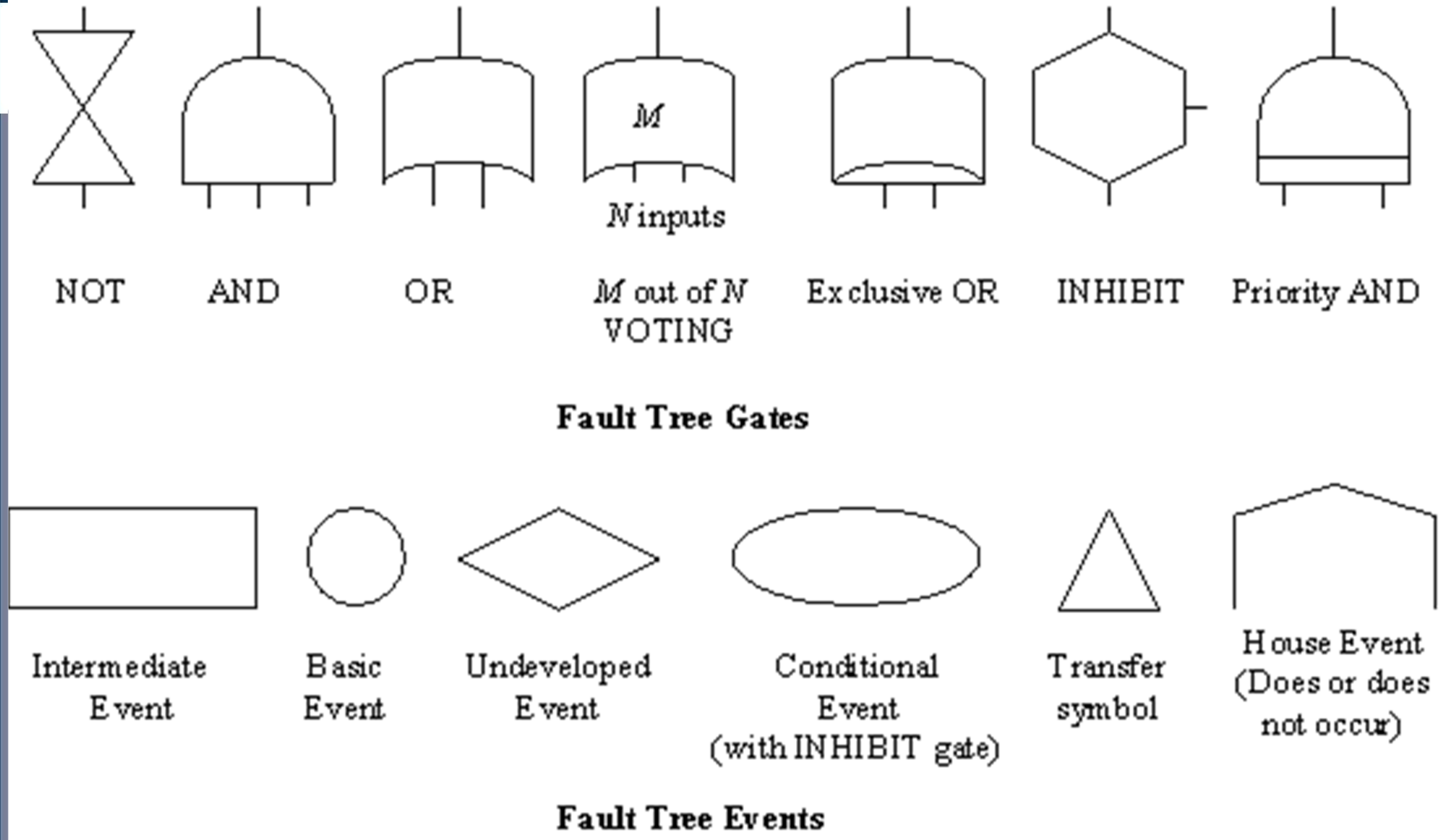
Fault Trees and Software PRA

Prof. Chris Johnson,
School of Computing Science, University of Glasgow.
johnson@dcs.gla.ac.uk
http://www.dcs.gla.ac.uk/~johnson

Introduction

- Fault Tree Analysis: Recap.
- Software Fault Trees.
- Software Probabilistic Risk Assessment.

Fault Trees: Recap



University of Glasgow

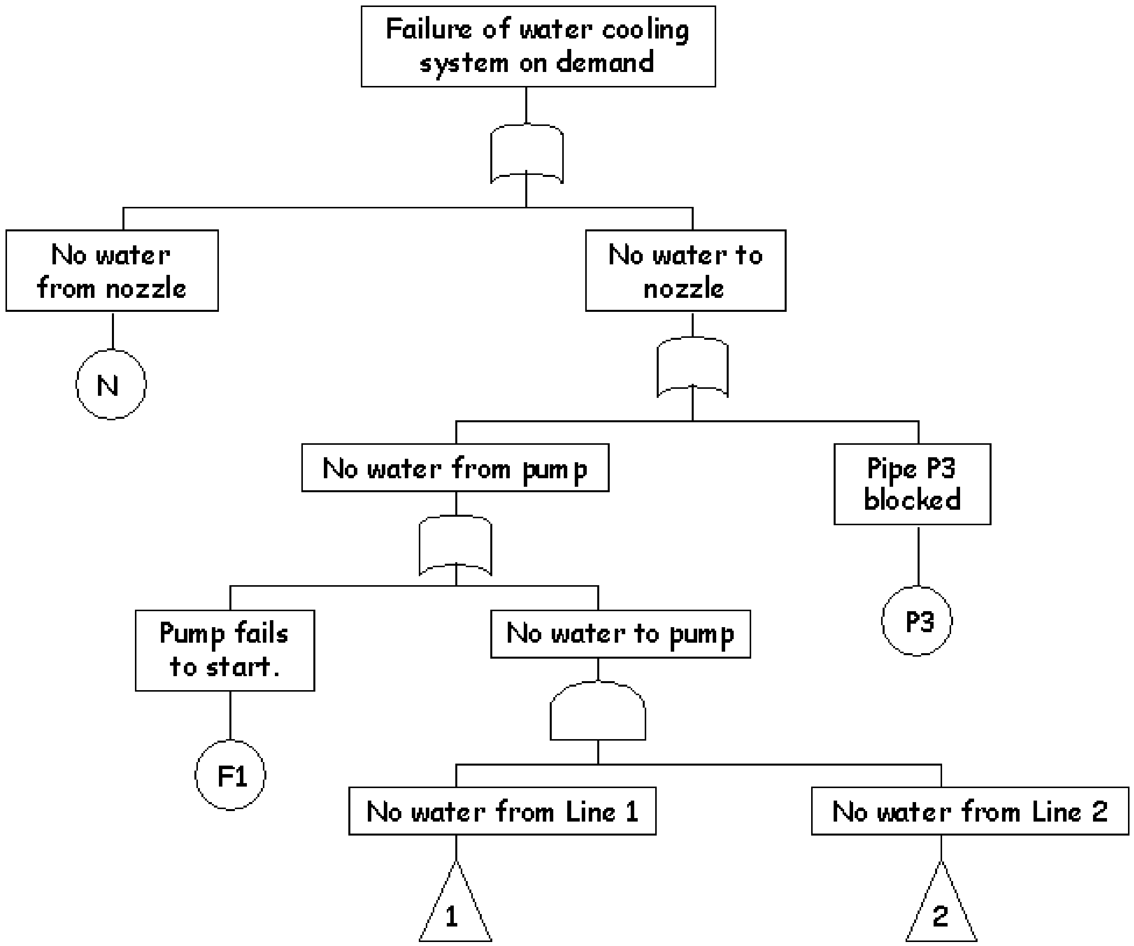
Fault Tree Analysis

- Each tree considers 1 failure:
  - Carefully choose top event;
  - Carefully choose system boundaries.

