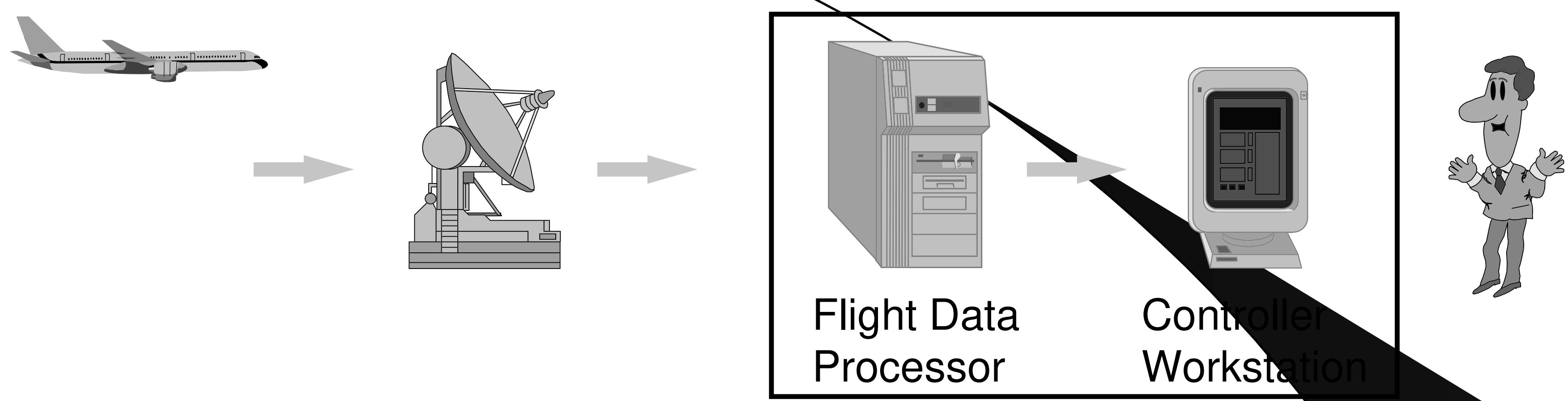
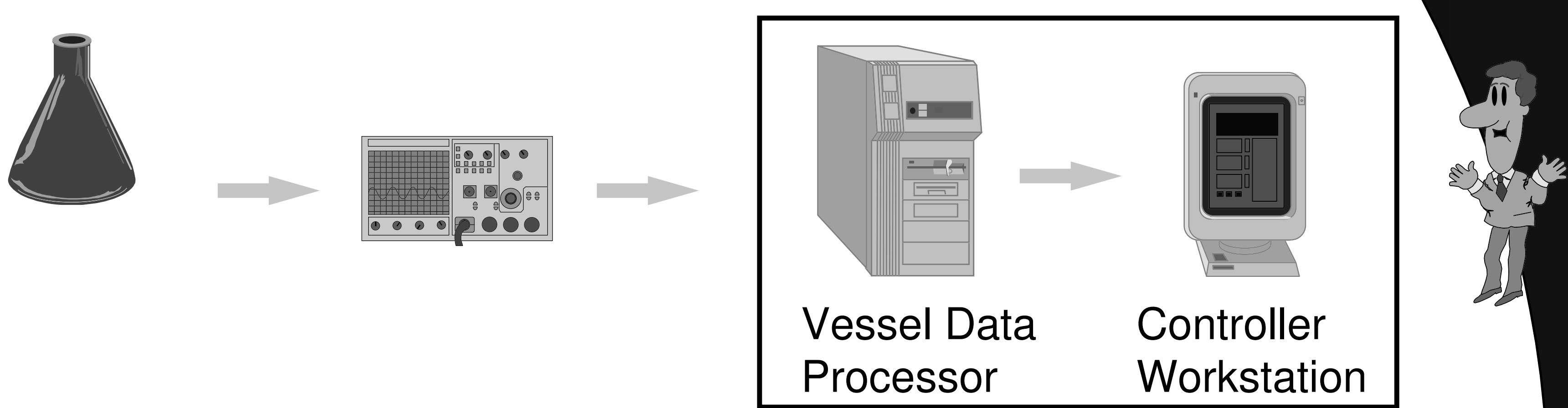
Software Safety Verification

Ken Wong
Department of Computer Science
University of British Columbia

Air Traffic Management (ATM) System



Chemical Factory Management (CFM) System



Vessel Temperature Display

- Functional requirement
  - *Upon receipt of a sensor update containing the information that a vessel V is at temperature D, the system shall update the displayed temperature of vessel V to D within S1 seconds.*

1. Temperature update arrives at time T

Vessel 1: 88.9

Vessel 2: 67.8

Vessel ...

