Bev Littlewood and Lorenzo Strigini

City University, London UK

What Do Standards Do?

- Encourage good development process
  - e.g., high-quality requirements
  - Ideally, prevents the introduction of faults
- Require assessment of the product
  - e.g., static analysis, MC/DC testing
  - Ideally, detects many/most/all faults
- But the quality required in safety-critical software (e.g., flight control) is so great that we do not expect to detect any faults at final assessment, nor to see any failures in operation
  - e.g., Catastrophic failure conditions: not expected to occur in the entire lifetime of all airplanes of one type
- So what standards (and operational experience) provide is evidence for the absence of faults
- How does this support certification?
- And how can we measure it?

Larger Hypothesis

- Before we can frame testable hypotheses about standards
- We need to posit a larger hypothesis that evidence for absence of faults provides a quantifiable basis for certification

How Does Assurance Relate To Reliability?

- Top level requirements are stated as reliability measures
  - e.g., failure condition of severity XX not expected to occur in YY hours/flights
  - Inverse relationship between severity and likelihood
- We do more assurance for software that could contribute to or cause higher failure severities
- e.g., DO-178C identifies five Software Levels (associated with failure severities) and 71 assurance objectives
  - 26 objectives at DO178C Level D (10^{-3})
  - 62 objectives at DO178C Level C (10^{-5})
  - 69 objectives at DO178C Level B (10^{-7})
  - 71 objectives at DO178C Level A (10^{-9})

  There are also independence requirements at higher levels
- How does doing more of these correctness-based objectives relate to lower probability of failure?

Confidence in Fault-Freeness

- Assurance makes us confident
- So more assurance makes us...
  - Confident in fewer faults, or
  - More confident in some given rarity of faults
- The last of these is what works
  - Specifically, zero faults (aka. perfection, fault-freeness)
- Degree of confidence that the software is fault-free is expressed as a probability: P(s/w fault-free)

Relationship Between Fault-Freeness and pfd

- By the formula for total probability

$$
\begin{aligned}
& P(\text{s/w fails [on a randomly selected demand]}) \qquad (1) \\
&= P(\text{s/w fails} \,|\, \text{s/w fault-free}) \times P(\text{s/w fault-free}) \\
&\quad + P(\text{s/w fails} \,|\, \text{s/w faulty}) \times P(\text{s/w faulty}).
\end{aligned}
$$

- The first term in this sum is zero
  - Because the software does not fail if it is fault-free
  - Which is why the theory needs this property
- Define p_{nf} as the probability the software is fault-free
  - Or nonfaulty
  - So that $P(\text{s/w faulty}) = 1 - p_{nf}$
- And define $p_{F|f}$ as the probability that it Fails, if faulty
- Then $pfd = p_{F|f} \times (1 - p_{nf})$

Relationship Between Fault-Freeness and Survival

- More importantly, $p_{srv}(n)$, the probability of surviving n independent demands (e.g., flights) without failure is given by

$$p_{srv}(n) = p_{nf} + (1 - p_{nf}) \times (1 - p_{F|f})^n \qquad (2)$$

- A suitably large n can represent "the entire lifetime of all aircraft of one type"
  - A320 series has had over 62 million flights to date, so n will be about 10^8 or 10^9
- First term in (2) establishes a lower bound for $p_{srv}(n)$ that is independent of n
- If assurance gives us the confidence to assess $p_{nf} > 0.99$
  - Or whatever threshold "not expected to occur" means
- Then it looks like we have sufficient evidence to certify the aircraft as safe (with respect to software aspects)

But What If The Software Does Have Faults?

- In this case, we need confidence that the second term in (2) will be well above zero, despite exponential decay
- Confidence could come from prior failure-free operation
- Calculating overall $p_{srv}(n)$ is a problem in Bayesian inference
  - We have assessed a value for p_{nf}
  - Have observed some number r of failure-free demands
  - Want to predict prob. of $n - r$ future failure-free demands
- Need a prior distribution for $p_{F|f}$
  - Difficult to obtain, and difficult to justify for certification
  - However, there is a distribution that delivers provably worst-case predictions
    - ⋆ One where $p_{F|f}$ is a prob. mass at some $q_n \in (0, 1]$
  - So can make predictions that are guaranteed conservative, given only p_{nf}, r, and n

Take Home Message

- For values of p_{nf} above 0.9
- $p_{srv}(n)$ is well above the floor given by p_{nf}
- Provided $r > \frac{n}{10}$
- So it looks like we need to fly 10^8 hours to certify 10^9
- No!
- Entering service, we have only a few planes, need confidence for only, say, first six months of operation
- Flight tests are enough for this
- Next six months, have more planes, but can base prediction on first six months (or ground the fleet, fix things)
- And bootstrap our way forward
- We think this is the first scientific explanation of how software certification actually works
- It provides a model that is consistent with practice

Experiments

- Objective is to validate our model
- Populate it with credible parameters
  - See if the overall numbers work
  - See if certifiers believe it
  - Then use it to improve current practice
- Three parameters: p_{nf}, r, and n, only the first is difficult
- Two approaches for a preliminary check
  - Consider how many such systems have been in use and never exhibited failures
  - Ask certifiers what p_{nf}, cast in a frequentist interpretation, they might assess (next page)

Both approaches have (different) weaknesses

Initial Experiments

- Typical question: "given 100 software systems assessed to have accomplished all 7 objectives of DO-178C Section 6.3.2, how many of those systems do you believe might ever suffer a software failure due to flawed low level requirements?"

- To do this well, need the argument for the different objectives and sections of DO-178C
  - Michael Holloway's Explicate'78 project provides this

- Can then construct a first-cut argument
  - E.g., using Bayesian Belief Nets and suitable conservative simplifications

  To yield assessment of p_{nf} for the whole of DO-178C

Further Development and Applications

- Refine the model
  - E.g. Using historical data about individual methods
  - Or *a priori* estimates based on analysis of the argument supporting each cluster of objectives

  Experiments of other participants would supply these

- Explore modified objectives
  - For lower cost or increased confidence

- Evaluate alternative means
  - E.g., software monitors, explicitly designed for high p_{nf}
  - p_{nf} of the monitor is conditionally independent of reliability of the primary and yields multiplicative increase in overall reliability
  - That's an aleatoric result, epistemic applic'n needs care