Safety and Hazard Analysis

- An F16 pilot was sitting on the runway doing the pre-flight and wondered if the computer would let him raise the landing gear while on the ground - it did…

- A manufacturer of torpedoes for the U.S. Navy wanted to make a 'safe' torpedo. Their initial solution was to cause the torpedo to self-destruct if it made a 180 degree change in course. On the test run for this new 'safe' torpedo the captain fired the torpedo and nothing happened. So the captain ordered the sub back to base, executing a 180 degree turn…

Nancy Leveson, talk on Software Safety and Reliability

Today's Lecture

1. Intro to Software Engineering
2. Inexact quantities
3. Error propagation
4. Floating-point numbers
5. Design process
6. Teamwork
7. Project planning
8. Decision making
9. Professional Engineering
10. Software quality
11. Software safety
12. Intellectual property

Agenda

- Hazardous software - Therac 25
- Safety analysis
- Software safety analysis
- Canadian success story - Darlington

Therac-25

The Therac-25 is a computer-controlled radiation-therapy machine developed by the Atomic Energy of Canada Limited (AECL), a crown corporation.

Dual-mode linear accelerator
- Electron therapy - 5 to 25 million electron volt (MeV)
- X-ray (or photon) therapy - 25 MeV

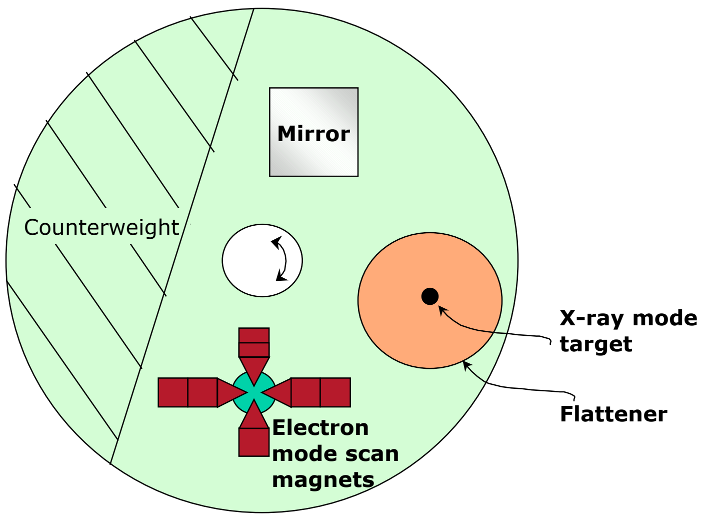
Therac-25 material from
Nancy Leveson and Clark Turner, “An Investigation of the Therac-25 Accidents”, IEEE Computer, July 1993

