On the reliability of Monitored Systems

John Rushby
Based on joint work with Bev Littlewood (City University UK)

Computer Science Laboratory
SRI International
Menlo Park CA USA

A Conundrum

- Critical systems are those where failures can have unacceptable consequences: typically safety or security
- Cannot eliminate failures with certainty (because the environment is uncertain), so top-level claims about the system are stated quantitatively
  - E.g., no catastrophic failure in the lifetime of all airplanes of one type ("in the life of the fleet")
- And these lead to probabilistic requirements for software-intensive subsystems
  - E.g., probability of failure in flight control $< 10^{-9}$ per hour
- To assure this, do lots of verification and validation (V&V)
- But V&V is all about showing correctness
- And for stronger claims, we do more V&V
- So how does amount of V&V relate to probability of failure?

Background

The Basis For Assurance and Certification

- We have claims or goals that we want to substantiate
  - Typically claims about a critical property such as security or safety
  - Or some functional property, or a combination

  E.g., no catastrophic failure condition in the life of the fleet

- We produce evidence about the product and its development process to support the claims
  - E.g., analysis and testing of the product and its design
  - And documentation for the process of its development

- And we have an argument that the evidence is sufficient to support the claims

- Surely, this is the intellectual basis for all certification regimes

Standards-Based Certification vs. Safety Cases

- Applicant follows a set process, delivers prescribed outputs
  - e.g., documented requirements, designs, analyses, tests

  These provide evidence; goals and argument largely implicit
- Common Criteria (security), DO-178B (civil aircraft) do this
- Works well in fields that are stable or change slowly
  - Can institutionalize lessons learned, best practice
- May be less suitable with novel problems, solutions, methods
- Alternative is a safety case: applicant
  - Makes an explicit set of goals or claims
  - Provides supporting evidence for the claims
  - And arguments that link the evidence to the claims
- The case is evaluated by independent assessors
- The main novelty is the explicit argument
- Generalized to security, dependability, assurance cases

Software Reliability

- Software contributes to system failures through faults in its requirements, design, implementation—bugs
- A bug that leads to failure is certain to do so whenever it is encountered in similar circumstances
  - There's nothing probabilistic about it
- Aaah, but the circumstances of the system are a stochastic process
- So there is a probability of encountering the circumstances that activate the bug
- Hence, probabilistic statements about software reliability or failure are perfectly reasonable
- Typically speak of probability of failure on demand (pfd), or failure rate (per hour, say)

Aleatory and Epistemic Uncertainty

- Aleatory or irreducible uncertainty
  - is "uncertainty in the world"
  - e.g., if I have a coin with $P(heads) = p_h$, I cannot predict exactly how many heads will occur in 100 trials because of randomness in the world

  Frequentist interpretation of probability needed here

- Epistemic or reducible uncertainty
  - is "uncertainty about the world"
  - e.g., if I give you the coin, you will not know p_h; you can estimate it, and can try to improve your estimate by doing experiments, learning something about its manufacture, the historical record of similar coins etc.

  Frequentist and subjective interpretations OK here

Aleatory and Epistemic Uncertainty in Models

- In much scientific modeling, the aleatory uncertainty is captured conditionally in a model with parameters
- And the epistemic uncertainty centers upon the values of these parameters
- As in the coin tossing example: p_h is the parameter

Back To The Main Thread

Measuring/Predicting Software Reliability

- For pfds down to about 10^{-4}, it is feasible to measure software reliability by statistically valid random testing
- But 10^{-9} would need 114,000 years on test
- So how do we establish that a piece of software is adequately reliable for a system that requires, say, 10^{-6}?
- Standards for system security or safety (e.g., Common Criteria, DO178B) require you to do a lot of V&V
  - e.g., 57 V&V "objectives" at DO178B Level C (10^{-5})
- And you have to do more for higher levels
  - 65 objectives at DO178B Level B (10^{-7})
  - 66 objectives at DO178B Level A (10^{-9})
- What's the connection between amount of V&V (mostly focused on correctness) and degree of software reliability?

Aleatory and Epistemic Uncertainty for Software

- The amount of correctness-based V&V relates poorly to reliability
- Maybe it relates better to some other probabilistic property of the software's behavior
- We are interested in a property of its dynamic behavior
    - There is aleatoric uncertainty in this property due to variability in the circumstances of the software's operation
- We examine the static attributes of the software to form an epistemic estimate of the property
    - More examination refines the estimate
- For what kinds of properties could this work?

