Canadian Nuclear Safety Commission | Commission canadienne de sûreté nucléaire

CMD 18-H6.72A

File / dossier: 6.01.07
Date: 2018-06-12
Edocs: 5558850

Supplementary Information

Oral Presentation

Presentation from Louis Bertrand

In the Matter of

Ontario Power Generation Inc., Pickering Nuclear Generating Station

Request for a ten-year renewal of its Nuclear Power Reactor Operating Licence for the Pickering Nuclear Generating Station

Commission Public Hearing – Part 2

June 2018

Renseignements supplémentaires

Exposé oral

Présentation de Louis Bertrand

À l'égard de

Ontario Power Generation Inc., centrale nucléaire de Pickering

Demande de renouvellement, pour une période de dix ans, de son permis d'exploitation d'un réacteur nucléaire de puissance à la centrale nucléaire de Pickering

Audience publique de la Commission – Partie 2

Juin 2018

Canada

Software Safety Considerations

Louis Bertrand, P.Eng.

Presentation to CNSC Public Hearing 2018-H-03
OPG Pickering NGS Relicensing

June 25-28, 2018

Security ≠ Safety

- Related but distinct properties of a system
- Security:
  - Ability to function correctly and preserve data integrity despite deliberate attempts to cause it to malfunction.
  - Implies agency; actions by an external agent (person(s) or automated)
- Software can be secure but unsafe: resist attack but still operate incorrectly due to design flaws.
- Software can be safe but insecure: operate correctly in the system but be open to attack (e.g. default user ID & password)

Reliability ≠ Safety

“Past performance is no guarantee of future results”

- The fact that the software has operated correctly so far does not automatically imply that it will continue to do so.
- Changing conditions in the process will change the values of the inputs and may affect the flow of control and the outputs.

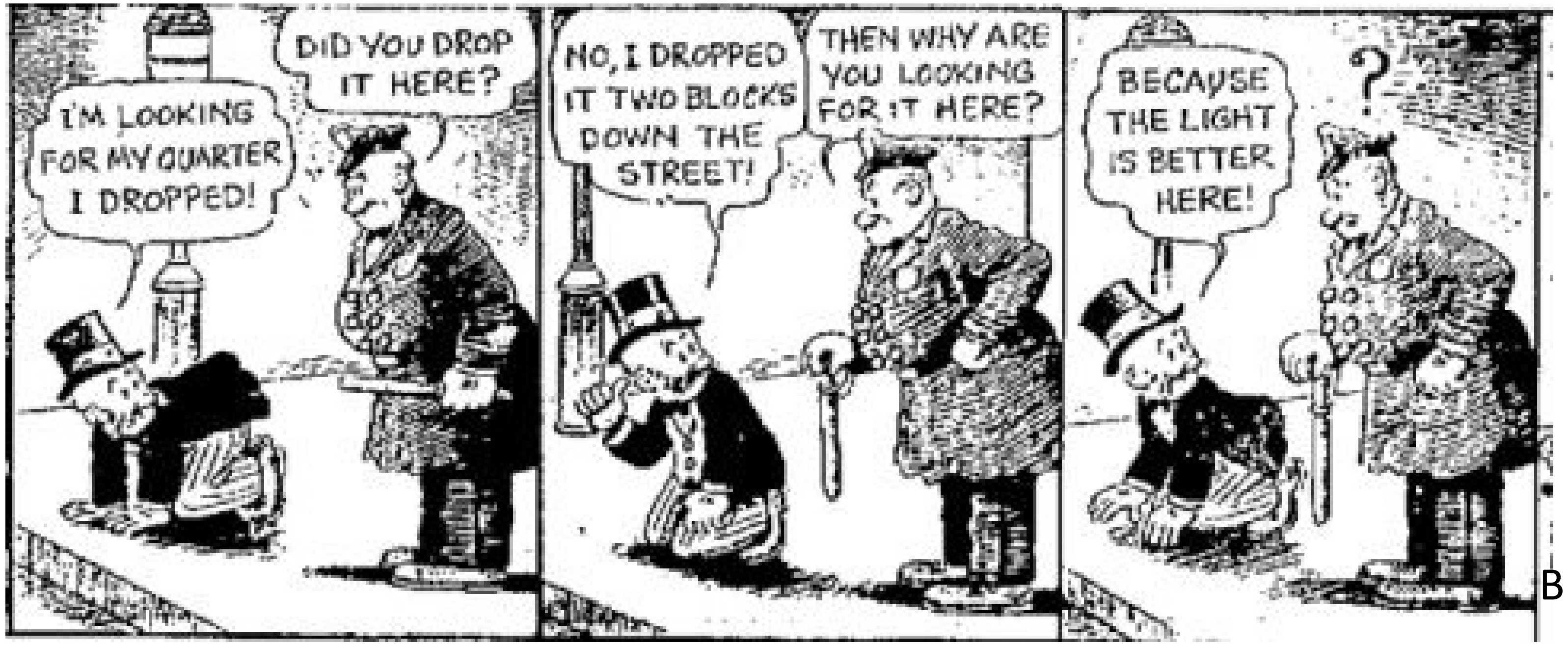
How is safety assessed?
Probabilistic safety assessment (PSA)