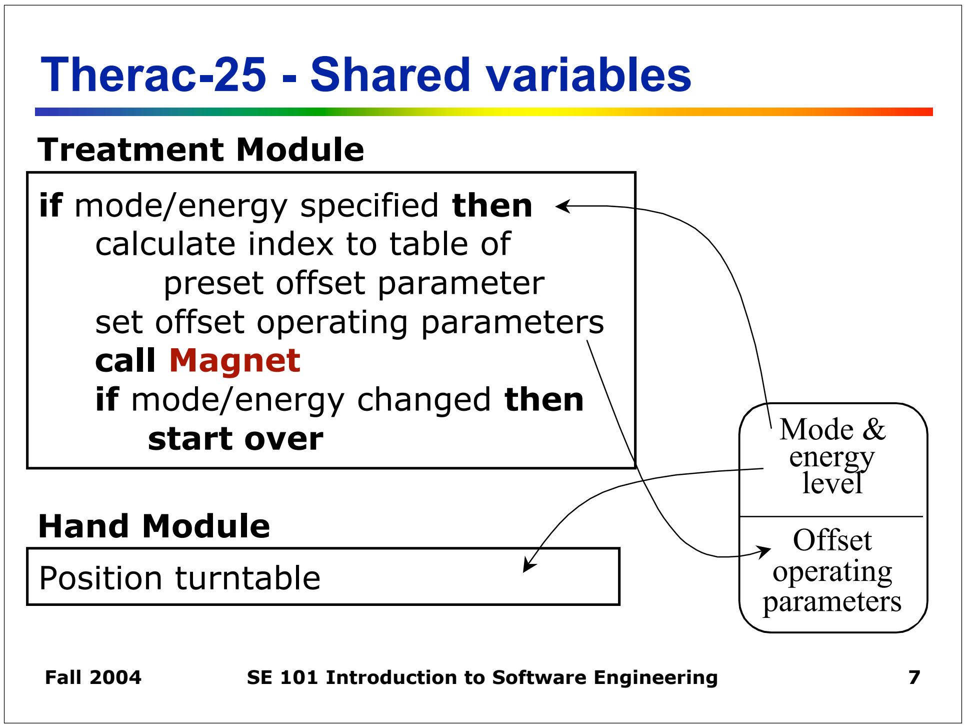
Therac-25

Eleven Therac-25s were installed, five in the States and six in Canada. Between June 1985 and January 1987, there were six massive overdoses.

- Kennestone Regional Oncology Center - paralyzed, in pain
- Ontario Cancer Foundation (Hamilton) - died
- Yakima Valley Memorial Hospital - minor disability, scarring
- East Texas Cancer Center - died
- East Texas Cancer Center - died
- Yakima Valley Memorial Hospital - died

Therac-25





Therac-25 - Shared variables

Treatment Module

```
if mode/energy specified then
    calculate index to table of
         preset offset parameter
    set offset operating parameters
    call Magnet
    if mode/energy changed then
        start over
```

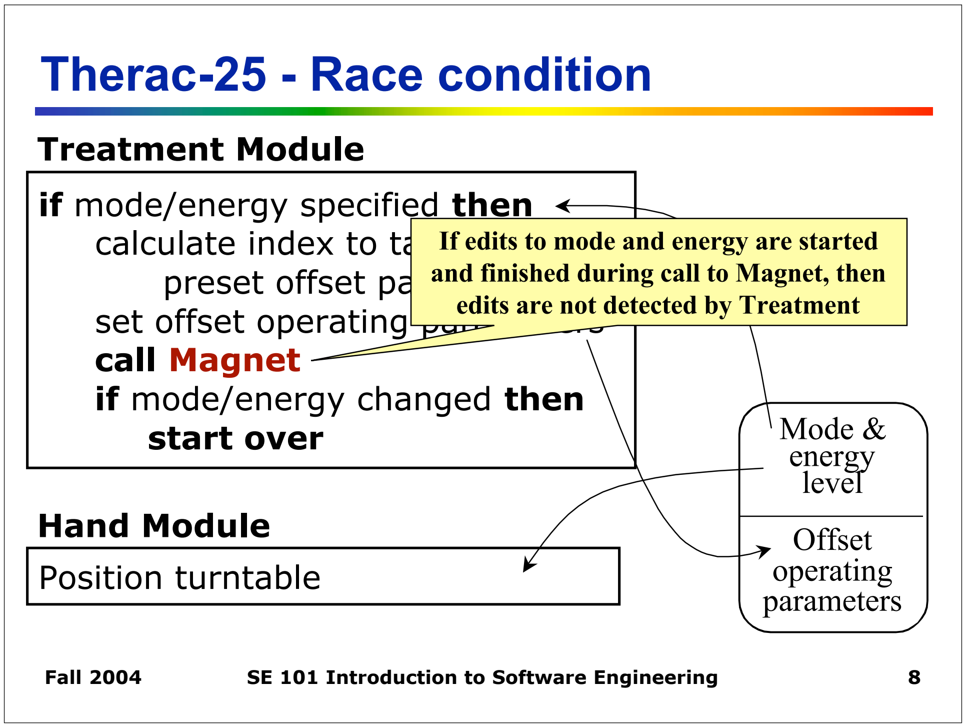
Hand Module

Position turntable

Mode & energy level

Offset operating parameters

Fall 2004 SE 101 Introduction to Software Engineering 7



Therac-25 - Race condition

Treatment Module

```
if mode/energy specified then
    calculate index to table of
         preset offset parameter
    set offset operating parameters
    call Magnet
    if mode/energy changed then
        start over
```

If edits to mode and energy are started and finished during call to Magnet, then edits are not detected by Treatment