2. Display updated at time T + X , where X < S1

3. Operator uses displayed temperature value to make control decisions

1. No Temperature update arrives after X seconds

Vessel 1: Unavailable

Vessel 2: 67.8

Vessel ...

2. Display temperature is unavailable

3. Operator obtains temperature by manual reading

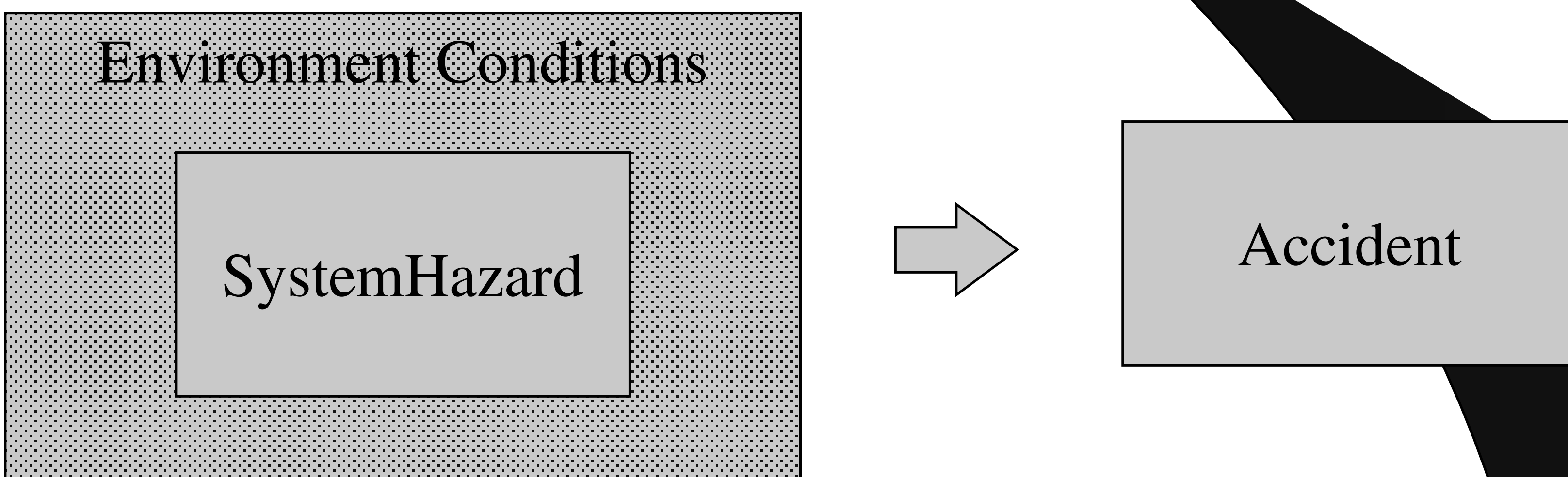
System Safety

- Accidents
  - Property damage, injury or death
- System Hazards
  - System conditions, together with environmental conditions, that will lead to an accident

CFM Safety

- Accidents
  - Vessel rupture, fire or explosion
- A System Hazard
  - *An "invalid" temperature, D, is displayed for vessel V at time T*

System Hazards



Safe Requirements?

If the requirement,

– *Upon receipt of a sensor update containing the information that a vessel V is at temperature D, the system shall update the displayed temperature of vessel V to D within S1 seconds,*

is satisfied, is the hazard,

– *An "invalid" temperature, D, is displayed for vessel V at time T,*

still possible?

