



## City Research Online

### City, University of London Institutional Repository

---

**Citation:** Strigini, L. ORCID: 0000-0002-4246-2866 (2014). Assessment techniques, certification and [what else we need for] confidence in software. Keynote Paper presented at the WoSoCer 2014, 3-6 Nov 2014, Naples, Italy.

This is the presentation version of the paper.

This version of the publication may differ from the final published version.

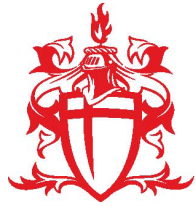
---

**Permanent repository link:** <https://openaccess.city.ac.uk/id/eprint/26125/>

**Link to published version:**

**Copyright:** City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

**Reuse:** Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.



CITY UNIVERSITY  
LONDON

# Assessment techniques, certification and *[what else we need for]* confidence in software

Lorenzo Strigini  
Centre for Software Reliability  
City University London, U.K.

**CSR** Building confidence in  
a computerised world

[www.csr.city.ac.uk](http://www.csr.city.ac.uk)

# Outline

Topic: assessment, certification, regulations and standards for all this

- their effects: desired, desirable, likely, undesired: what can certification be about and what does it prove?
- a couple of ways that certification of critical software may give quantifiable confidence
  
- various disclaimers
  - a view from a non-certifier, observing practice and claims for close to 30 years
  - skewed towards safety, and very stringent requirements

# Certification and its roles

- “certify”, from Latin, “to make certain”
- in common use, attestation by someone trustworthy that a certain statement is true to the best of his (expert!) knowledge
- e.g. a medical certificate
- for software, a range of stated meanings, e.g.
  - “to perform a set of checks on reusable components in order to guarantee that they are *error-free* (or at least without *major* error) and that their reuse will not lead to *problems*..” [Coulange 1998]
  - “planned and systematic set of activities that ensure that software life cycle *processes* and *products* conform to requirements, *standards*, and *procedures*.” [IEEE 1992]
  - “process for demonstrating that *system safety* is *satisfactory* for flight operation” [NASA]

## **There is a consensus (?) that confidence that can be had from certification is limited**

“Certification is not a proof of correctness. Accepted tests and reviews, as well as widespread usage are factors contributing to an increased confidence that the software has been subjected to a qualified development”

*Swedish Armed Forces (2005) M7762-000621-7*

*handbook for software in safety-critical applications, cited  
[Kornecki & Zalewski 2009]*

# Continuing discontent, and good news

- various criticism:
  - too much emphasis on *process*, too little on *product*
  - lack of evidence that the prescribed practices achieve the intended results (or even have a bearing on them)
  - negative effects
    - + prescriptive regimes may encourage box-ticking mind set, complacency
    - + “safety cases may suffer from ad-hocness, confirmation bias”
- all this doubt is good: it has driven adoption and/or improvement of useful techniques, e.g.
  - proof
    - + e.g. model checking tools suitable for industrial use
  - fault injection
    - + e.g. progress in injecting faults that are *realistic*
  - extensive study of *arguments* and *cases*
    - + seeking ways to make sense of complex evidence to understand/communicate the strength of support for the claims made

## Certification may achieve...

- certainty or high probability that certain activities, seen as “due diligence”, have been applied [properly/competently]
- “independent“ confirmation that certain checks performed yield certain encouraging results
- desired: strong, justified confidence that the software will satisfy certain requirements
  - including pretty extreme ones, in the “ultra-high reliability” class
  - e.g. (aviation): for catastrophic failures, “failure must be unlikely to happen over the lifetime of a type”  
(the “ $10^{-9}$ “ requirement)
  - many reasons for believing current practices do *not* support this
  - yet, there are cases of apparent *success*, e.g. civil aviation
    - + few software-related incident and accidents
    - + but sources of doubt... e.g. some evidence that stricter practice does not get better results
    - + is “success” really causation or just correlation?
      - both certification practice and high safety caused by this being a “good” industry (safety politically important, well funded)