Hand Module

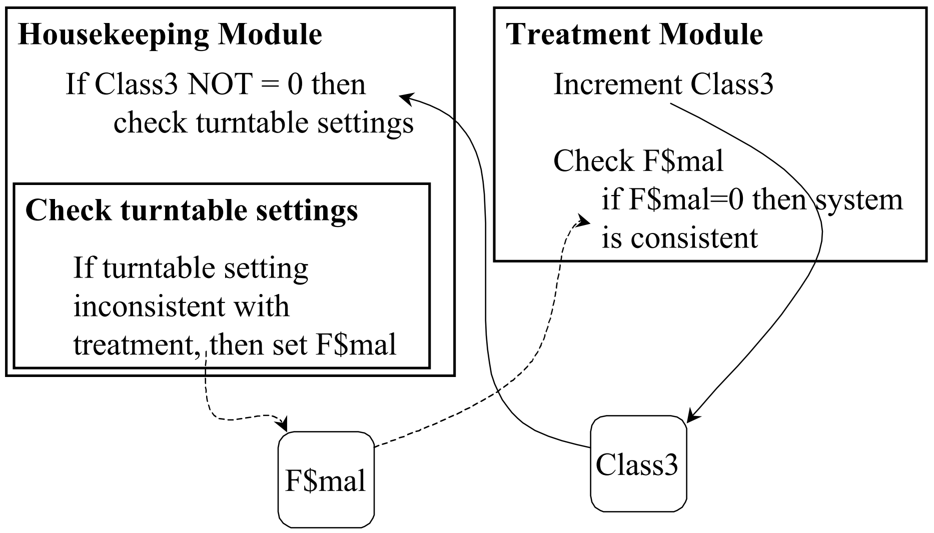
Position turntable

Mode & energy level

Offset operating parameters

Fall 2004 SE 101 Introduction to Software Engineering 8

Therac-25 - “Overflow” error



Housekeeping Module

If Class3 NOT = 0 then
check turntable settings

Check turntable settings

If turntable setting inconsistent with treatment, then set F$mal

Treatment Module

Increment Class3

Check F$mal
if F$mal=0 then system is consistent

Hazardous Software

Software is hazardous if

- It can cause a hazard (i.e., cause other components to become hazardous)
- It is used to control a hazard

NASA Software Safety Guidebook

History of Safety Engineering

- Prior to 1950, engineers used a "fly-fix-fly" process to develop safety-critical systems.
- In the 1950s, the military and aerospace industries started to develop and use predictive safety analysis techniques.
    - Identify hazards
    - Eliminate, reduce, or control hazardous conditions, to avoid or lessen the severity of accidents.

Safety Analysis

Different safety analysis techniques address different aspects of the problem.

- Identify hazards
- Demonstrate the absence of specific hazards.
- Determine the possible damaging effects resulting from hazards
- Determine the causes of a hazard
- Identify safety design criteria that will eliminate, reduce, or control identified hazards.
- Evaluate the adequacy of hazard controls

Safety Analysis

Most safety-analysis techniques are aimed at hardware failures or at external threats

Failure Modes and Effects Analysis (FMEA)

- Identify critical hardware components, interfaces
- Identify possible failure modes for each critical component
- Determine the worst-case effect from each failure mode

Hazard Analyses

- Identify environmental hazards
- Identify deviations in designs
- Identify potential deviations in operational use
- Identify failures at interfaces between components

Software Failures

Software does not break. Software failures are due to logic or design errors:

- The software has no coding errors, but is written from incorrect requirements

- The requirements are correct, but the software has coding errors that deviate from requirements

Thus, software-safety processes often try to improve software safety by improving software correctness.

NASA Space Shuttle Safety Process

- FMEAs are performed on all shuttle hardware and ground support equipment that interfaces with shuttle hardware at launch sites.

- FMEA-identified hazards that could result in loss of life or vehicle, loss of mission, loss of backup systems, are put on the Critical Items List (CIL)

- Closed-loop system for hazard documentation, resolution, and approval: Items on the CIL must be addressed or a waiver request must be approved before the Shuttle can fly.

NASA Space Shuttle Safety Process

NASA did not perform hazard analysis on Shuttle software during the software's development, and it does not perform hazard analysis on software upgrades.