Safety vs Correctness

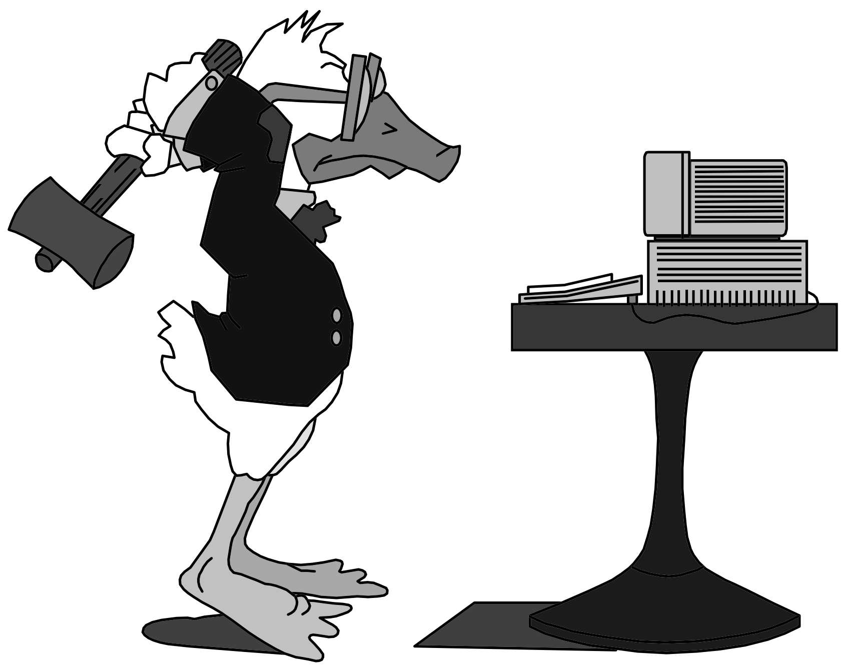
Does the requirement,

- *Upon receipt of a sensor update containing the information that a vessel V is at temperature D, the system shall update the displayed temperature of vessel V to D within S1 seconds,*

have anything do at all with the hazard,

- *An "invalid" temperature, D, is displayed for vessel V at time T?*

System Safety Engineering



Safety Engineering Process

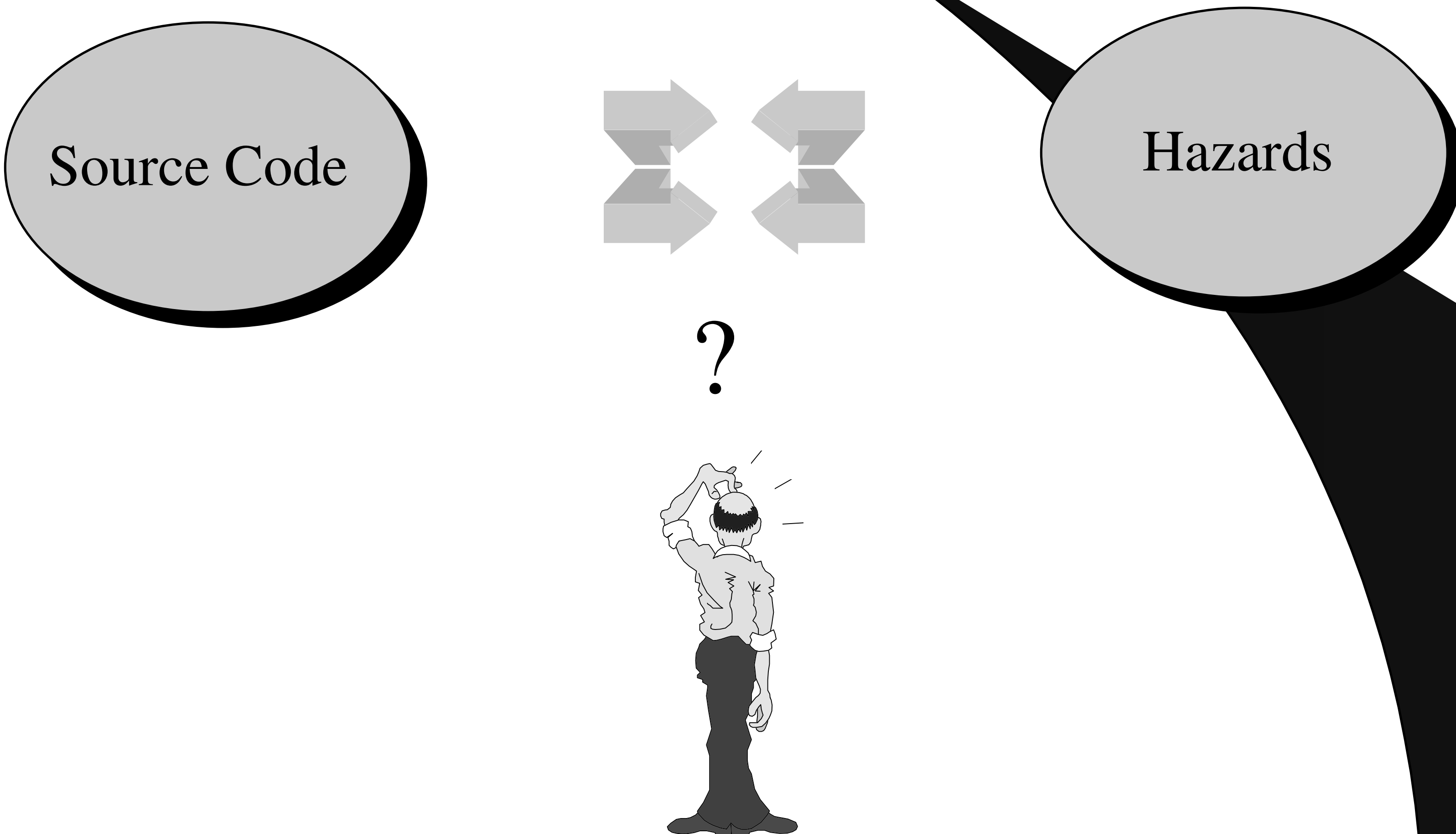
- Focus on system hazards
  - Hazard Analysis
  - Risk Assessment
  - Hazard Mitigation
- Focus on building in safety from the start
  - Involved throughout the system life-cycle
- e.g., CAATS System Safety Engineering