# Certification practices may be *harmful* ...

- shifting of responsibility (legal or psychological)
  - to certifier or to prescriptive standard
    - + effect akin to using safety interlock for normal operation
    - + e.g. novel application/technology: eager certifiers may apply old existing standards, adopters feel safe that they use certified components
- consensus on “the best *now* known, *feasible* practice”
  - may become standard of “all that is needed”
  - may de-incentivise inventing something better
- degradation of culture: aim for certification, not safety
- ritualization: “accepted” assumptions, need for reproducibility may trump criticism, true challenge to claims

a good read: When the Chick Hits the Fan - Representativeness and Reproducibility in Technological Tests by John Downer



## What else? (or: the baby vs the bath water)

- even though certification practices may fail at *ensuring* that products satisfy stated requirements
- it may achieve other desirable results:
  - *more adequate products than if these practices were not applied*
  - better functioning of markets?
  - barriers to entry against incompetent vendors
  - avoiding use of definitely inadequate products
  - cultural effect: promote safety/quality/security focus
  - policing effect (e.g. vendor management cannot cut assurance activities if they are part of dominant certification practice)
  - giving something in return for applying “good” practice, even if real returns on safety/dependability/ ... are unknown
- different “purposes” (societal goods) -> different requirements on certification practices
- improvements to practice should avoid disrupting good effects

*An interesting workshop: AESSCS 2014: Planning the Unplanned Experiment: Assessing the Efficacy of Standards for Safety Critical Software, April 2014*

## Back to the “main” goal

- proving that the software can be used without undue danger
  - (for *safety* certification: otherwise without undue downtime, undue costs... etc.)
  - e.g. that failures of a certain class of severity are less likely than  $10^{-x}$  to happen (e.g. per mission, per demand, per hour, per deployed item, ..)

# What can be certified

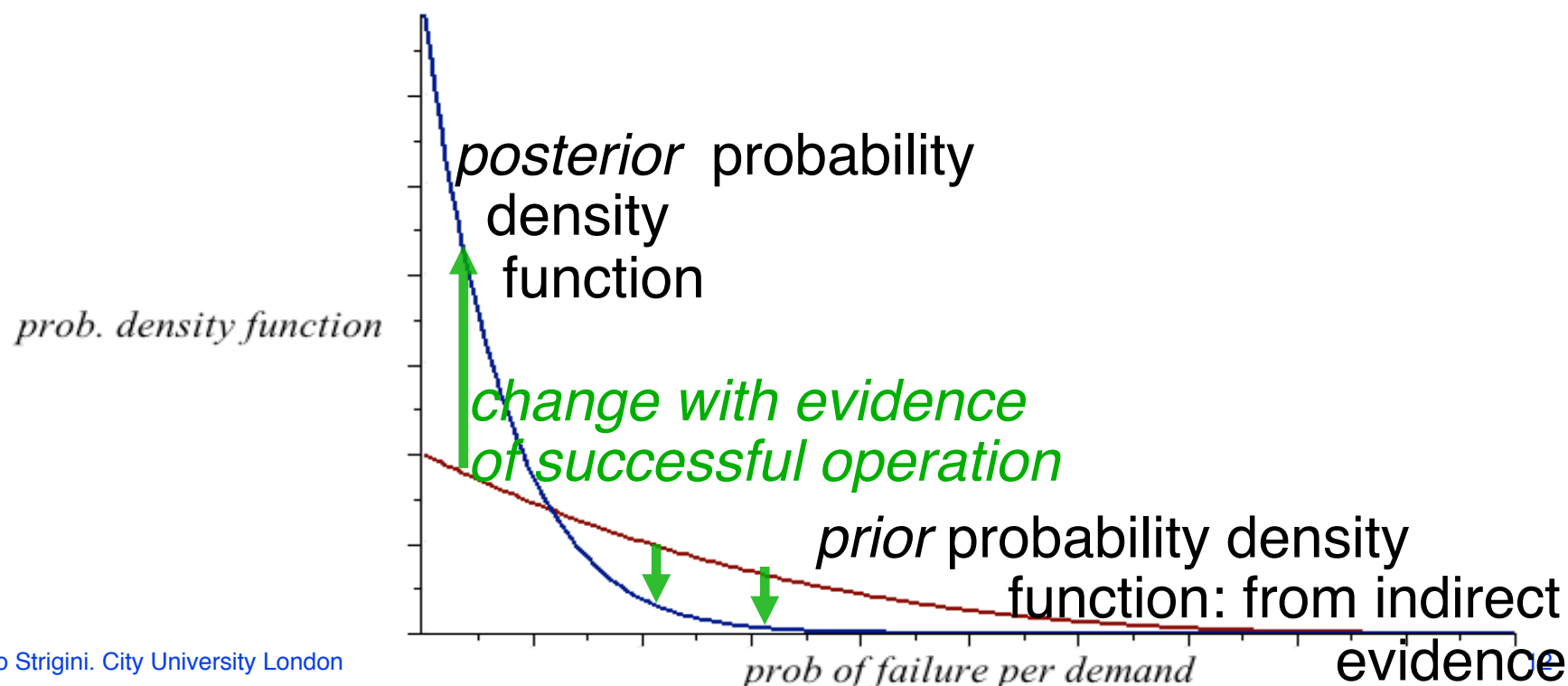
- process applied: especially conformance to recommendations/prescriptions by standards
  - + for that category of product
  - + for that criticality of function
- results of applying checks to *this* product
  - esp. those recommended by standards [...]
- behaviour observed
  - in real or simulated (operational testing) operation
  - the only direct evidence that software will behave acceptably