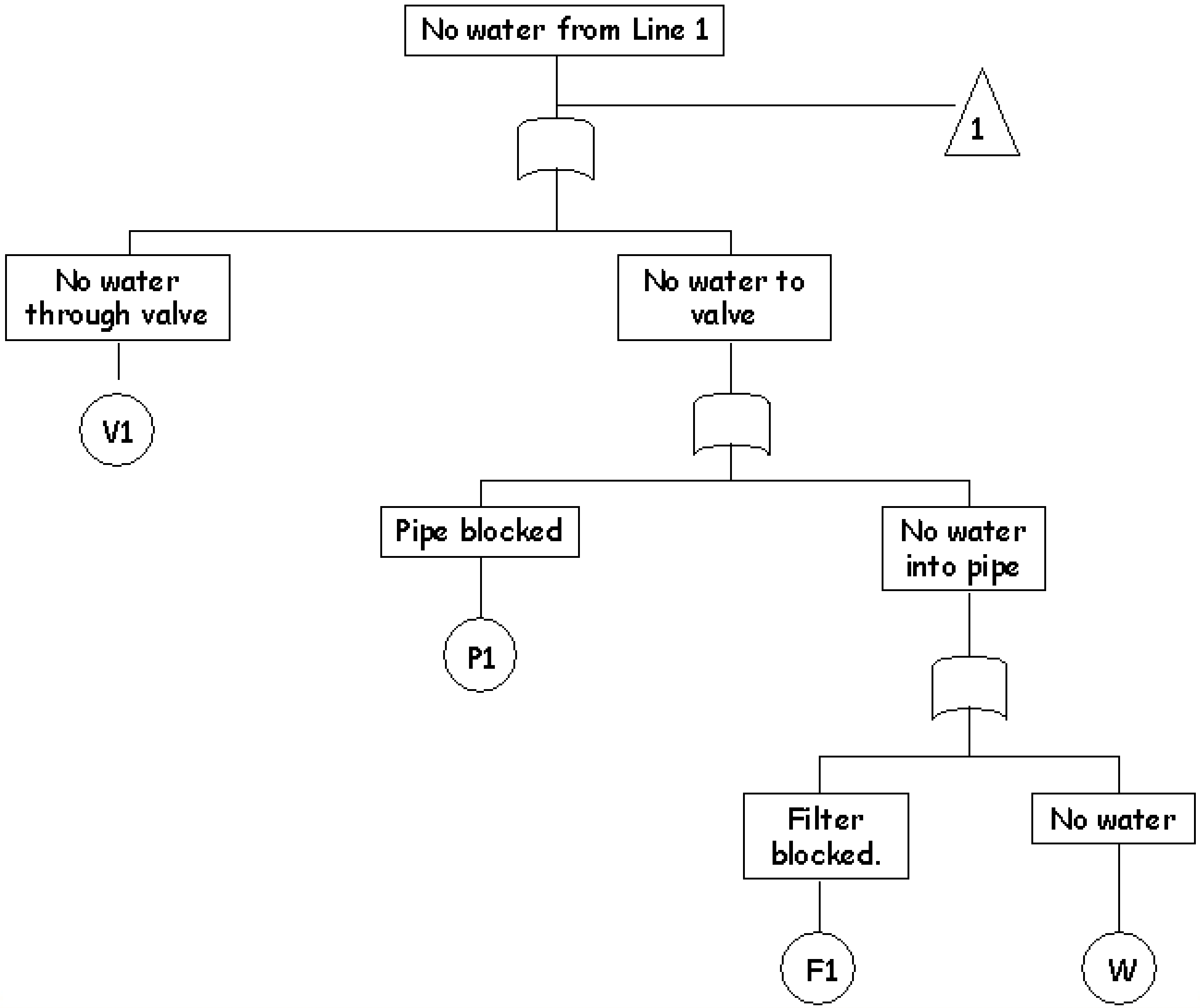
- Assign probabilities to basic events:
  - Stop if you have the data;
  - Circles denote basic events.

- Simple but tool support is critical.

Fault Tree Analysis: Hardware



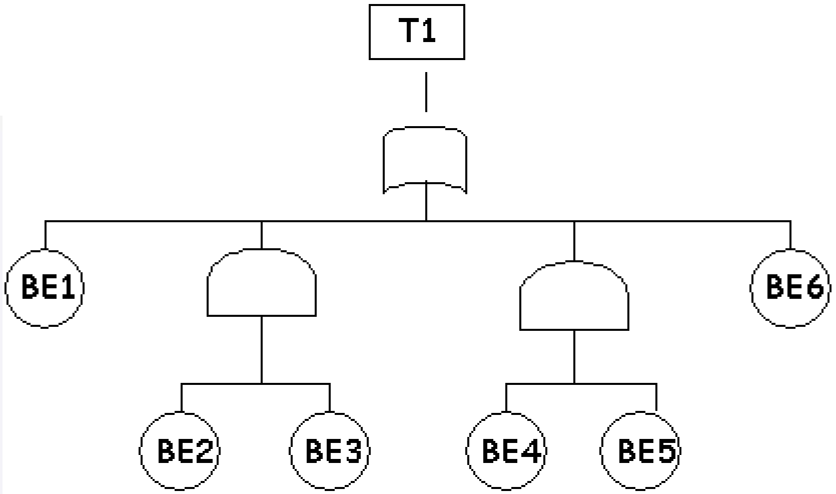
Fault Tree Analysis: Hardware



Fault Tree Analysis - Cut Sets

- Each failure has several modes:
  - 'different routes to top event''.
- Cut set:
  - basic events that lead to top event.
- Minimal cut set:
  - removing a basic event avoids failure.
- Path set:
  - basic events that avoid top event;
  - list of components that ensure safety.

Fault Tree Analysis - Cut Sets



T1 = BE1 + BE2.BE3 + BE4.BE5 + BE6

- Top_Event = K1 + K2 + ... K_n
  - K_i minimal cut sets, + is logical OR.
- K_i = X_1 . X_2 . X_n
  - MCS are conjuncts of basic events.

Fault Tree Analysis - Cut Sets

- Top-down approach:
  - replace event by expression below;
  - simply if possible (C.C = C).
- Can use Karnaugh map techniques;
  - cf logic circuit design;
  - recruit tool support in practice.
- Notice there is no negation.
- Notice there is no XOR.