- Probability that the events along a branch of a fault tree will result in core damage & release of radioactive substances
- Probabilities of individual events are based on manufacturer’s specifications, history, operation experience, and expert estimates
- Used to set priorities for safety related work given limited resources

Normal Accident Theory (Perrow, 1999)

- "…complex systems like nuclear plants necessarily harbor risks that escape the calculus of formal assessments." (Downer, 2013)
- Paraphrasing Downer's description of Normal Accident Theory:
  - Accidents caused by very improbable confluences of events are "probable" in systems where there are many opportunities for them to occur.
  - In other words, in systems with billions of "billion-to-one" potential accidents, it is to be expected that we would see them from time to time.

The Streetlight Effect: Looks like PSA to me…


I'M LOOKING FOR MY QUARTER I DROPPED!
DID YOU DROP IT HERE?
NO, I DROPPED IT TWO BLOCKS DOWN THE STREET!
THEN WHY ARE YOU LOOKING FOR IT HERE?
BECAUSE THE LIGHT IS BETTER HERE!
?

Bud Fisher

- We can assign a number to the risk, but is the risk fully assessed?
- Is the number realistic? Where does the number come from?
- And how do you estimate the failure rate of software?

Software makes general purpose hardware into a specific purpose machine

"Software in essence is the design of a machine abstracted from its physical realization. In other words, the logical design of a machine (the software) is separated from the physical design of that machine (the computer hardware)."

Leveson, Nancy G. (2011) *Engineering A Safer World: Systems Thinking Applied to Safety*, p.48 Cambridge, Massachusetts: The MIT Press

Software cannot fail (people make mistakes)

- Software does not wear out, it does not age, it does not fail randomly
- Software embodies the decisions and assumptions of its designers and carries out the instructions coded from that effort
  - Requirements improperly or incompletely understood
  - Specifications incorrectly translated from requirements, incomplete or ambiguous
  - Implementation incorrect – programmer errors or toolchain flaws
  - Hardware limitations (in turn caused by underestimated needs)
  - Inadequate verification (testing does not cover all input combinations)
  - Inadequate validation (final product does not meet actual requirements)

Where was the failure?

