February 15, 2015

Ministry of Defence

Independent Thinking:

Uncertainty and Risk from Single Points of Failure

- Abstract: The design intent is that safety-critical and safety-related systems should be engineered without single points of failure. This is often achieved through the use of multiple systems that perform the same function (eg, a main system and a backup, multiple copies running in parallel, etc). Achieving, and demonstrating, independence between such systems can be a difficult and subtle problem. This presentation uses several simple examples to illustrate potential difficulties before presenting key challenges for the research and practitioner communities.

February 15, 2015

Ministry of Defence

Independent Thinking:

Uncertainty and Risk from Single Points of Failure

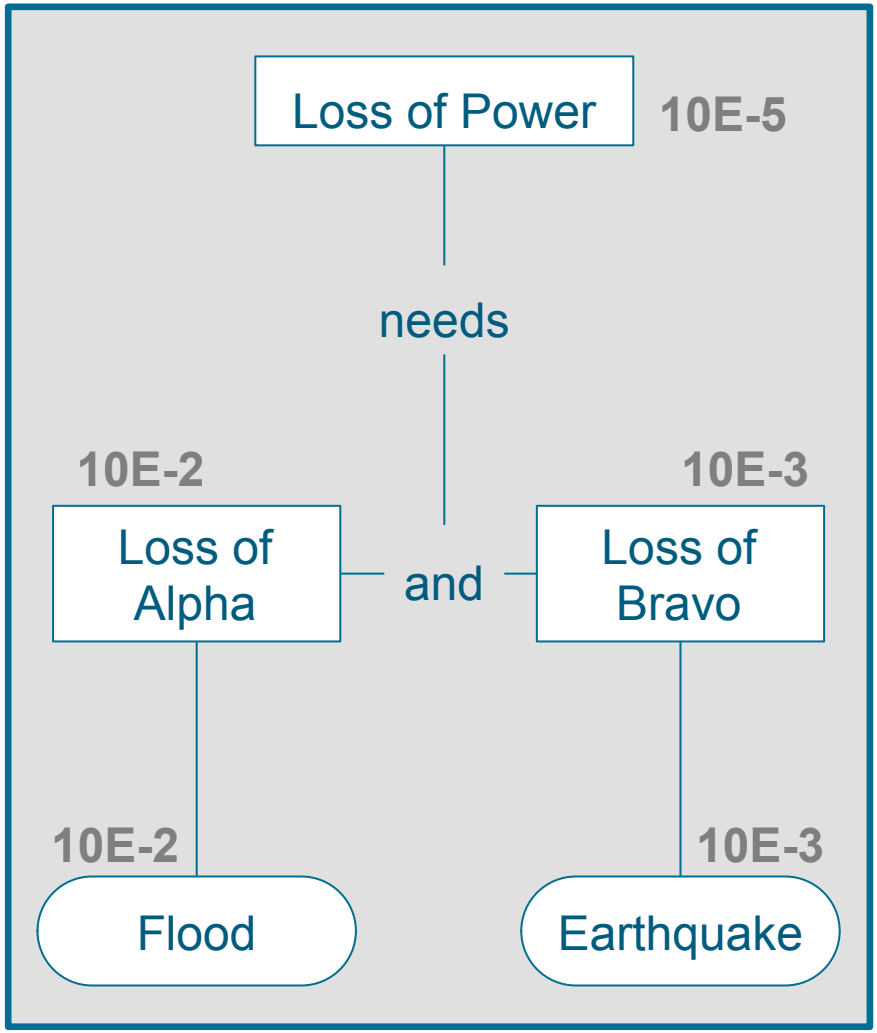
Sue Haines, Rob Ashmore

Dstl Information Management Department

Dstl/DOC85493, January 2015

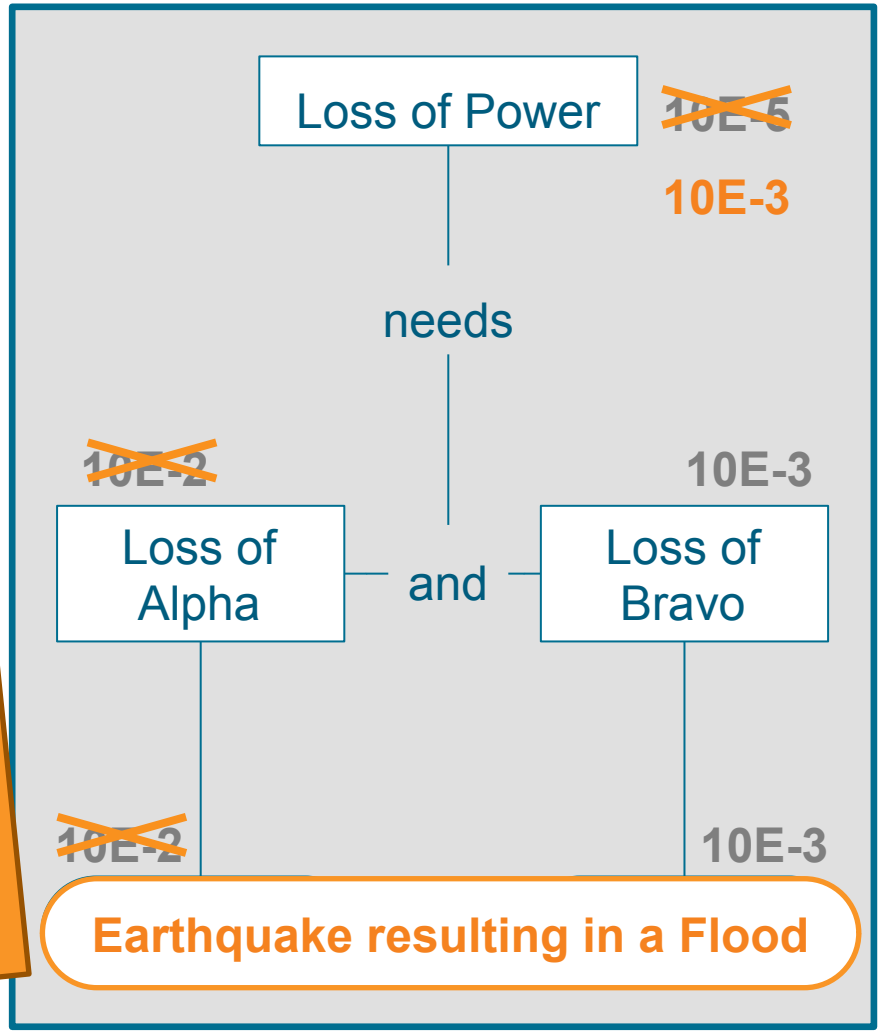
An Example

- Two separate power systems (Alpha and Bravo)
- One has (perfect) protection against earthquakes and no protection against floods
- Reverse applies to the other
- For (semi-)plausible flood and earthquake frequencies then the risk of losing power is 10E-5 per operating year

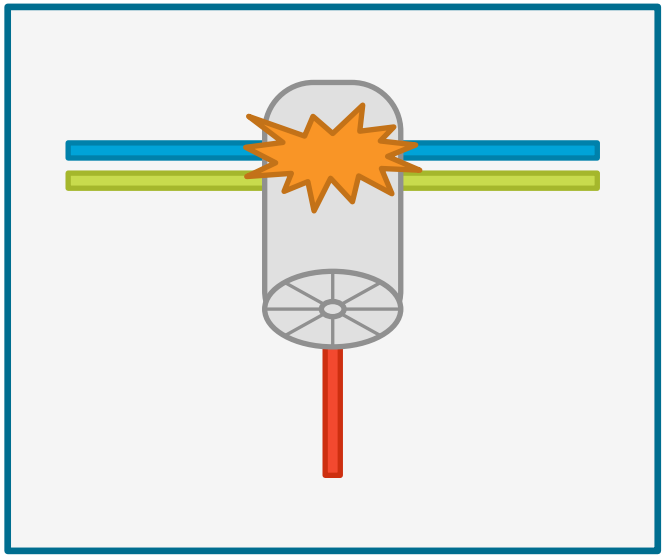


An Example

- But this calculation assumes that floods and earthquakes occur independently
- If our installation is near the coast this may not be true
- Removing the independence assumption has a very large effect on system-level risks