Safety Verification

Unlike the fairy tale Rumplestilskin, do not think that by having named the devil that you have destroyed him. Positive verification of his demise is required.

- System Safety Handbook for Acquisition Manager, U.S. Airforce (quoted in Nancy Leveson's SAFEWARE)

Software Safety Verification



Safety Standards

Safety-Critical Software → Safety Requirements

- IEC 1508, MIL-STD-882C, DO 178B, ...
- e.g., IEC 1508: Software safety validation
  - Demonstrate *safety-related software* is compliant with the *software safety requirements specification*

Safety Requirements

- Basic safety requirement:
  - The hazard shall **not** occur
- Safety requirements from hazard analysis
  - Mitigation of hazard causes

CFM Safety Requirement

- Hazard
  - *An "invalid" temperature, D, is displayed for vessel V at time T*
- Safety Requirement
  - *If temperature D is displayed for Vessel V at time T, then at some time not more than S2 seconds before T, the actual temperature of Vessel V was within C degrees of D.*

Safety-Critical Software

- Source code that may contribute to a hazard
  - Backward trace of the hazard
- "Long thin slice" of hazard-related code
  - Chemical factory example

Safety Verification Techniques

- Dynamic Analysis
  - Functional testing, probabilistic testing, ...
- Static Analysis
  - **Informal** - code inspection, hazard analysis, ...
  - **Formal** - SCR, code verification tools, ...

Limitations of Testing

Program testing can be used to show the presence of bugs, but never to show their absence

– E. Dijkstra in Structured Programming

- Limited in verifying the absence of hazards

Feasibility of Static Analysis

The major drawbacks to static analysis arise simply from practicality and feasibility concerns

– Nancy Leveson in SAFEWARE

- Labor-intensive
- Emphasis on "correctness" verification

Safety Standards' Recomendations

- Combination of static and dynamic analysis
- Choice of techniques guided by level of criticality
- e.g., IEC 1508
  - Four "Safety Integrity Levels" (SIL)
  - "Highly recommends" formal proofs for SIL 4 modules

Software Fault-Injection

- PiSCES SafetyNet (PSN) Tool
- Extended Progagation Analysis (EPA)
  - instrument code with functions that cause corrupted internal states

Extended Propagation Analysis

- Simulate internal code faults and external hardware and software failures
- Input:
  - Set of code inputs (operational profile)
  - Places in code to fault-inject
  - Definition of unacceptable outputs
- Identify sources of software hazards