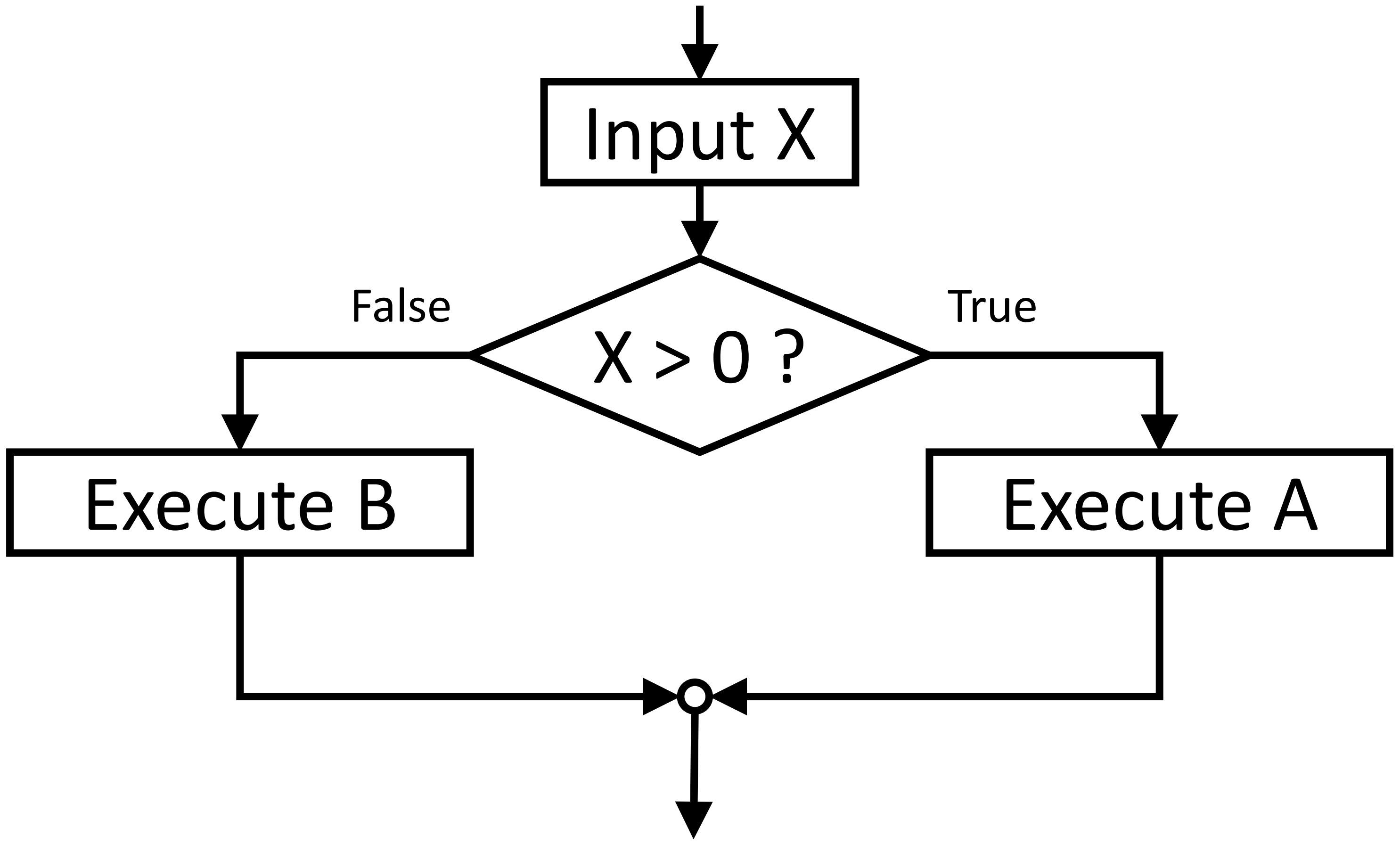
- Requirements capture?
- Specifications drafting?
- Implementation?

What does it mean for software to be “unsafe”?

- System theory models the interactions between components at progressively higher levels of abstraction.
- Tightly coupled systems more vulnerable to disturbances, cascading events or feedback loops
  - Non-linear interactions difficult to model with PSA
- System operation is limited by constraints to keep the system state within safe limits
- By definition, an unsafe state is failure to be bounded by constraints