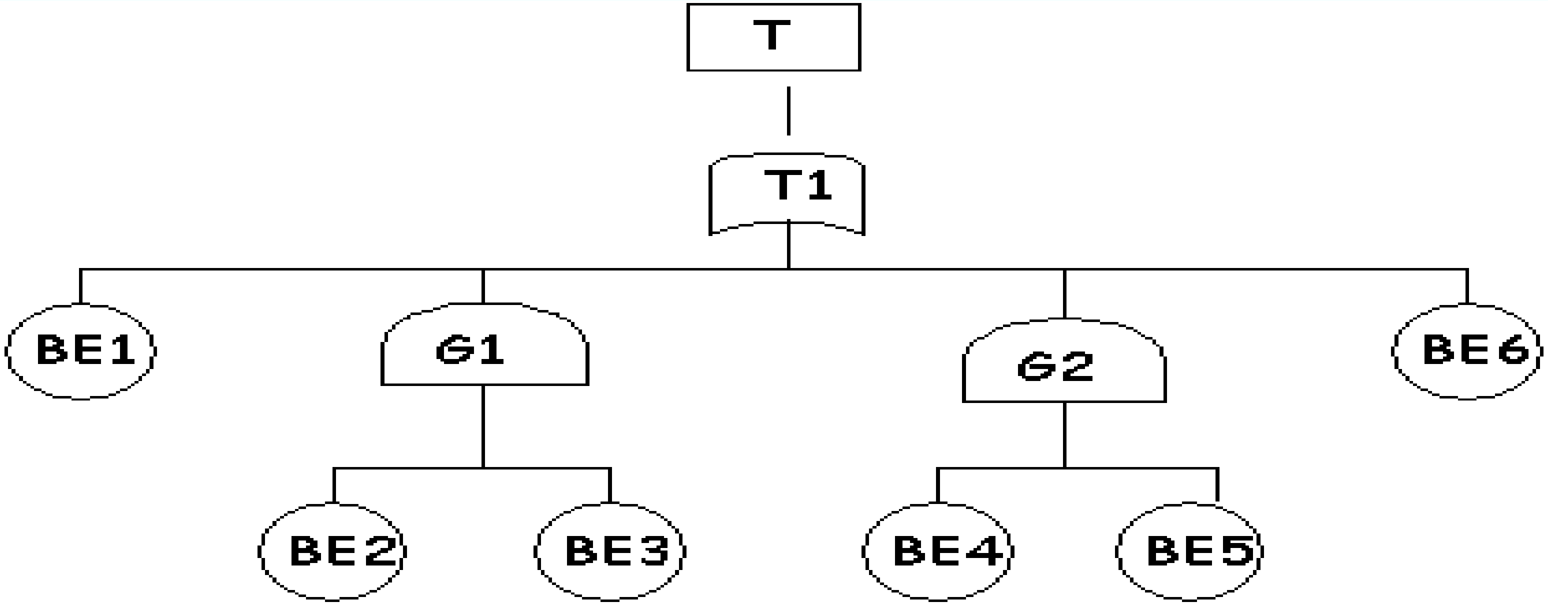
MOCUS Cut Set Algorithm

1. Assign unique label to each gate.
2. Label each basic event.
3. Create a two dimensional array A.
4. Initialise A(1,1) to top event.
5. Scan array to find an OR/AND gate:

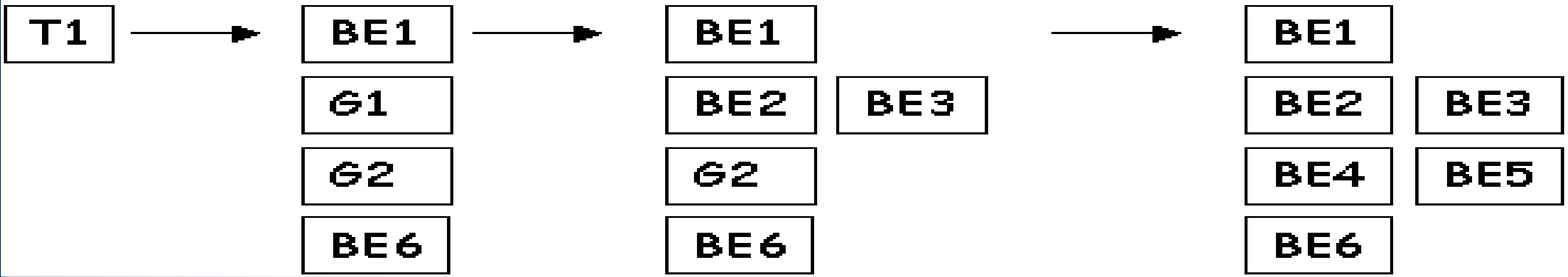
    If current position in A is OR gate...

    - replace current position with a column;
    - put gate's input events in new row of that column.
    - replace current position with a row;
    - put gate's input events in new column of that row.
6. Repeat 5 until no gates remain in array.
7. Remove any non-minimal cut sets.

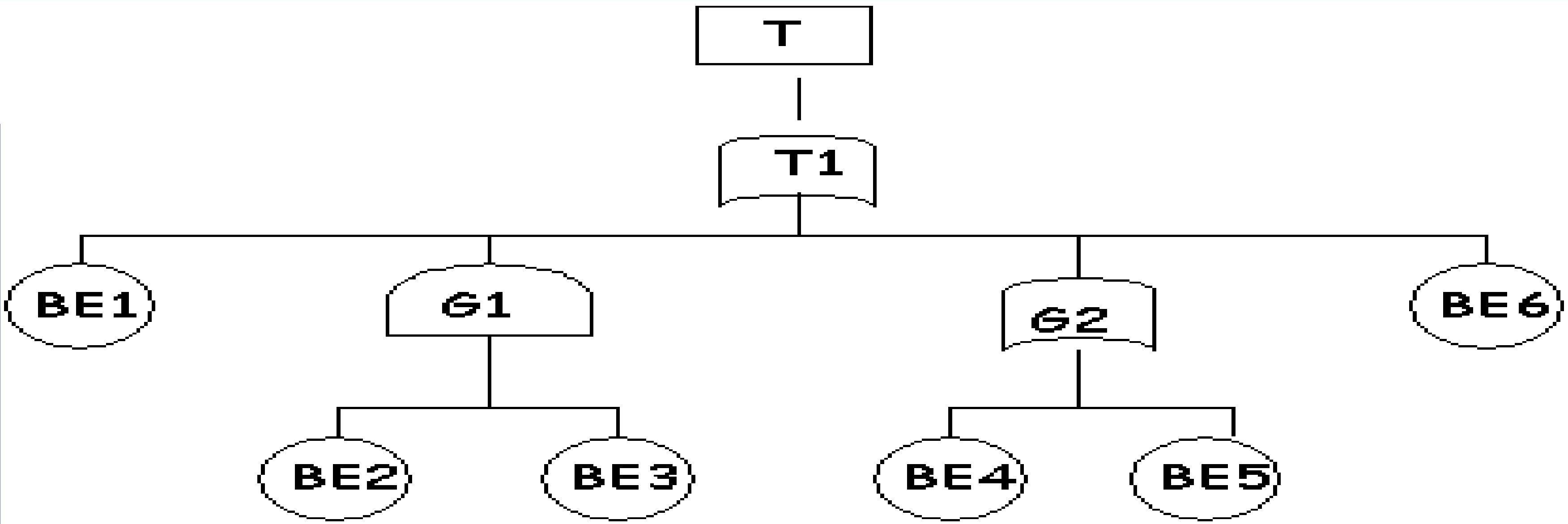
Fault Trees: MOCUS Cut Set Algorithm



T1 = BE1 + BE2.BE3 + BE4.BE5 + BE6



Fault Trees: Probabilistic Analysis



For simplicity assume probability of all basic events is 0.1

P(G1) = P(BE2).P(BE3) = 0.1 * 0.1 = 0.01.

P(G2) = P(BE4).P(BE5) = 0.1 + 0.1 = 0.2.