Perfect Software

- Property cannot be about some executions of the software
  - Like how many fail
  - Because the epistemic examination is static (i.e., global)
  - This is the disconnect with reliability
- Must be a property about all executions, like correctness
- But correctness is relative to specifications, which themselves may be flawed
- We want correctness relative to the critical claims
  - Taken directly from the system's assurance case
- Call that perfection
- Software that will never experience a failure in operation, no matter how much operational exposure it has

Possibly Perfect Software

- You might not believe a given piece of software is perfect
- But you might concede it has a possibility of being perfect
- And the more V&V it has had, the greater that possibility
- So we can speak of a (subjective) probability of perfection
- For a frequentist interpretation: think of all the software that might have been developed by comparable engineering processes to solve the same design problem
    - And that has had the same degree of V&V
    - The probability of perfection is then the probability that any software randomly selected from this class is perfect

Probabilities of Perfection and Failure

- Probability of perfection relates to correctness-based V&V
- But it also relates to reliability:

  By the formula for total probability

$$
\begin{aligned}
& P(\text{s/w fails [on a randomly selected demand]}) \qquad (1) \\
= \;& P(\text{s/w fails} \mid \text{s/w perfect}) \times P(\text{s/w perfect}) \\
& + P(\text{s/w fails} \mid \text{s/w imperfect}) \times P(\text{s/w imperfect}).
\end{aligned}
$$

- The first term in this sum is zero, because the software does not fail if it is perfect (other properties won't do)
- Hence, define
  - p_{np} probability the software is imperfect
  - p_{fnp} probability that it fails, if it is imperfect
- Then $P(\text{software fails}) \leq p_{fnp} \times p_{np}$
- This analysis is aleatoric, with parameters p_{fnp} and p_{np}

Epistemic Estimation

- To apply this result, we need to assess values for p_{fnp} and p_{np}
- These are most likely subjective probabilities
  - i.e., degrees of belief
- Beliefs about p_{fnp} and p_{np} may not be independent
- So will be represented by some joint distribution $F(p_{fnp}, p_{np})$
- Probability of software failure will be given by the Riemann-Stieltjes integral

$$\int_{\substack{0 \le p_{fnp} \le 1 \\ 0 \le p_{np} \le 1}} p_{fnp} \times p_{np} \, dF(p_{fnp}, p_{np}). \tag{2}$$

- If beliefs can be separated F factorizes as $F(p_{fnp}) \times F(p_{np})$
- And (2) becomes $P_{fnp} \times P_{np}$

  Where these are the means of the posterior distributions representing the assessor's beliefs about the two parameters

Practical Application—Nuclear

- Traditionally, nuclear protection systems are assured by statistically valid random testing
- Very expensive to get to pfd of 10^{-4} this way
- Our analysis says pfd $\leq P_{fnp} \times P_{np}$
- They are essentially setting P_{np} to 1 and doing the work to assess $P_{fnp} < 10^{-4}$
- Any V&V process that could give them $P_{np} < 1$
- Would reduce the amount of testing they need to do
  - e.g., $P_{np} < 10^{-1}$, which seems very plausible
  - Would deliver the the same pfd with $P_{fnp} < 10^{-3}$
- This could reduce the total cost of assurance

Practical Application—Aircraft, Version 1

- No plane crashes due to software, and enough operational exposure to validate software failure rate $< 10^{-9}$
- Aircraft software is assured by V&V processes such as DO-178B Level A
- They do a massive amount of all-up testing but do not take assurance credit for this
- Our analysis says software failure rate $\leq P_{fnp} \times P_{np}$
- So they are setting $P_{fnp} = 1$ and $P_{np} < 10^{-9}$
- Littlewood and Povyakalo show (under independence assumption) that large number of failure-free runs shifts assessment from imperfect but reliable toward perfect
- So flight software might indeed have probabilities of imperfection $< 10^{-9}$
- And DO-178B delivers this

Practical Application—Aircraft, Version 2

- Although no crashes due to software, there have been several incidents
- So actual failure rate may be only around 10^{-7}
- Although they don't take credit for all the testing they do, this may be where a lot of the assurance is really coming from
- Our analysis says software failure rate $\leq P_{fnp} \times P_{np}$
- So perhaps testing is implicitly delivering, say, $P_{fnp} < 10^{-3}$
- And DO-178B is delivering only $P_{np} < 10^{-4}$
- I do not know which of Version 1 or 2 is true
- But there are provocative questions here