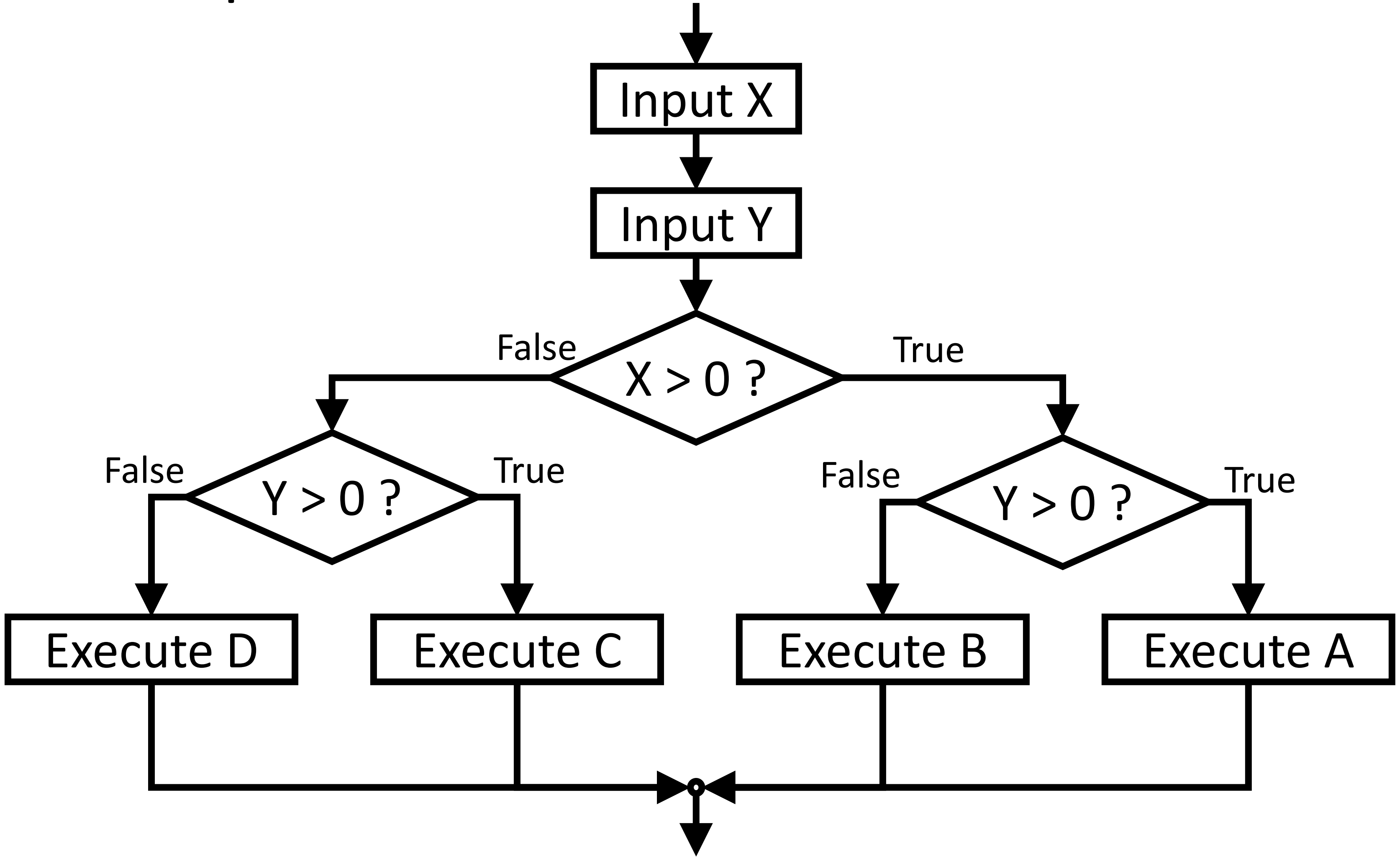
Single branch: 2 paths for control flow

```
Input X
If X > 0
    Then execute A
    Else execute B
```



