Evidence-critical systems: what they are and why we need them

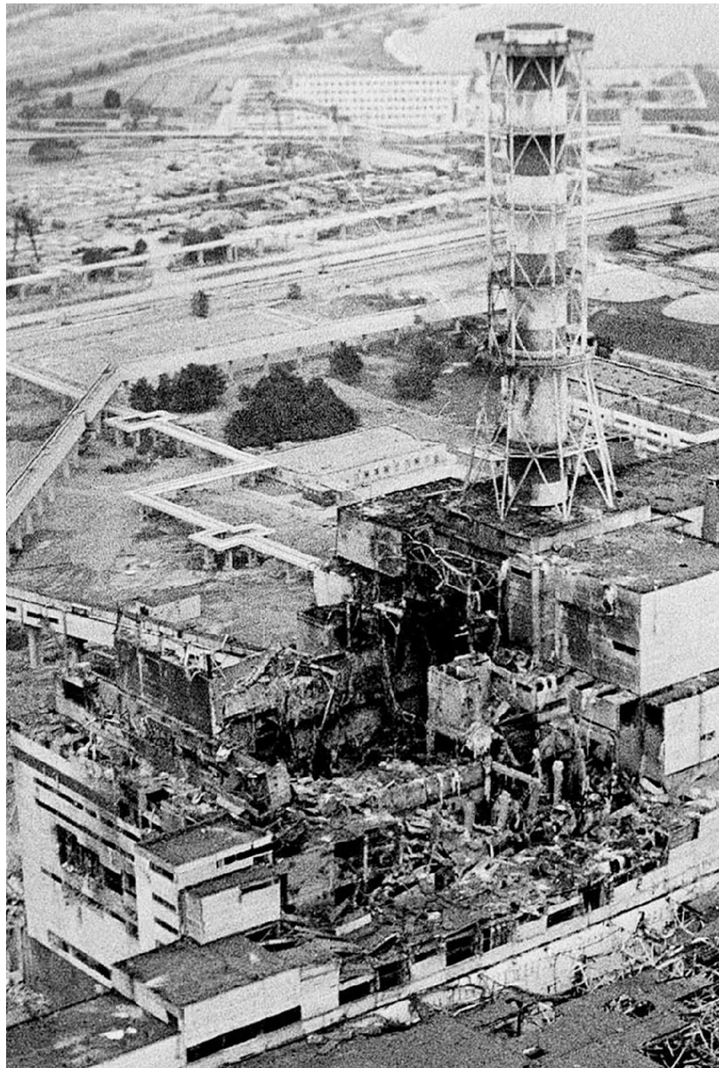
Steven J. Murdoch
University College London

Get these slides here

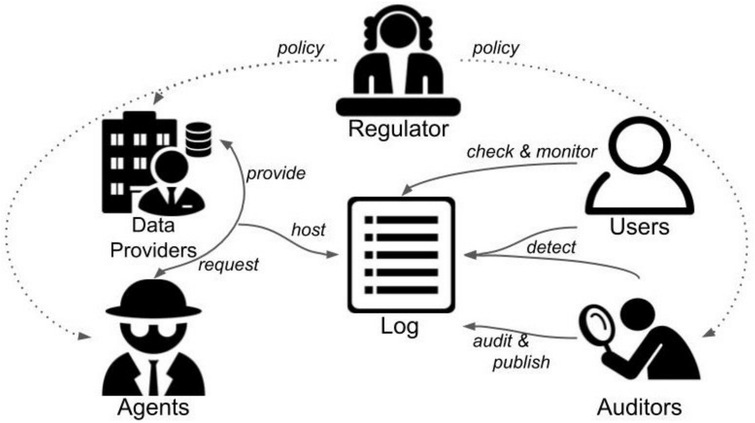
murdoch.is/:/shb20

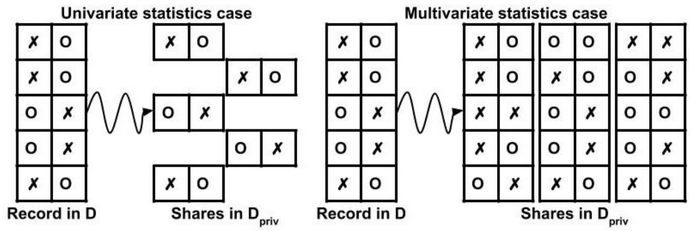
Computers are limited to enforcing policies that can be unambiguously expressed in code