Another Example

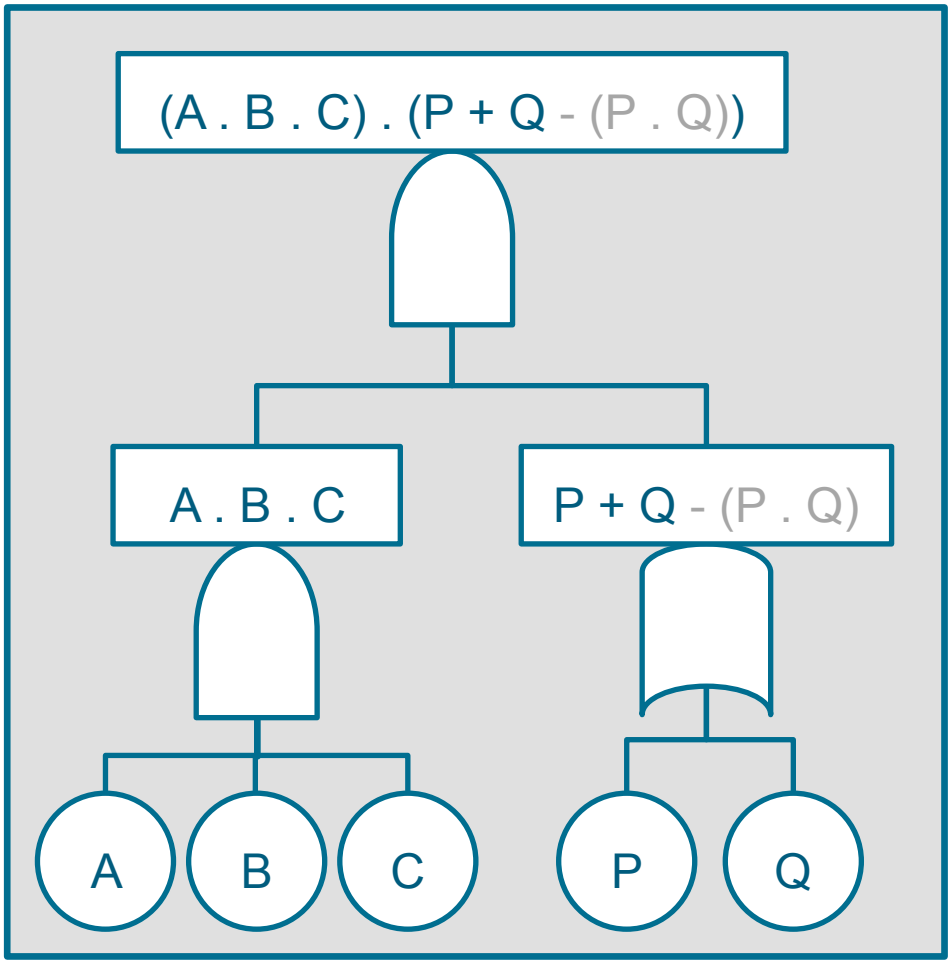


- Suppose an aircraft has three independent hydraulic systems
- But all three route close to an engine (and none are equipped with hydraulic fuses)
- In this case an uncontained engine failure could lead to catastrophic loss of all three hydraulic systems
- Again, the assumption of independence is violated

Common Cause Analysis (CCA)

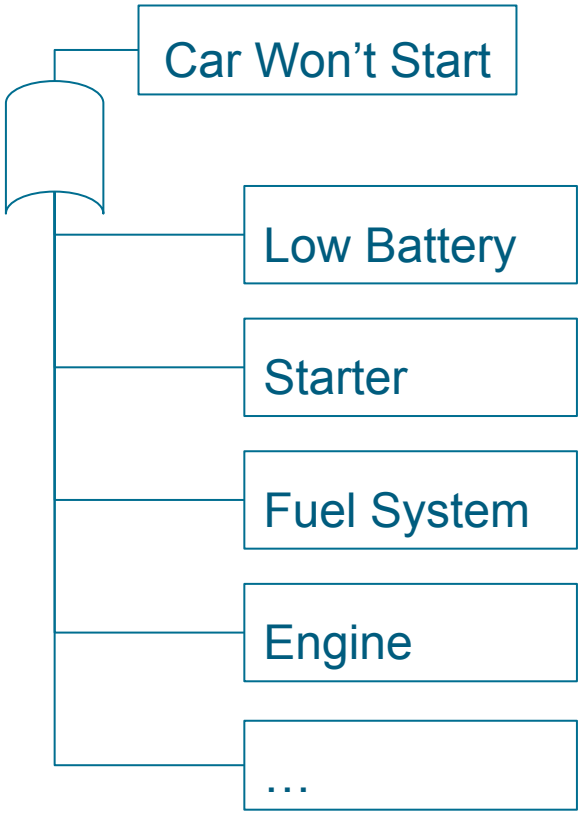
- Common Cause Analysis seeks to identify cases where assumptions about independence may be violated
- ***Particular Risk Analysis*** considers items outside the system that affect multiple zones at the same time (eg, bird strike, tyre burst)
- ***Zonal Safety Analysis*** works through on a zone-by-zone basis (eg, installation, potential interference between systems)
- ***Common Mode Analysis*** verifies that items combined with AND gates in a Fault Tree are independent

Fault Trees



- Built from undeveloped base events
- Combined via logical operators (primarily AND, OR gates)
- Probabilities (almost always expressed without uncertainty) propagate upwards
- Support quantitative and qualitative analysis

Fault Trees



- What's missing from this example?
- How would you allocate probabilities to the nodes?
- How big will the final tree be?