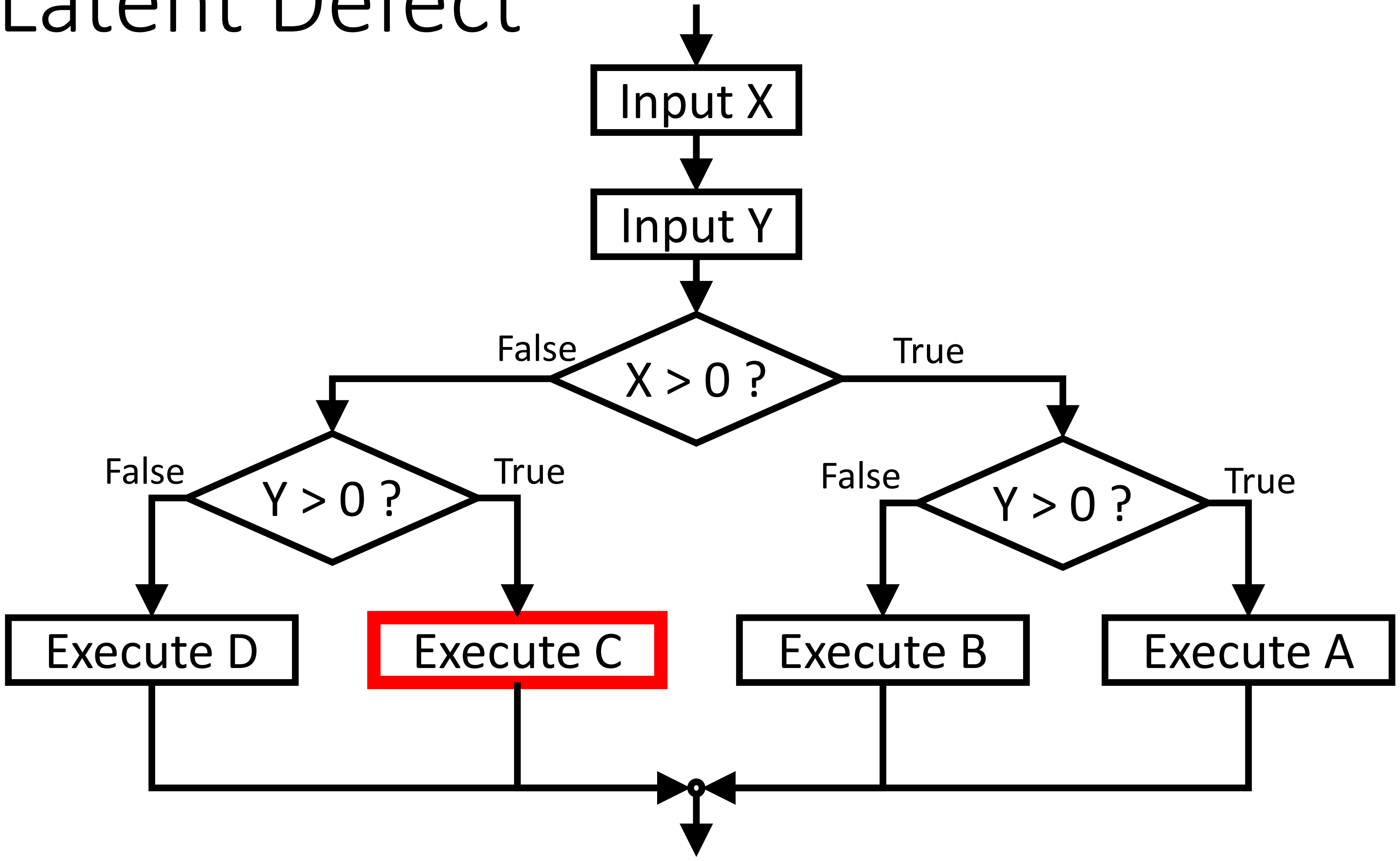
Nested branches: 4 paths for control flow

```
Input X
Input Y
If X > 0
   Then
      If Y > 0
         Then execute A
         Else execute B
   Else
      If Y > 0
         Then execute C
         Else execute D
```



Scenario: The Latent Defect

- Suppose procedure C has an error.
- In previous operation, the flow of control only ever executed procedures A, B and D.
- Changing conditions now cause procedure C to be executed.
- The defect results in an incorrect computation.