- If you want to a computer to require actions meet certain criteria, the actions and criteria must be precisely described in a programming language, e.g.
  - “Only people who know this password can read this confidential file”
  - “Within 1 second of overheating the nuclear reactor must be shut down”
- Computers are not so good when human interpretation is required to enforce policies
  - “Data must be disclosed if and only if it is necessary and proportionate to do so”



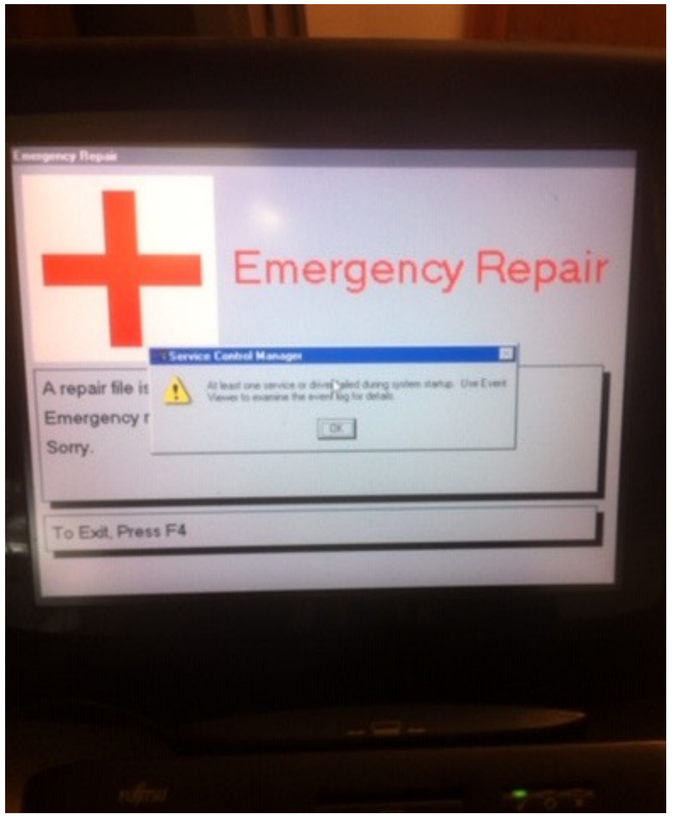
Transparency can help detect violations of the ambiguous policy, but only if victims have power to do so





- One option: let anything happen and trust people to act in a trustworthy way
- We can do better: transparency enhancing technologies enforce that actions are visible, so failures can be identified using audit logs
- One challenge is how to allow the public to audit logs of actions that are cannot be made public
  - VAMS does this by allowing statistics to be verified without having access to underlying data
- Transparency doesn't necessary imply agency

What's the right process to turn verifiable data into fair outcomes for users of the system?



- The legal system is the way we usually turn evidence into justice, but it's imperfect and has proved particularly problematic where computers are used
- Consider the prosecutions of 900+ subpostmasters on the basis of evidence generated by the Horizon accounting system finally shown to be not "remotely robust"
- Part of the problem is that the English legal system presumes that computers are reliable unless shown otherwise
  - Obtaining evidence that a computer is unreliable is expensive and may be infeasible, particularly for users

Bad news: it's hard; good news: it's easier than building safety critical code

- High assurance engineering is expensive even for the simplest applications of computers so could be argued as unrealistic for all legally relevant computer systems
- Actually it's not so bad: safety critical systems must produce correct and timely responses
- Evidence-critical systems need only to never produce an undetectable incorrect response
  - It's OK to fail to produce a response
  - It's OK if an incorrect response can be detected

How can we design and build evidence-critical systems?

- What are the right technologies and design principles to build systems that will produce adequate evidence to fairly resolve disputes?
- What are the right criteria to evaluate the likelihood of a failure being detected
  - To know the likelihood that a failure occurred given some evidence, we need to know how likely is it to see the evidence, assuming a failure has occurred (Bayes' law)
  - i.e. based on the system design, what is the likelihood of an undetected failure?
- How do we create incentives to ensure that systems are built to these criteria

More discussions on this topic – www.benthamsgaze.org