P(T1) = 0.01 + 0.2 + P(BE1) + P(BE2)
= 0.01 + 0.2 + 0.1 + 0.1
= 0.41

Fault Trees: Probabilistic Analysis

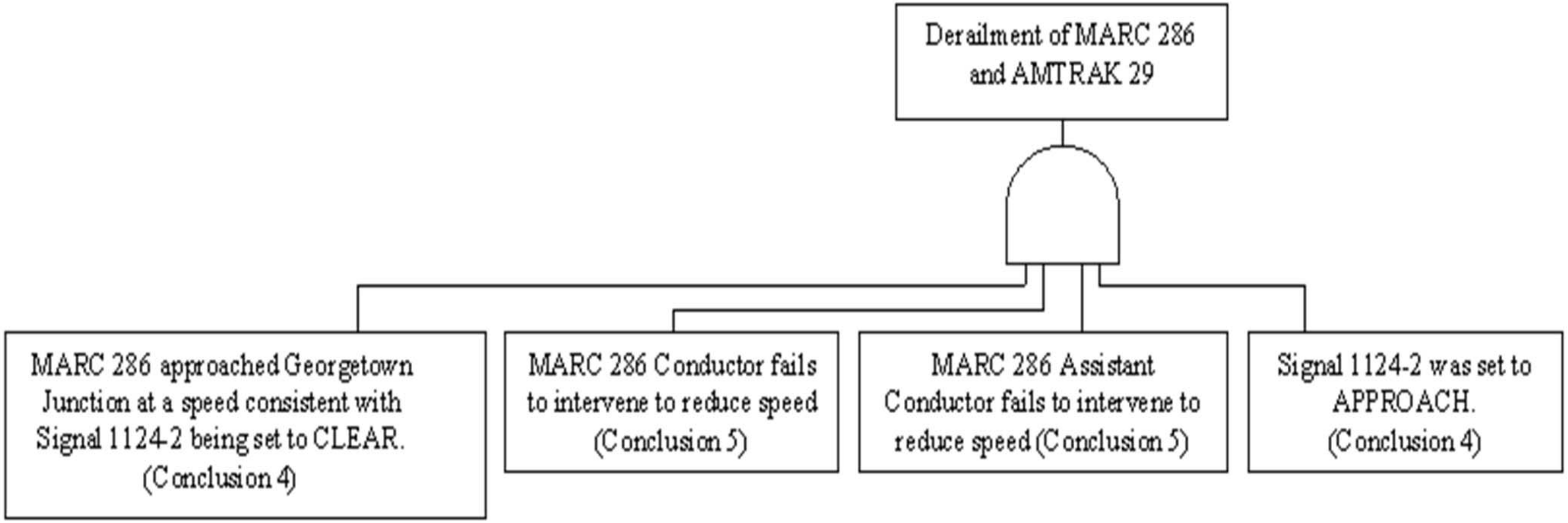
- Beware: independence assumption.

"If the same event occurs multiple times/places in a tree, any quantitative calculation must correctly reduce the boolean equation to account for these multiple occurrences. Independence merely means that the event is not caused due to the failure of another event or component, which then moves into the realm of conditional probabilities."

Clif Ericson, ISSS.

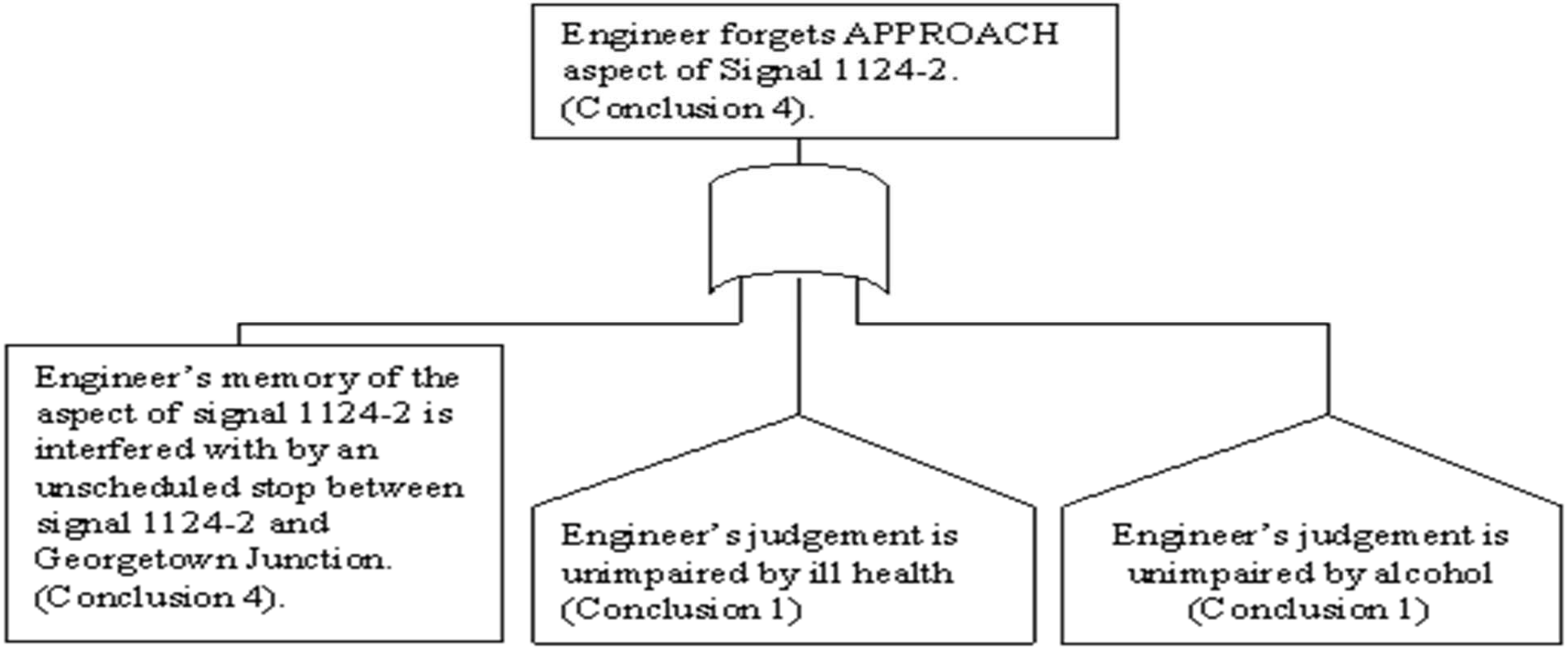
- Inclusion-exclusion expansion (Andrews & Moss).