- Striving for correct requirements and code
- Making software fault-tolerant through the use of redundancy

These activities improve software quality and reliability, but they say nothing about whether the software is free of hazards.

Software Safety Analysis

Software Failure Modes and Effects Analysis

Analyzes software components, or interactions between software and hardware components

Software Faults
- Data sampling rate
- Data collisions
- Illegal commands
- Commands out of sequence
- Time delays, deadlines
- Multiple events
- Safe modes

Failures
- Broken sensors
- Memory overwritten
- Missing parameters
- Parameters out of range
- Bad input
- Power fluctuations
- Gamma radiation

NASA Software Safety Guidebook

Software Fault Trees

Software Fault Tree Analysis (SFTA) is a top-down approach to software-failure analysis that starts with an undesired software event and determines all the ways it can happen.

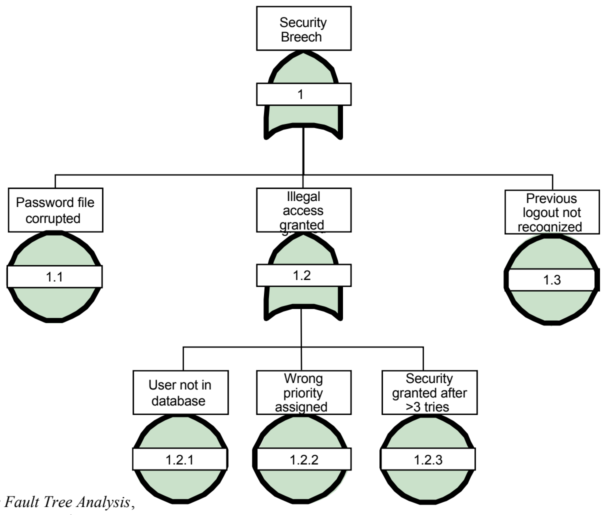


Figure adapted from Wallace et al., *Software Fault Tree Analysis*, NASA Software Assurance Technology Center, December 2003

Therac-25

AECL performed a fault-tree analysis on the Therac-25 that apparently excluded the software. It assumed

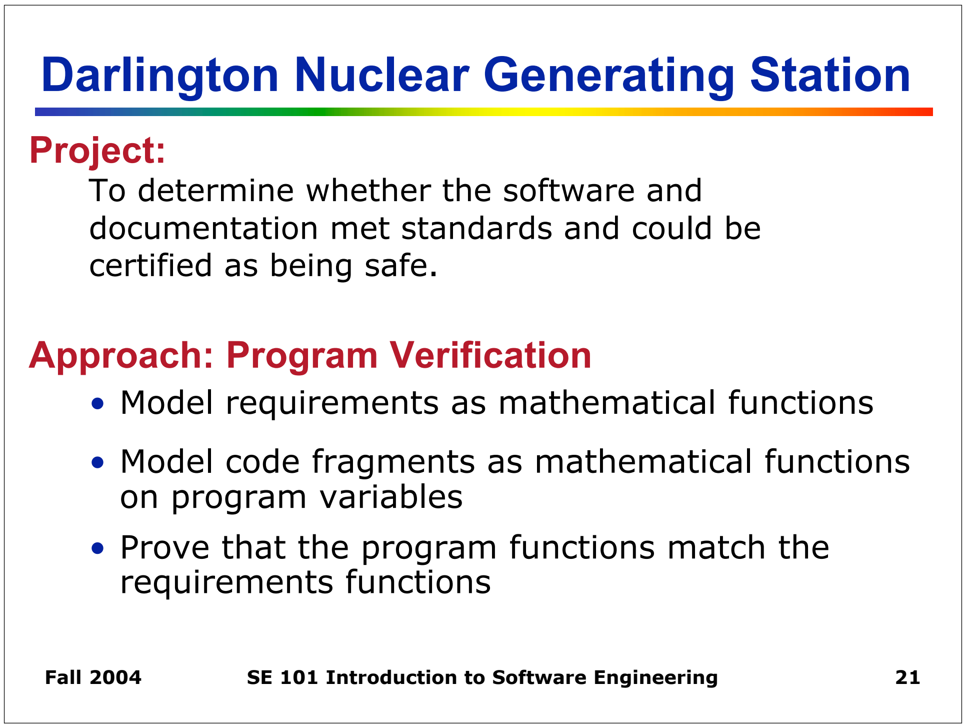
• Programming errors had been reduced by extensive testing on a hardware simulator and under field conditions.