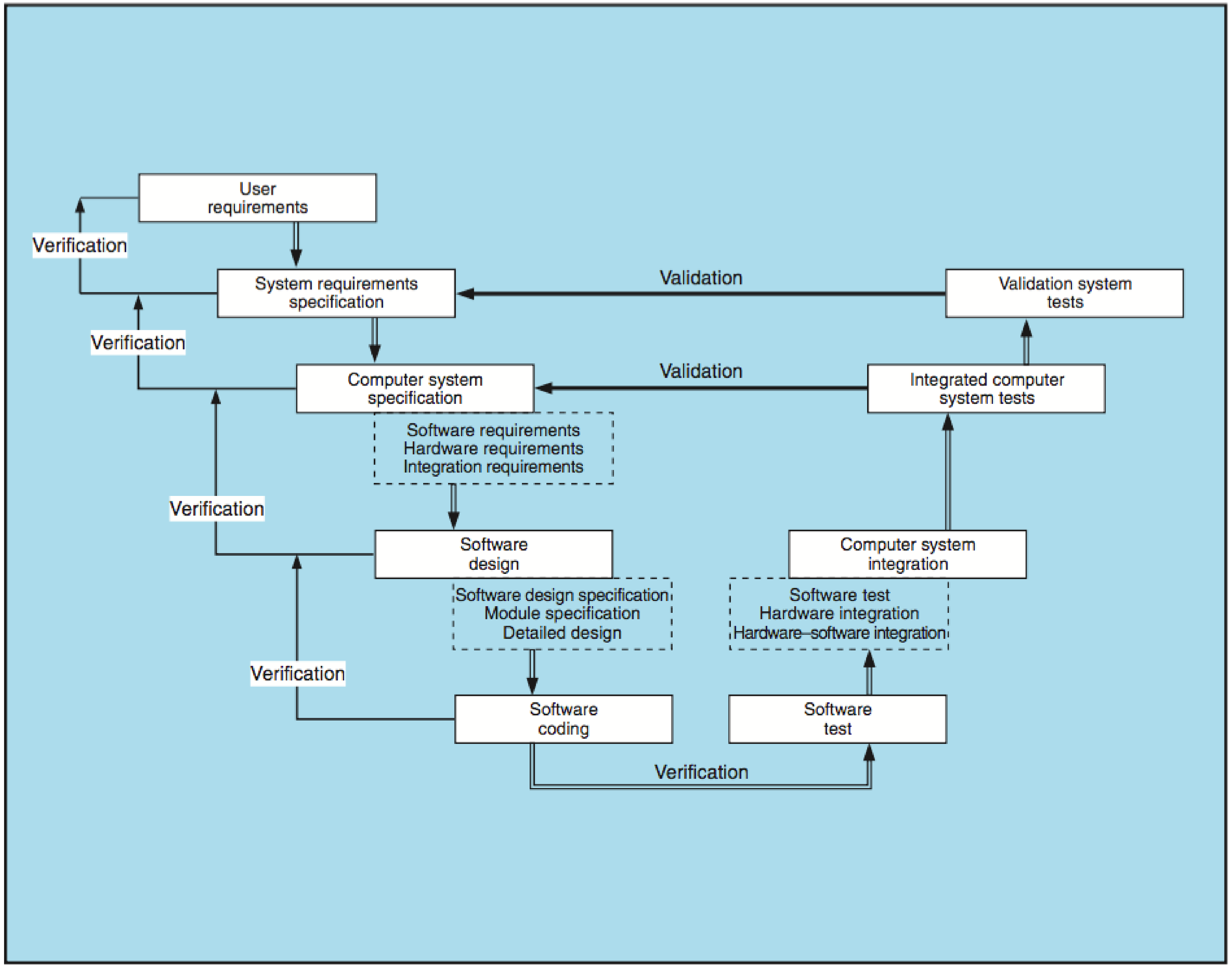


How to ensure correct software?

- CNSC and CSA guidance documents describe the development process activities but are short on details
- IAEA TRS-384* (appendix III.1 – Experience from Canada): Process followed for automated control and shutdown systems at Pickering, Darlington and Wolsong (S.Korea).
  - Hazard identification and mitigation
  - V-model of software development
  - Formal methods
  - Vendor and COTS qualification

* Verification and Validation of Software Related to Nuclear Power Plant Instrumentation and Control, TRS-384, IAEA, Vienna (1999)

V-model
IAEA
TRS-384



TECHNICAL REPORTS SERIES No. **384**

Concerns about software at Pickering NGS

- Aging instrumentation and control hardware
  - 1970s era DEC PDP-11 minicomputers long out of production
- "Software rot" – the tendency for software to "erode" over time; the software doesn't change, but the operating environment does.
- Have conditions changed enough to require updating software?
  - New instrumentation (microcontroller based, sampled vs. continuous, etc.)
  - New end effectors and device controllers (solenoids, pumps, valves, motors, etc.)
- If software updated, what process will be followed?
- If software is not updated, are latent defects ready to be triggered?
- Has OPG as an institution maintained expertise that was developed over 30 years ago?

Questions for OPG Pickering

1. How is the instrumentation and control digital system addressed as part of the station aging management plan? (per CNSC RegDoc-2.6.3)
2. Has the digital control software been changed to meet the changes to sensors and end effectors as the station ages?
3. What changes or retrofits to sensors and end effectors are planned that would require changes to the digital control software?
4. Since non-functional requirements impact function, have there been changes to the configuration of the control computer hardware that would require changes to the software?
5. If changes have been or are being made to the software, what is the development process? Is the documentation trail thorough and available for review by independent 3rd party experts in software development?

Thank you

Questions?

cafepress.co.uk