# Current practice suffers from inconsistency

- privileging *indirect evidence*
  - e.g., process quality , logs of V&V activities
- ... mismatched to quantitative *claims* like low probability of dangerous failure
  - the evidence is about methods applied for *having fewer faults*
  - methods *which are not known to seek out faults with high probability of failure occurrence*
  - reducing number of faults may NOT greatly reduce probability of failure!

# What indirect evidence really proves

- it (probably!) increases the *probability* that the product is satisfactorily dependable (e.g.  $<10^{-x}$  probability of dangerous failures)
- then improved further by operation / operational test (with *actual* future profile of use), through Bayesian inference



# Aleatory vs epistemic uncertainty

## ***aleatory*** uncertainty:

- given this product and this environment of use, we cannot say *when* one of the failures of interest will happen
- due to “inherent randomness in the world”:  
unpredictability of when those specific rare circumstances will occur that cause those failures
- although there is a *probability* of their occurring

## ***epistemic*** uncertainty:

- we don't know this probability: it might be  $10^{-6}$ ,  $10^{-5}$ , 0.00034....
  - each with different probability
  - perhaps unlikely to be  $> 0.1$ , more likely  $< 10^{-4}$ ...
  - so it is a random variable with a probability distribution
  - the more evidence, the more we can reduce the uncertainty
- .... what certification is about!

## Two ways that current practices may “work”

.. towards high confidence in required software dependability

1. given effective enough development/ V&V practices, the product has a reasonable probability of having probability of dangerous failure as low as required
  - and in particular, being *free* from dangerous defects
    - + thus operational testing and operation build up confidence that *either* it is so or, if not, it does not have *very* high probability of failure
2. given strong run-time mitigation measures, monitoring and safety management, any dangerous defect has a chance of revealing itself and being rectified before any accident occurs

both forms of reasoning considered reasonable by some vendors/ regulators... for formal mathematical structure, see e.g. the two papers by [Povyakalo and Strigini](#), and by [Bishop](#) in SAFECOMP 2013

# Is absence of faults plausible?

in many cases, yes

- limited to faults that are safety-relevant
- given *simplicity* of systems and of safety requirements
- strong verification (including proof) can exclude certain defects, almost with certainty
- indeed, most evidence now used is only pertinent to probabilistic claim when claiming ***absence of defects!***
  - obviously *with some residual doubt* (***confidence <100%***)



## Is absence of faults demonstrable?

- indeed, most indirect evidence now used is only pertinent to probabilistic claim when claiming **absence of defects!**
  - obviously *with some residual doubt* (**confidence <100%**)
- how to claim *a certain level of confidence?*
- needs empirical basis. E.g., if
  - we know that 9 out of 10 systems developed in similar conditions turned out to be faults-free in use
  - or have studied the effectiveness of the assurance methods applied over many applications
- ... we can estimate a *probability* that *the current* system is defect-free, e.g.