• Computer execution errors are caused by faulty hardware components and by "soft" (random) errors introduced by alpha particles and electromagnetic noise.

Darlington Nuclear Generating Station

• The first nuclear shutdown system implemented completely in software

• Two independent sub-systems, each of which is responsible for shutting down the nuclear reaction in the event of an accident.



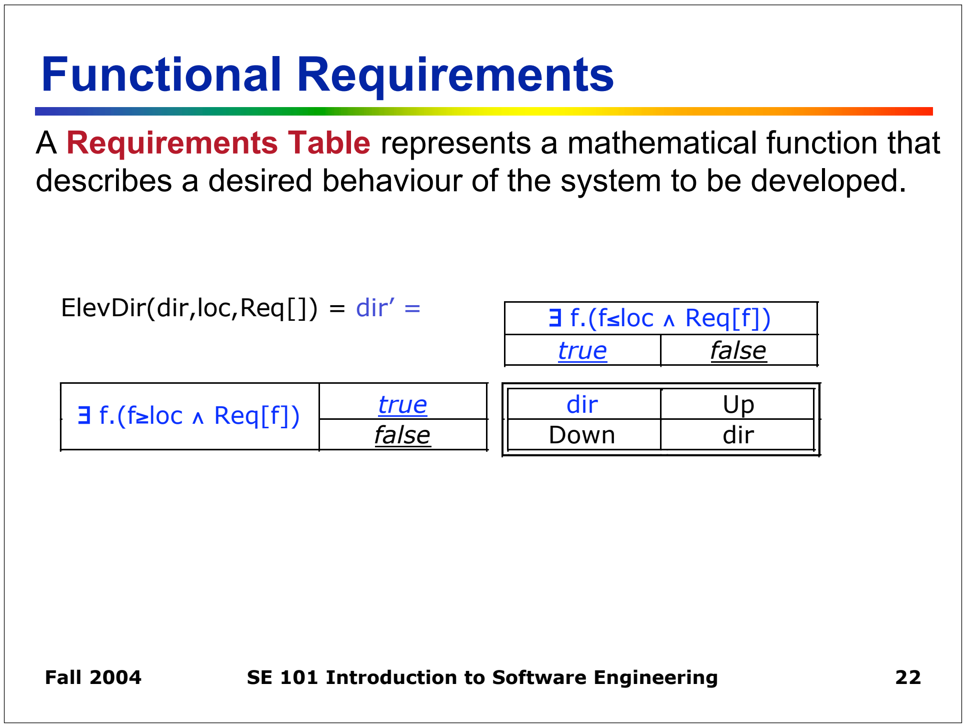
Darlington Nuclear Generating Station

Project:

To determine whether the software and documentation met standards and could be certified as being safe.

Approach: Program Verification

- Model requirements as mathematical functions
- Model code fragments as mathematical functions on program variables
- Prove that the program functions match the requirements functions



Functional Requirements

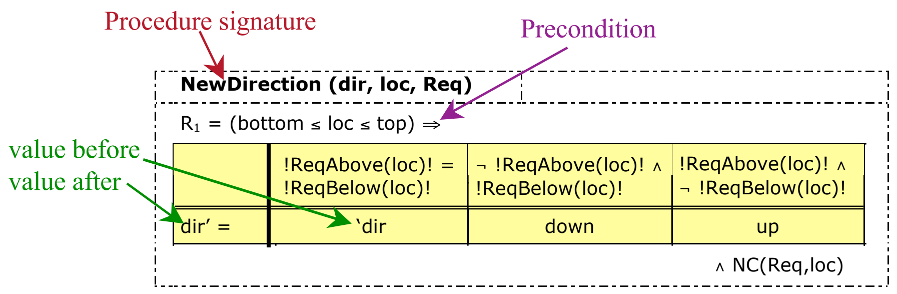
A **Requirements Table** represents a mathematical function that describes a desired behaviour of the system to be developed.