Fault Tree Analysis: Applied to Systems



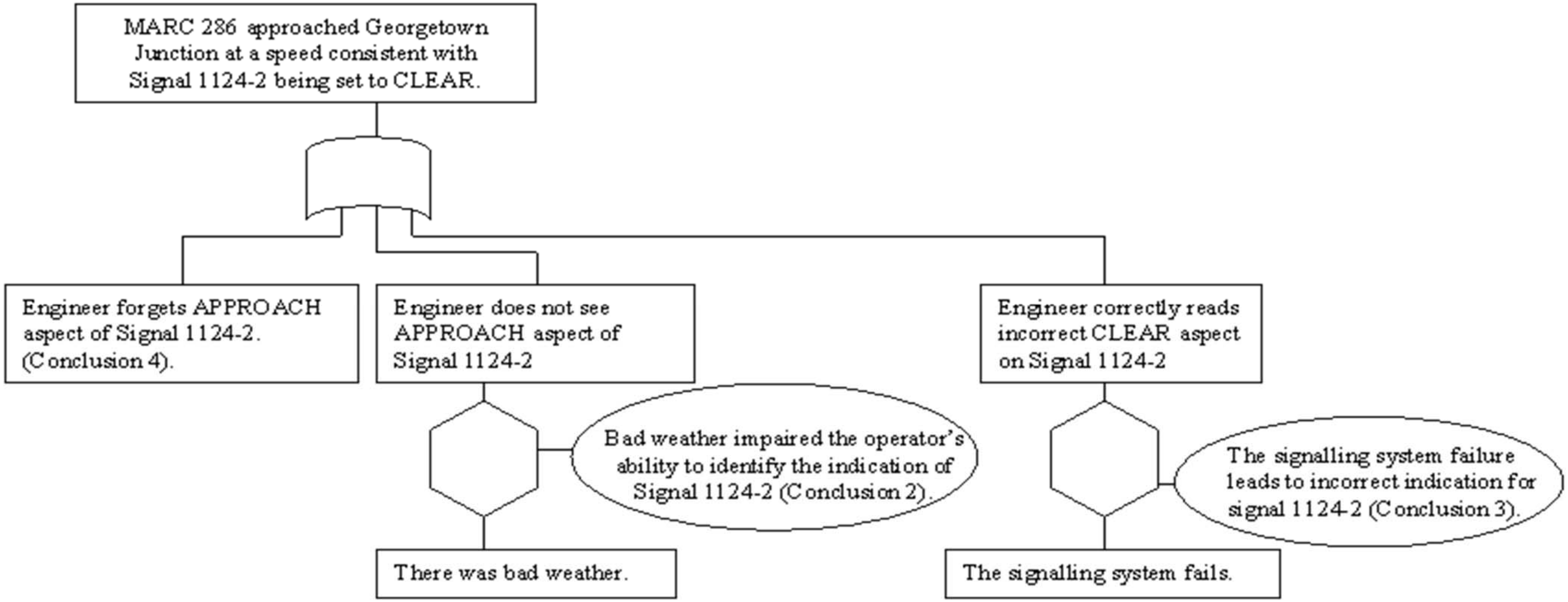
- Usually applied to hardware...
- Can be used for software (later).

Fault Tree Analysis: Applied to Systems



- House events; "switch"' true or false.
- OR gates - multiple fault paths.

Fault Tree Analysis: Applied to Systems



- Probabilistic inhibit gates.
- Used with Monte Carlo techniques
  - True if random number < probability.

Software Fault Trees

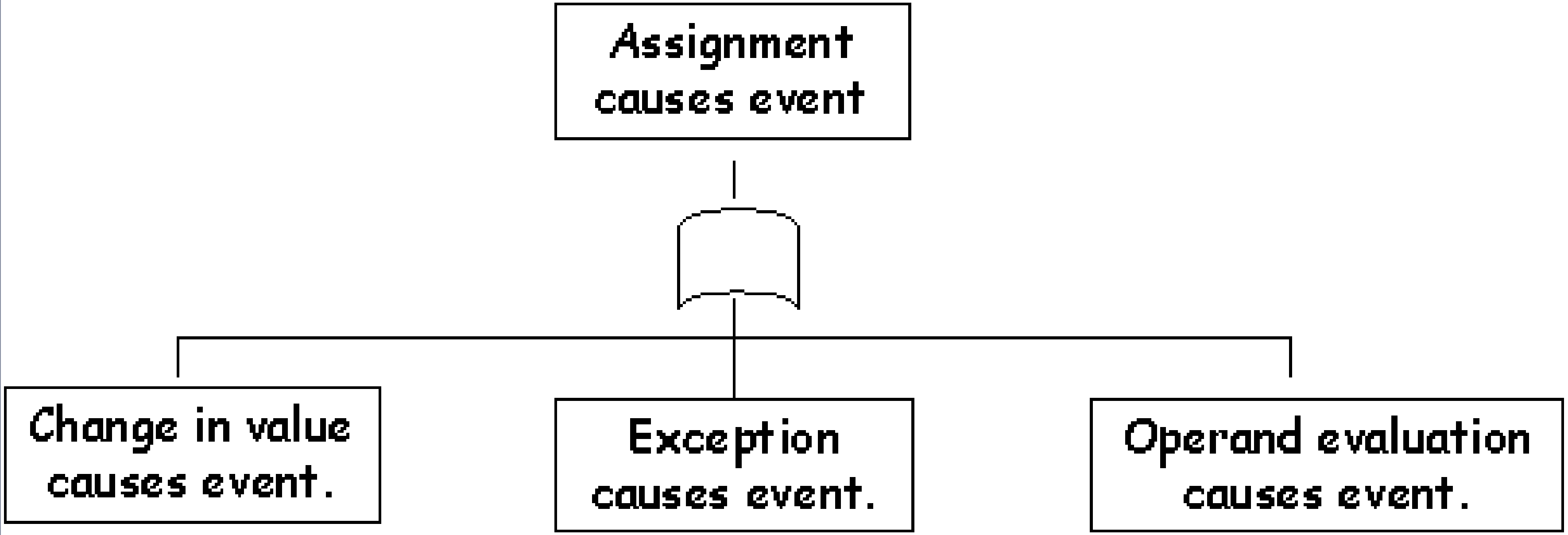
- As you'd expect.
- Starts with top-level failure
  - Trace events leading to failure.
- But:
  - Don’t use probabilistic assessments;
- If you find software fault path REMOVE IT!

Software Fault Trees

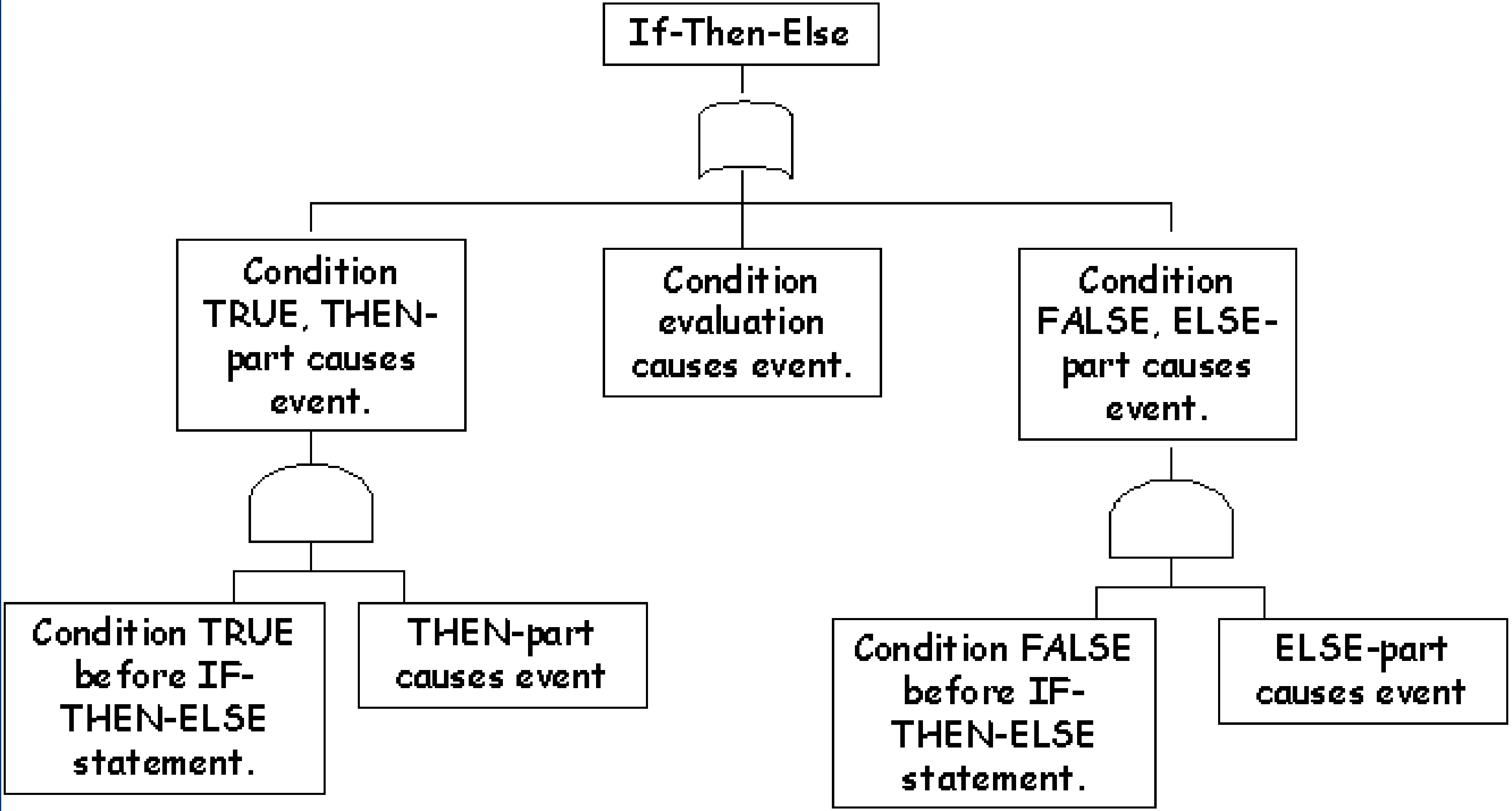
Leveson, N.G., Cha, S.S., Shimeall, T.J. “Safety Verification of Ada Programs using Software Fault Trees," IEEE Software, July 1991.