$P(\text{faults in product} \mid \text{passing verification}) =$

$$\frac{P(\text{faults})P(\text{passing} \mid \text{faults})}{P(\text{faults})P(\text{passing} \mid \text{faults}) + P(\text{no faults})P(\text{passing} \mid \text{no faults})}$$

repeating for additional assurance steps

## Using “probability of absence of faults”

- a special case of “the requirement that failures are less likely than  $10^{-x}$  is satisfied”
  - for which evidence is given by the fault avoidance and fault detection/removal practices used
  - and with nice properties: 0 probability of *any* failure under *any* kind of usage
- of course even if I were 99% certain of *no* dangerous faults
  - I’d want some assurance that if my system is one of the unlucky 1% it des not fail *too often!*
  - *operation or operational testing* will give this assurance

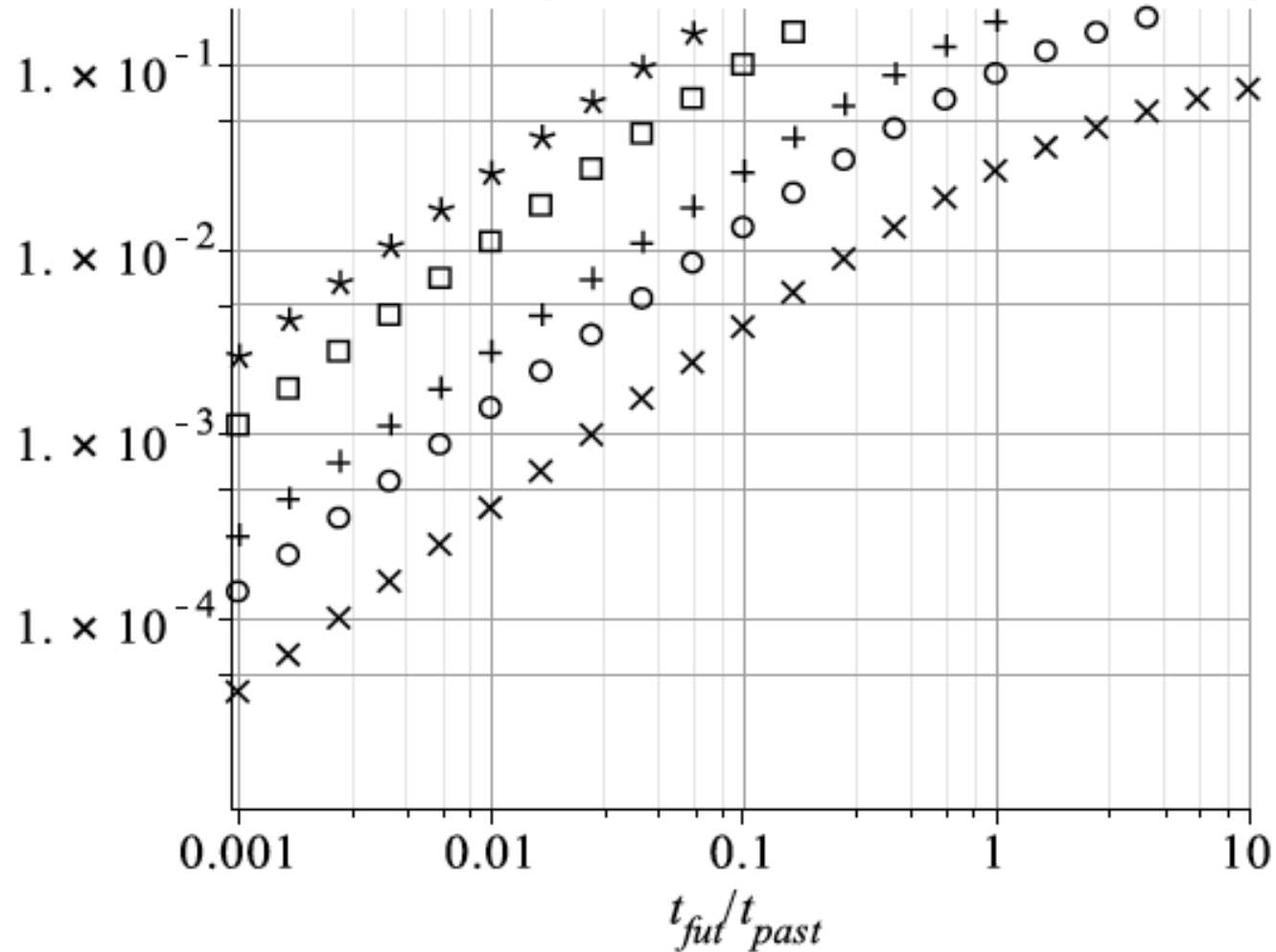
*this reasoning requires* not just fault avoidance/removal practices (standard for critical systems) but *evidence of their effectiveness*