ElevDir(dir,loc,Req[]) = dir′ =

		$\exists f.(f \le loc \wedge Req[f])$	
		true	*false*
$\exists f.(f \ge loc \wedge Req[f])$	*true*	dir	Up
	false	Down	dir

Program Function Tables

A **Program Function Table** represents a mathematical function that describes the behaviour of a procedure.



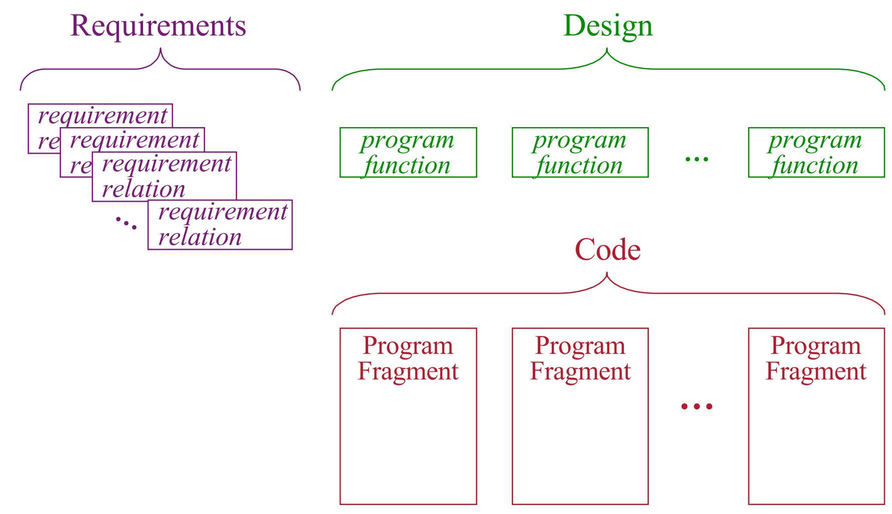
NewDirection (dir, loc, Req)

R_1 = (bottom ≤ loc ≤ top) ⇒

	!ReqAbove(loc)! = !ReqBelow(loc)!	¬ !ReqAbove(loc)! ∧ !ReqBelow(loc)!	!ReqAbove(loc)! ∧ ¬ !ReqBelow(loc)!
dir' =	\`dir	down	up

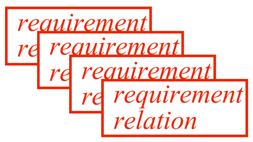
∧ NC(Req,loc)

Systematic Inspections



Reviews and Inspections

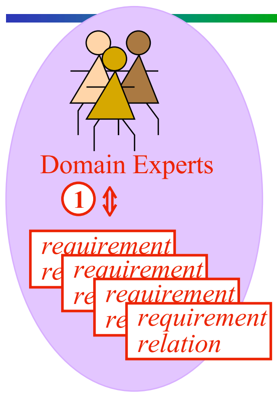
0 Well-formedness of tabular expressions