- Backwards reasoning.
- Weakest pre-condition approach.
- Similar to theorem proving.
- Uses language dependent templates.

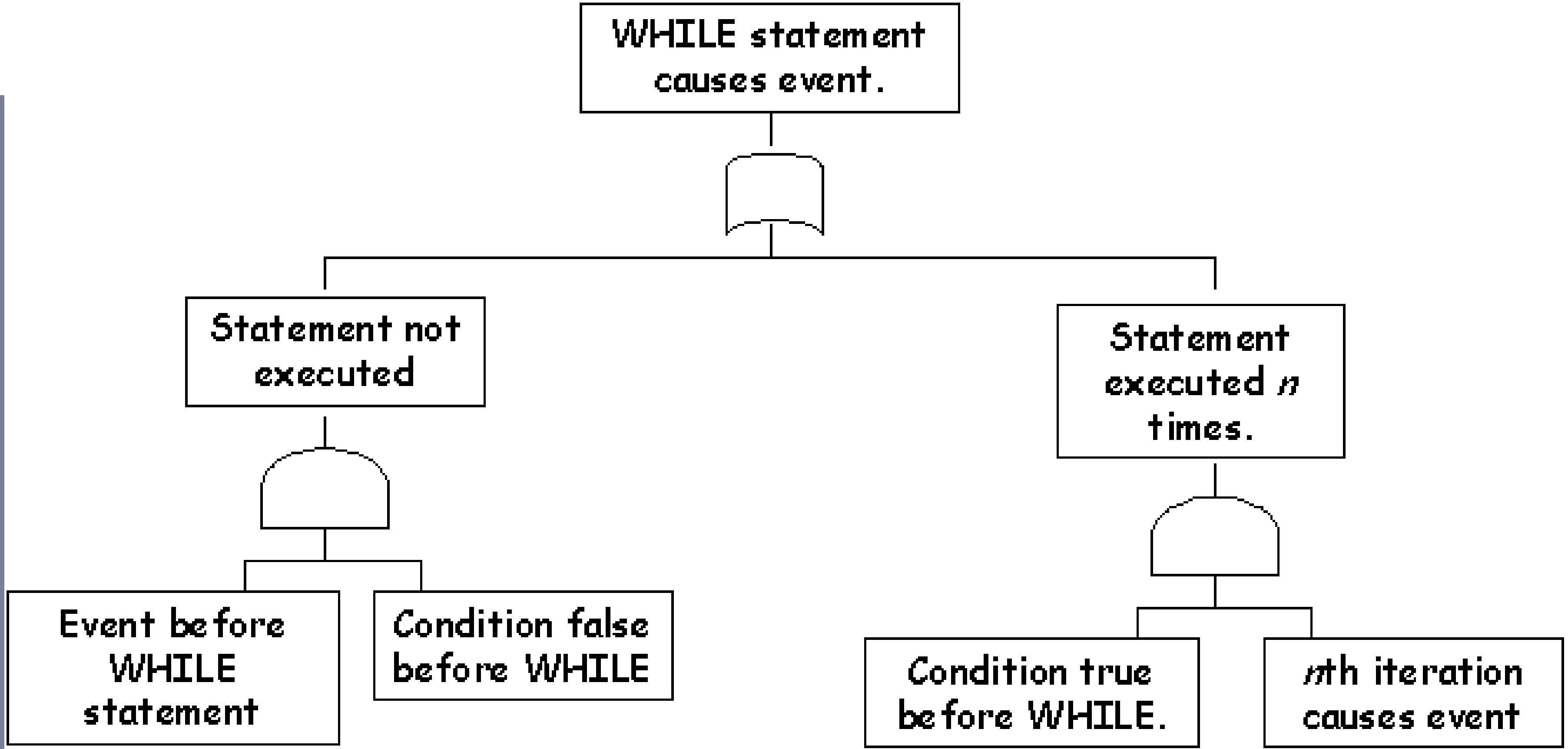
Software Fault Trees



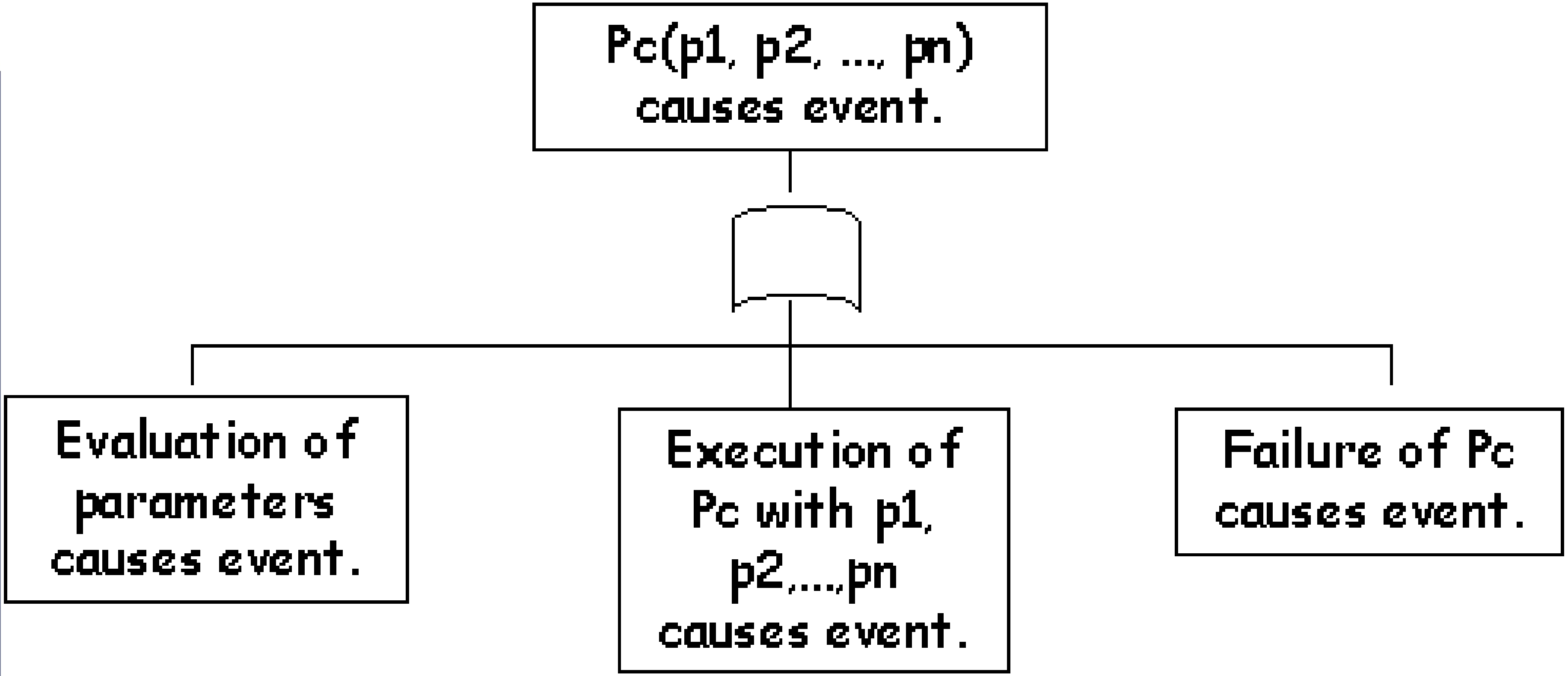
Software Fault Trees



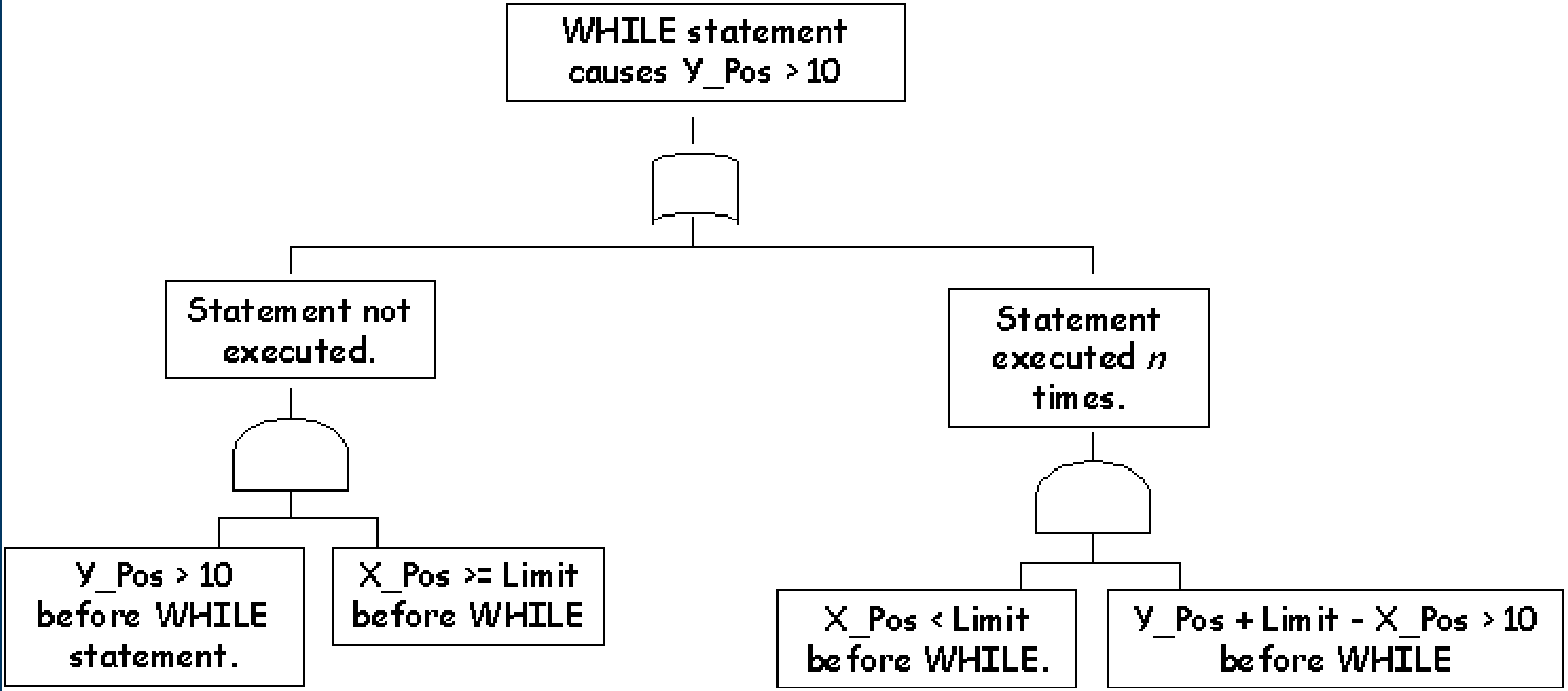
Software Fault Trees



Software Fault Trees



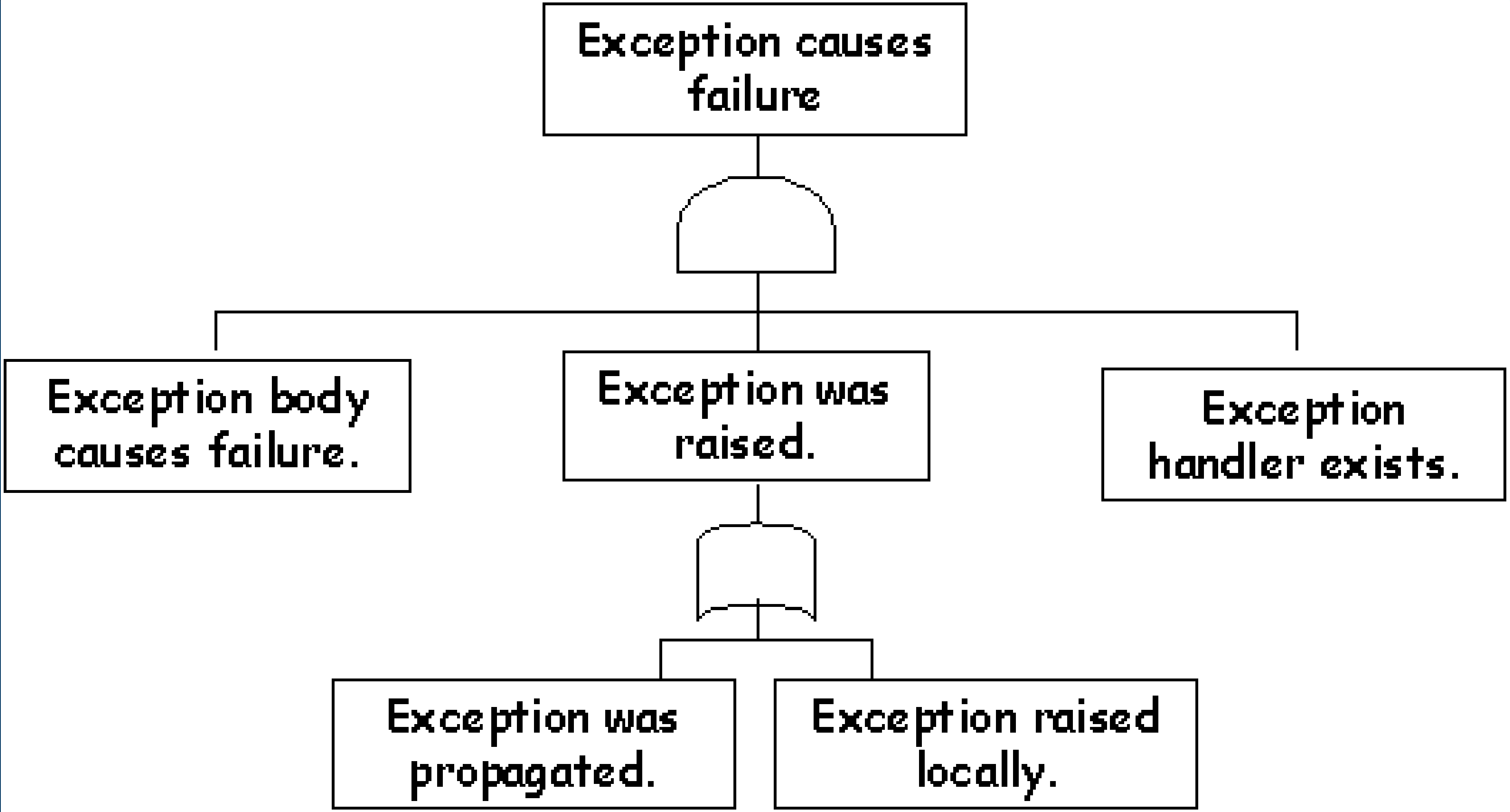
Example Software Fault Tree



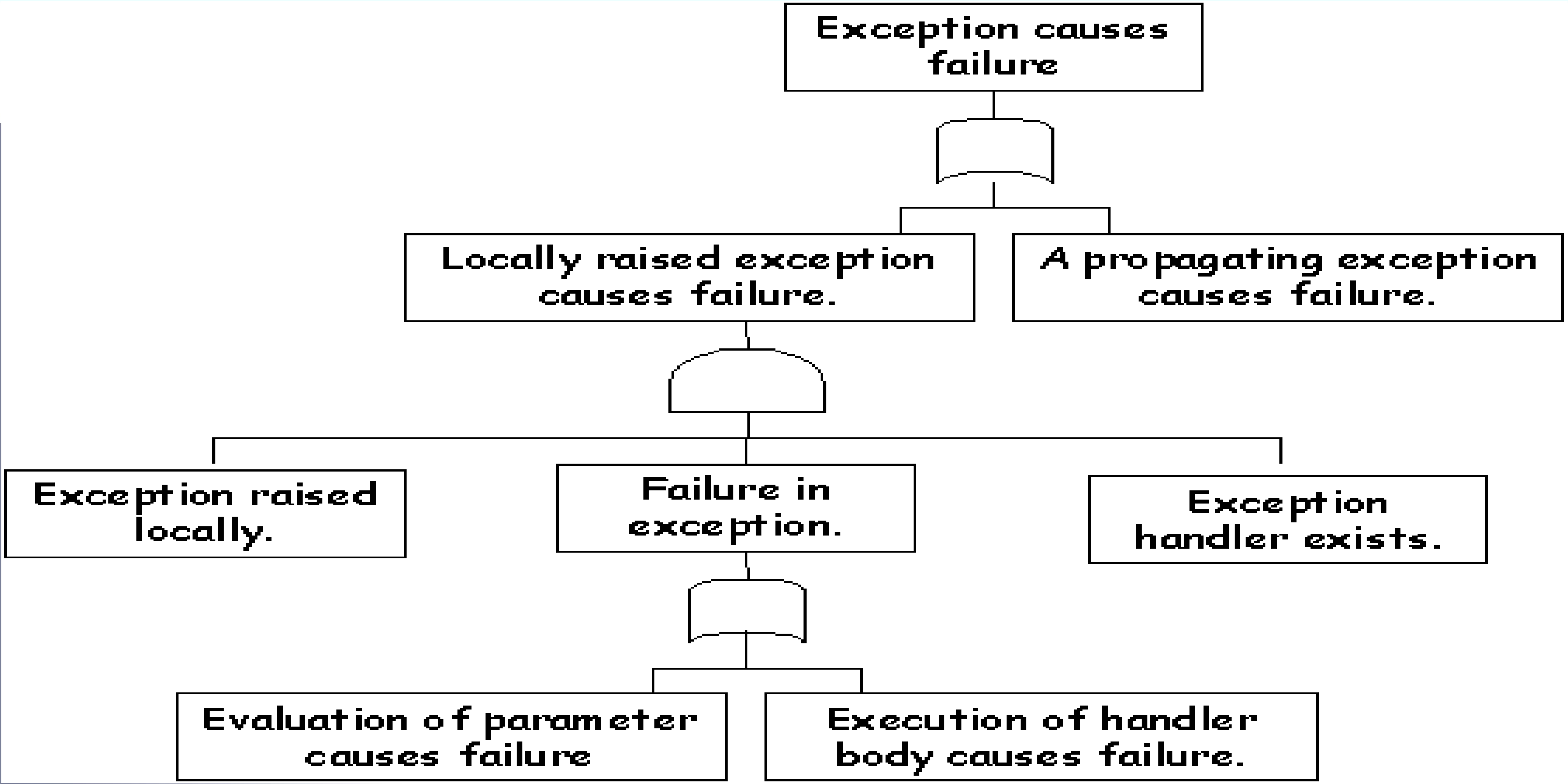

```
while X_Pos < Limit do
begin
        X_Pos := X_Pos + 1;
        Y_Pos := Y_Pos + 1;
end;
```

```
-- Let's assume dangerous condition
-- Y_Pos > 10
```

Exception template for Ada83



Exception template for Ada95



- See: S.-Y. Min, Y-K. Jang, A-D Cha, Y-R Kwon and D.-H. Bae, Safety Verification of Ada95 Programs Using Software Fault Trees. In M. Felici, K. Kanoun and A. Pasquini (eds.) Computer Safety, Reliability and Security, Springer Verlag, LNCS 1698, 2002.