Two Channel Systems

- Many safety-critical systems have two (or more) diverse "channels" arranged in 1-out-of-2 (1oo2) structure
  - E.g., nuclear shutdown
- A primary protection system is responsible for plant safety
- A simpler secondary channel provides a backup
- Cannot simply multiply the pfds of the two channels to get pfd for the system
  - Failures are unlikely to be independent
  - E.g., failure of one channel suggests this is a difficult case, so failure of the other is more likely
  - Infeasible to measure amount of dependence

So, traditionally, difficult to assess the reliability delivered

Two Channel Systems and Possible Perfection

- But if the second channel is simple enough to support a plausible claim of possible perfection
  - Its imperfection is conditionally independent of failures in the first channel at the aleatory level
  - Hence, system pfd is conservatively bounded by product of pfd of first channel and probability of imperfection of the second
  - $P(\text{system fails on randomly selected demand} \le pfd_A \times pnp_B$

  This is a theorem

- Epistemic assessment similar to previous case
  - But may be more difficult to separate beliefs
  - Conservative approximations are available

Details: Aleatory Uncertainty for 1oo2 Architectures

$$
\begin{aligned}
&P(\text{system fails [on randomly selected demand]} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&= \quad P(\text{system fails} \mid A \text{ fails}, B \text{ imperfect}, \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&\qquad\qquad \times P(A \text{ fails}, B \text{ imperfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&\qquad + P(\text{system fails} \mid A \text{ succeeds}, B \text{ imperfect}, \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&\qquad\qquad \times P(A \text{ succeeds}, B \text{ imperfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&\qquad + P(\text{system fails} \mid A \text{ fails}, B \text{ perfect}, \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&\qquad\qquad \times P(A \text{ fails}, B \text{ perfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&\qquad + P(\text{system fails} \mid A \text{ succeeds}, B \text{ perfect}, \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
&\qquad\qquad \times P(A \text{ succeeds}, B \text{ perfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B)
\end{aligned}
$$

Assume, conservatively, that if A fails and B is imperfect, then B will fail on the same demand

$$
\leq \quad 1 \times P(A \text{ fails}, B \text{ imperfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) + 0 + 0 + 0
$$

Aleatory Uncertainty for 1oo2 Architectures (ctd.)

$$
\begin{aligned}
& P(A\,\text{fails}, B\,\text{imperfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
& \quad = \quad P(A\,\text{fails} \mid B\,\text{imperfect}, \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
& \qquad\qquad \times P(B\,\text{imperfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B)
\end{aligned}
$$

(Im)perfection of B tells us nothing about the failure of A on this demand; hence,

$$
\begin{aligned}
& = \quad P(A\,\text{fails} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
& \qquad\qquad \times P(B\,\text{imperfect} \mid \mathit{pfd}_A = p_A, \mathit{pnp}_B = p_B) \\
& = \quad p_A \times p_B
\end{aligned}
$$

Compare with two (un)reliable channels, where failure of B on this demand does increase likelihood A will fail on same demand

$$
\begin{aligned}
& P(A\,\text{fails} \mid B\,\text{fails}, \mathit{pfd}_A = p_A, \mathit{pfd}_B = p_B) \\
& \quad \geq \quad P(A\,\text{fails} \mid \mathit{pfd}_A = p_A, \mathit{pfd}_B = p_B)
\end{aligned}
$$

Aleatory Uncertainty for 1oo2 Architectures (ctd. 2)

I could have factored the conditional probability involving the perfect channel the other way around:

$$\begin{aligned} & P(A \,\text{fails}, B \,\text{imperfect} \mid pfd_A = p_A, pnp_B = p_B) \\ = \quad & P(B \,\text{imperfect} \mid A \,\text{fails}, pfd_A = p_A, pnp_B = p_B) \\ & \times P(A \,\text{fails} \mid pfd_A = p_A, pnp_B = p_B) \end{aligned}$$

You might say knowledge that A has failed should affect my estimate of B's imperfection, but we are dealing with aleatory uncertainty where these probabilities are known; hence

$$\begin{aligned} = \quad & P(B \,\text{imperfect} \mid pfd_A = p_A, pnp_B = p_B) \\ & \times P(A \,\text{fails} \mid pfd_A = p_A, pnp_B = p_B) \\ = \quad & p_B \times p_A \text{ as before} \end{aligned}$$

Note: the claim must be perfection, other global properties (e.g., proven correct) are not aleatory (they are reducible)