## Using “probability of absence of faults”

- a difficulty is the need for a prior distribution for *pdf* or failure rate if defects are present
- too little evidence now known about effects of development/assurance practices
- we showed there is a *worst case* prior, guaranteeing *pessimistic* errors on probability of future failure
- sufficient in certain scenarios

# Example of application

Worst case value of probability of at least one failure over  $t_{fut}$  future demands, given  $t_{past}$  successes and  $P(pfd = 0) = P_p$



* $P_p = 0.01$	□ $P_p = 0.1$	+ $P_p = 0.5$
○ $P_p = 0.7$	× $P_p = 0.9$	

# Using “probability that faults will be corrected before accidents”



- accept that faults will be present despite *software* assurance and certification
- consider defence-in-depth in *system* design
- implying that many “dangerous” faults of the software
  - will not cause accident at the first failure[s] (if any) that they cause
  - will be found, corrected and never cause accidents
- thus a system
  - is objectively more dangerous at start of use than later
  - but low lifetime risk is achieved

*this reasoning requires* not just detection/mitigation mechanisms (standard in critical systems) but *evidence of their effectiveness*

# So, what more may certification practice need?

- ... to directly give *confidence* in quantitative claim of safe [or secure or reliable or ...] operation
- ... using these [examples of] reasoning approaches would require
  - documentation of effectiveness of assurance practices
    - + of whole processes
    - + of individual practices
    - and of diversity among them
  - documentation of effectiveness of run-time mitigation and *reporting*
    - + coverage of defence-in-depth mechanisms
    - + and of error detection, reporting *practices*

## Notes: *diversity* of fault avoidance/removal

- with increasing criticality of function, we demand more stringent assurance practices and more of them
- with what *cumulative* effects? Only knowable from experience
- e.g., suppose I apply two defect-finding techniques
  - suppose documented effectiveness: one detects existing problems 60% of times, one 50% of times
  - together they might detect problems somewhere *between*
    - + 100% of times - they are complementary
  - + *and* 60% of times – their coverages overlap

(provided that the knowledge of having both has not reduced the care in application)
- note: this learning is important not only for confidence, but for effectiveness and cost-effectiveness of practices

# Detection and reporting of in-operation events

- emphasis on operation-time *detection/monitoring, logging* of errors/failures/ near misses
- conventional wisdom for *achieving* safety
  - at organisation level, related to resilience, safety culture
  - at technical level, covered (more or less strongly) by standards
- *also* crucial for arguing that any faults will not undermine safety requirements
- evidence of effectiveness is essential
  - but more difficult to collect than evidence of failure-free behaviour of the subsystem they protect
  - mechanisms rarely invoked
  - subsystem failures are rarer than operation, their profile less predictable



# Notes: System vs components

(more important distinction than “software vs hardware”)

- the more minute a part of a system we consider  
the greater the role for simply showing it “correct”
- the more we climb towards considering the whole system,  
the more it matters
  - whether the spec was correct (*any system design* faults?)
  - what mitigation measures exist in the rest of the system and how well they address the failure modes of the components and system
  - more difficult to assess
    - + normal use, operational test will rarely test mitigation measures
  - hence importance of *fault injection*  
and of its limitations: it can document coverage factor against
    - + *known* faults in this system
    - + realistically common faults of *this class* of products
    - + but: *unknown* faults in *this* system??

# Conclusions?

- software certification practices as they are bring many likely benefits
  - improved by ongoing technical advances
    - + together with some possible harm
- but not generally the main desired benefit: clearly justified confidence in a (quantitative) level of dependability
- improving their potential requires
  - explicit, mathematical statement of the reasoning through which the claim is argued to be satisfied
    - I have given two examples
      - \* using probability of “perfection” plus operational testing/use
      - \* using effectiveness of defence-in-depth plus fixes
  - collection of quantitative evidence as input to this reasoning
- these requirements are not technically disruptive
  - they support/extend practices already recognised as desirable
- though they may incur political/economic difficulties

Thank you for listening!

Any comments, questions?