PRA for Software

- John Musa's work at Bell Labs.
- Failure rate of software before tests.
- Faults per unit of time λ_0):
  - function of faults over infinite time.
- Based on execution time:
  - not calendar time as in hardware;
  - so no overall system predictions.

Musa's PRA for Software

- $\lambda\_0 = K \times P \times W\_0$
- K: Constant that accounts for the dynamic structure of the program and the varying Machines, k = 4.2E-7.
- P: Estimate of the number of executions per time unit, p = r/SLOC/ER
- r : Average instruction execution rate, determined from the manufacturer or Benchmarking, Constant
- SLOC: Source lines of code (not including reused code).

Musa's PRA for Software

- $\lambda_0 = K \times P \times W_0$
- ER: Expansion ratio constant per programming language: Assembler, 1.0; Macro Assembler, 1.5; C, 2.5; COBAL, FORTRAN, 3; Ada, 4.5
- W_0: Estimate of the initial number of faults in the program. Can be calculated using: $w0 = N \times B$, or a default of 6 faults/1000 SLOC can be assumed
- N: Total number of inherent faults. Estimated based upon judgment or past experience.
- B: Fault to failure conversion rate; proportion of faults that become failures. Proportion of faults not corrected before the product is delivered. Assume B = .95; i.e., 95% of the faults undetected at delivery become failures after delivery

PRA for Software

- Considerable debate about this:
  - No account for experience of coders?
  - No account for number of teams?
  - No account for complexity of requirements?
  - What about configuration management?
- Many variants on the theme.
- Metrics are crude...
- In meantime, be sceptical.

Conclusions

- Fault Trees:
  - cut sets, cut paths;
  - quantitative analysis.
- Software Fault Trees:
  - language dependent templates;
  - if you see faults, remove them!
- Software PRA: the Musa formula…

Any Questions…