Why do we need something different?

- Fast pace of technological change
- Reduced ability to learn from experience
- Changing nature of accidents
- New types of hazards
- Increasing complexity and coupling
- Decreasing tolerance for single accidents
- Difficulty in selecting priorities and making tradeoffs
- More complex relationships between humans and automation
- Changing regulatory and public views of safety

Assumptions Underlying What We Do

1. Safety is increased by increasing system or component reliability. If components or systems do not fail, then accidents will not occur.

   High reliability is neither necessary nor sufficient for safety.

2. Accidents are caused by chains of directly related events. We can understand accidents and assess risk by looking at the chain of events leading to the loss.

   Accidents are complex processes involving the entire socio-technical system. Traditional event-chain models cannot describe this process adequately.

Assumptions Underlying What We Do

3. Probabilistic risk assessment based on event chains is the best way to assess and communicate safety and risk information.

   Risk and safety may be best understood and communicated in ways other than probabilistic risk analysis.

4. Most accidents are caused by operator error. Rewarding safe behavior and punishing unsafe behavior will eliminate or reduce accidents significantly.

   Operator error is a product of the environment in which it occurs. To reduce operator “error” we must change the environment in which the operator works.

Assumptions Underlying What We Do

5. Highly reliable software is safe.

   Highly reliable software is not necessarily safe. Increasing software reliability will have only minimal impact on safety.

Software is simply design abstracted from its physical realization.

Black Box Testing

Test data derived solely from specifiation (i.e, without knowledge of internal structure of program).

- Need to test every possible input

```
x := y * 2
if x = 5 then y := 3
```

(Since black box, only way to be sure to detect this is to try envery input condition)

  - Valid inputs up to max size of machine (not astronomical)
  - Also all invalid input (e.g., testing Ada compiler requires all valid and invalid programs)
  - If program has "memory", need to test all possible unique valid and invalid sequences.

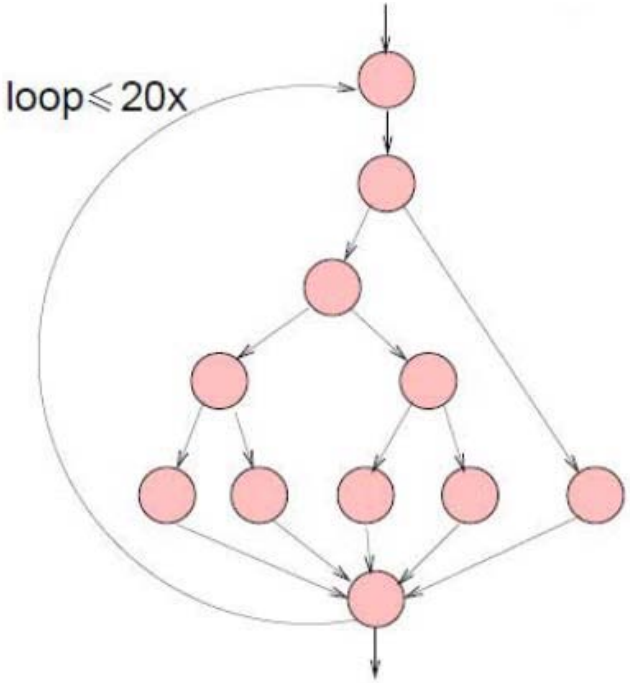
- So for most programs, exhaustive input testing is impractical.

White Box Testing

Derive test data by examining program's logic.

Exhaustic path testing: Two flaws

1) Number of unique paths through program is astronomical.



(control-flow graph)

$$5^{20} + 5^{19} + 5^{18} + \ldots + 5 = 10^{14}$$

= 100 trillion

If could develop/execute/verify one test cases every five minutes = 1 billion years

If had magic test processor that could develop/execute/evaluate one test per msec = 3170 years.

White Box Testing (2)

2) Could test every path and program may still have erroes!

- Does not guarantee program matches specification, i.e., wrong program.
- Missing paths: would not detect absence of necessary paths
- Could still have data-sensitivity errors.

e.g. program has to compare two numbers for convergence

if (A - B) < epsilon ...

is wrong because should compare to abs(A - B)

Detection of this error dependent on values used for A and B and would not necessarily be found by executing every path through program.

Assumptions Underlying What We Do

6. Major accidents occur from the chance simultaneous occurrence of random events.

   Systems tend to migrate toward states of higher risk. Such migration is predictable and can be prevented by appropriate system design or detected during operations using leading indicators or increasing risk.

7. Assigning blame is necessary to learn from and prevent accidents or incidents.

   Blame is the enemy of safety. Focus should be on understanding how the system behavior as a whole contributed to the loss and not on who or what to blame for it.

16.863J / ESD.863J System Safety

Spring 2011