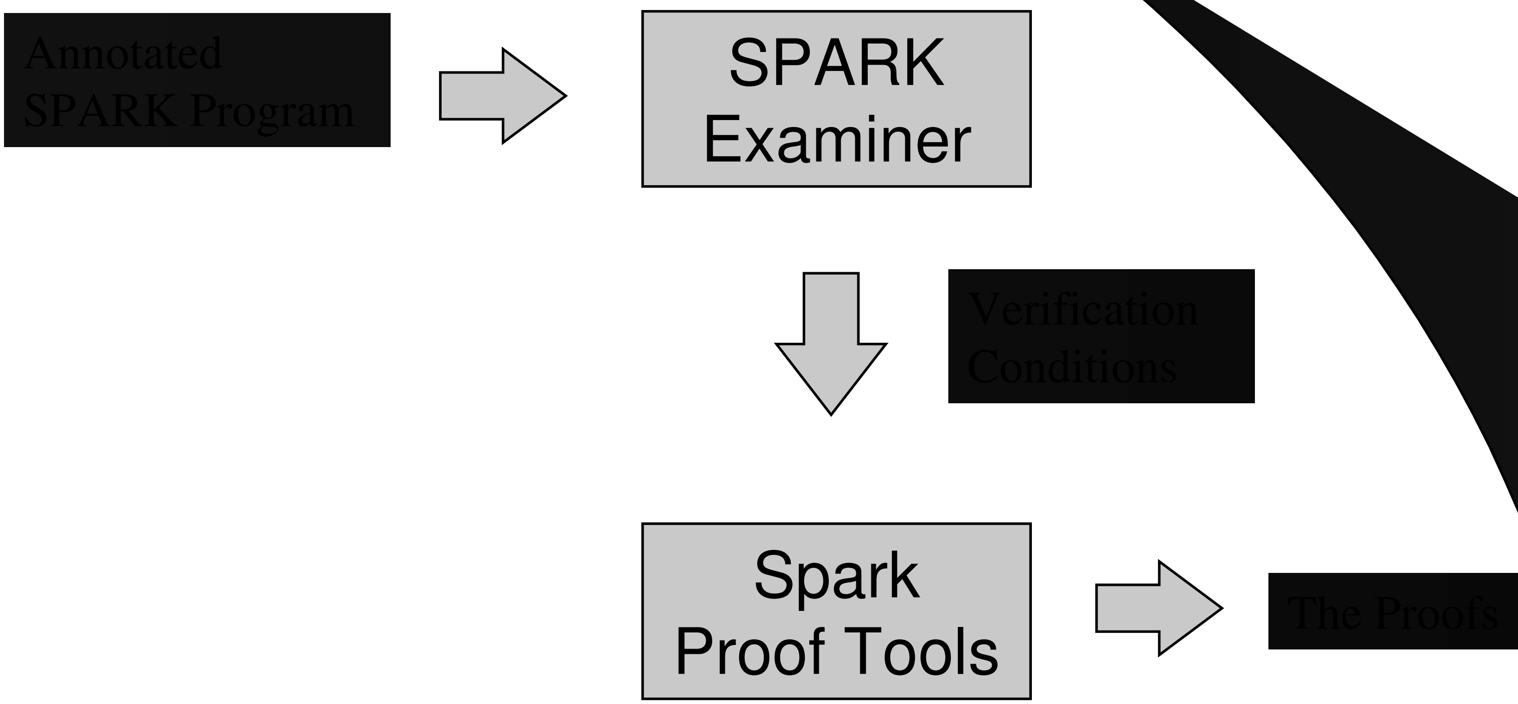
Case Study: Bay Area Rapid Transit

- Hughes Aircraft Company
- Advanced Automated Train Control
- Goal: Identify additional locations for safety code assertions
- One scenario:
  - 2000 locations
  - 12 locations requiring corrective action

Program Verification

- Praxis Critical Systems
- SPARK Examiner
- SPARK Language
  - Ada subset and Annotations
- Proof of program correctness
  - With respect to program pre- and post-conditions

The SPARK Tools



Case Study: C-130J Hercules Aircraft

- Lockheed Aeronautical Systems Company
- Operational Flight Programs
  - Process and transfer information
  - 200 K lines of source code
- Tabular specification of requirements
  - straight-forward code specification and proof
- Correctness (not safety) verification

References

- http://www.cs.ubc.ca/formalWARE/safety.htm
- Program Verification
  - Praxis Critical Systems: http://www.praxis-cs.co.uk/
  - SPARK Examiner: http://www.praxis-cs.co.uk/spark/spark.htm
  - Hercules: http://www.praxis-cs.co.uk/spark/sp_serv.htm
- Software Fault-Injection
  - Reliable Software Technologies: http://www.rstcorp.com/
  - SafetyNet: http://www.rstcorp.com/safnet1.html
  - BART: http://www.rstcorp.com/hughes.html