- Despite their limitations, fault trees are a practical and widely-used way of understanding how individual components affect overall system safety

Models

Some models are **“deep”**

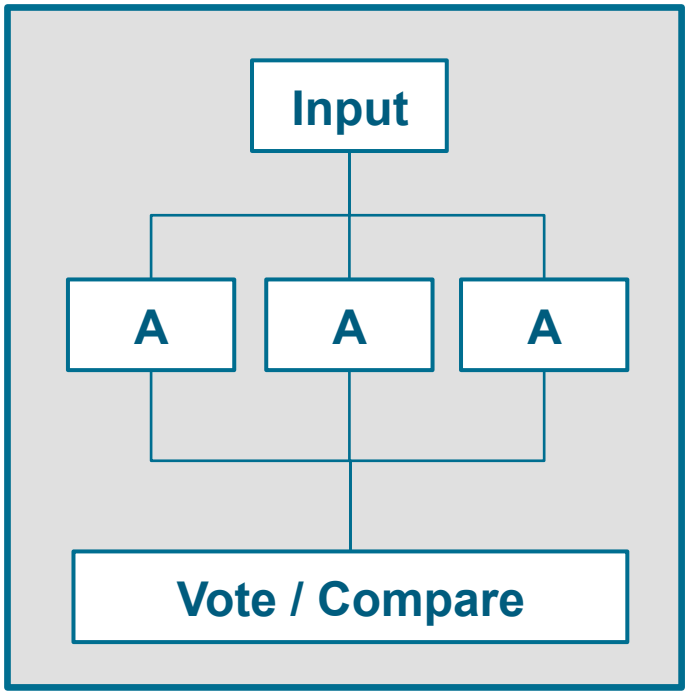
- They have complicated internal structures
- They take a long time to run
- Sensitivity / uncertainty analyses are often based on emulators
- Building an emulator often involves pre-selecting “key” inputs and outputs

Some models are **“shallow”**

- Internal structures are built from a small number of simple components
- They can be very quick to run
- But they are large scale, often featuring many thousands of equivalent-level inputs

Common Software

- The nature of software means it is particularly susceptible to common mode failure
- Providing identical software with identical inputs (almost always) gives identical results
- This can undermine assumptions associated with the provision of independent (computer) hardware



Software - Adding Independence

Software

- Can try to introduce independence by using different (independent) development teams
- But all will be working from the same requirements
- And this approach significantly increases costs

Data

- Can add diversity into the input data
- Occasionally, this is naturally available (eg, multiple sensors)
- Theoretically, artificial diversity can also be added
- But it is difficult to justify any given level of diversity

Hardware

- Can compile the same source code for different targets
- This offers some protection against errors in compiler (and processor) implementation
- But these are not the main sources of software errors

Software Independence

- There is ***no way of achieving (total) independence*** between common software
  - As a minimum, they share common requirements
- There is ***no (easy) way of accurately measuring software independence***
  - Since this is conceptually equivalent to the problem of fully testing the software
- The level of independence could be "***almost none***"
  - Close to that which arises from Single Event Upsets and hardware faults
- Or it could be "***quite large***"
  - Independent development teams, using different languages, targeting different hardware
- Or "***anywhere in between***"

Challenges

1. **Quantifying software reliability**
   - Mostly a philosophical discussion - not for this conference!
2. **Effect of uncertainty in fault tree structure**
   - Uncertainty in overall structure
   - Uncertainty in independence assumptions - especially with regards to “undeveloped base events”
3. **Conducting meaningful sensitivity / uncertainty analysis on large-scale (1000s of components) “shallow” models**
   - Challenges computation, analysis and visualisation

Questions / Comments?

Sue Haines
slhaines@dstl.gov.uk
96770 4069
030 6770 4069

Rob Ashmore
rdashmore@dstl.gov.uk
96770 3185
030 6770 3185

Selected Bibliography

- Fischoff, Fault Trees: Sensitivity of Estimated Failure Probabilities to Problem Representation, Defense Advanced Research Projects Agency, 1977.
- Hecht, H., Rare Conditions - An Important Cause of Failures, Proceedings of the Eighth Annual Conference on Computer Assurance, 1993 (COMPASS'93).
- Knight, J.C. and Leveson, N.G., An Experimental Evaluation of the Assumption of Independence in Multiversion Programming, IEEE Trans. Soft. Eng. 1986 (SE-12) 96-109.
- NTSB Aircraft Accident Report, United Airlines Flight 232 McDonnell Douglas DC-10-10, Sioux Gateway Airport, Sioux City, Iowa, July 19, 1998, PB90-910406, NTSB/AAR-90/06.
- Gramacy, R. B., tgp: An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models, Journal of Statistical Software, 19(9) 2007.
- SAE International, ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.
- Yeh, Y. C., Design considerations in Boeing 777 fly-by-wire computers, Third IEEE High-Assurance Systems Engineering Symposium, 1988.

February 15, 2015

Ministry of Defence