Epistemic Uncertainty for 1oo2 Architectures

- We have shown that the events "A fails" "B is imperfect" are conditionally independent at the aleatory level
- Knowing aleatory probabilities of these allows probability of system failure to be conservatively bounded by $p_A \times p_B$
- But we do not know p_A and p_B with certainty: assessor formulates beliefs about these as subjective probabilities
- The beliefs may not be independent, so they will be represented by a joint probability density function $dF(p_A, p_B) = P(pfd_A < p_A, pnp_B < p_B)$
- The unconditional probability of system failure is then

$$P(\text{system fails on randomly selected demand})$$

$$= \int_{\substack{0 \le p_A \le 1 \\ 0 \le p_B \le 1}} p_A \times p_B \, dF(p_A, p_B)$$

(That's a Riemann-Stieltjes integral)

Reliability Estimate for 1oo2 Architectures

- The only source of dependence is in the assessor's bivariate density function $dF(p_A, p_B)$
- But it is really hard to elicit such bivariate beliefs
- What stops beliefs about the two parameters being independent?
- It's not difficulty variation over the demand space
  - Formal verification is uniformly credible
- Surely, it's concern about common-cause errors such as misunderstood requirements, common mechanisms, etc.
- So combine all beliefs about common-cause faults in a third parameter C
  - Place probability mass C at point $(1, 1)$ in (p_A, p_B)-plane as subjective probability for such common faults

Reliability Estimate for 1oo2 Architectures (ctd.)

- With probability C, A will fail with certainty, and B will be imperfect with certainty (and conservatively assumed to fail)
- If assessor believes all dependence between his beliefs about the model parameters has been captured conservatively in C, the conditional distribution factorizes, so

$$P(\text{system fails on randomly selected demand})$$

$$= C + (1 - C) \times \int_{0 \le p_A < 1} p_A \, dF(p_A) \times \int_{0 \le p_B < 1} p_B \, dF(p_B)$$

$$= C + (1 - C) \times P_A^* \times P_B^*$$

where P_A^* and P_B^* are the means of the marginal distributions excluding $(1, 1)$

Reliability Estimate for 1oo2 Architectures (ctd. 2)

- If C is small (as will be likely), can approximate as

$$C + P_A \times P_B$$

  where P_A and P_B are the means of the marginal distributions

- Construct probability C by considering top-level development
  - Or by claim limits (10^{-5})
- Construct probability P_A by statistically valid random testing (10^{-3})
- Construct probability P_B by considering mechanically checked formal verification (10^{-3})
- Hence overall system *pfd* is about 1.1×10^{-5}

Type 1 and Type 2 Failures in 1oo2 Systems

- So far, considered only failures of omission
  - Type 1 failure: both channels fail to respond to a demand
- Must also consider failures of commission
  - Type 2 failure: either channel responds to a nondemand
- Demands are events at a point in time; nondemands are absence of demands over an interval of time
- So full model must unify these
- Details straightforward but lengthy

Monitored Architectures

- One operational channel does the business
- Simpler monitor channel can shut it down if things look bad
- Used in airplanes
- Analysis is a variant of 1oo2:
  - No Type 2 failures for operational channel
- Monitored architecture risk per unit time
  $$\leq c_1 \times (M_1 + F_A \times P_{B1}) + c_2 \times (M_2 + F_{B2|np} \times P_{B2})$$
  where the Ms are due to mechanism shared between channels
- May provide justification for some of the architectures suggested in ARP 4754
  - e.g., 10^{-9} system made of Level C operational channel and Level A monitor

Monitors Do Fail

- Fuel emergency on Airbus A340-642, G-VATL, 8 February 2005
  - Type 1 failure
- EFIS Reboot during spin recovery on Airbus A300 (American Airlines Flight 903), 12 May 1997
  - Type 2 failure
- Current proposals are for formally synthesized/verified monitors for properties in the safety case

Conclusion

- Probability of perfection is a radical and valuable idea
  - It's due to Bev Littlewood, and Lorenzo Strigini
- Provides the bridge between correctness-based verification activities and probabilistic claims needed at the system level
- Explains what software assurance is
- Asymmetric 1oo2 systems, and monitored systems are plausible ways to achieve high reliability
- With a possibly perfect channel they also provide a credible way to assess it
- Risk of failures of commission (false alarms) requires careful consideration and engineering: for formal monitors, focus should be on choice of monitored properties