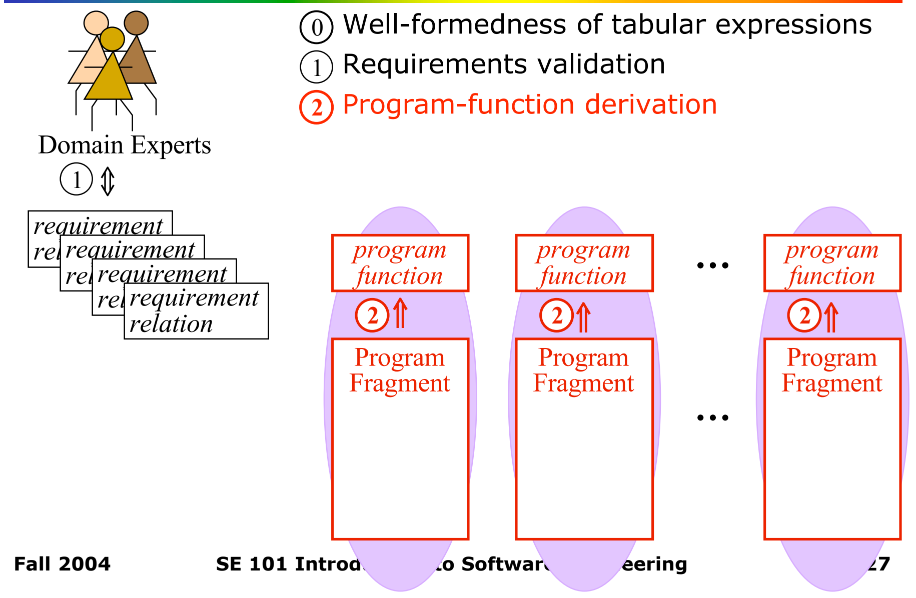
Reviews and Inspections

0 Well-formedness of tabular expressions

1 Requirements validation

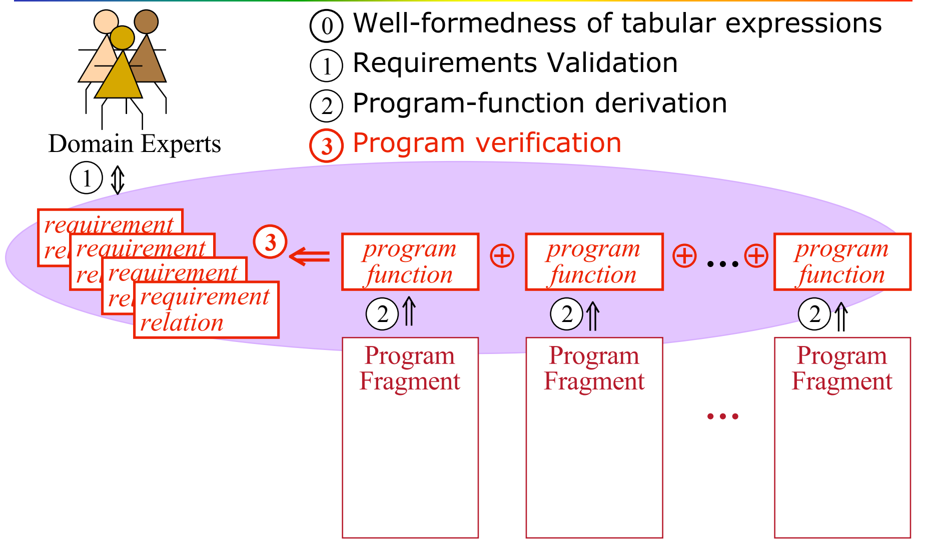


Reviews and Inspections



0 Well-formedness of tabular expressions
1 Requirements validation
2 Program-function derivation

Reviews and Inspections



0 Well-formedness of tabular expressions
1 Requirements Validation
2 Program-function derivation
3 Program verification

Darlington Experience

Experience:

35-person-years task; cost $4 million; found relatively few important discrepancies; but gained confidence in the code

Since then:

Ontario Hydro and the Atomic Energy Canada Limited (AECL) have developed a family of standards, procedures, and guidelines for developing safety-critical software for use in nuclear power plants, incorporating

- mathematical models of requirements, design, and code
- systematic inspections of requirements
- mathematical verification or rigorous argument that
  - the design meets the requirements
  - the code meets the design

Summary

- Ensuring software safety is hard (Therac-25) and expensive (Darlington).
- The state of the practice focuses on software quality and reliability.
- The state of the art attempts to adapt safety analysis techniques so that they apply to software hazards.

Announcements

- In **this week's lab**
  - WHMIS training and quiz
    Required to progress to 1B
    Bring your WATCARD to lab (to get sticker)
  - Promotion Rules
- For **next week's lecture and web review**:
  Intellectual Property IPE Ch. 17
  Word Choice #3 Dupré 20,30,81,127,136

Announcements

- **Quiz #3 in lab next Thursday.**
  Covers lectures on professional engineering, quality attributes, safey, and intellectual property
  - IPE 2, 3, 4, 17, 19, 20

  And word choice
  - Dupré 3,14,17,20,28,30,45,51,61,64,71,
    75,81,99,115,127,129,136
- **SSF Award for Teaching Assistance Excellence**
  - The class collectively nominates one teaching assistant from any of your courses.
  - Nominations due **Friday November 26**
  - Nomination forms **available from Class Reps**