Software Safety

Designing Safe Software

17-939A

Jennifer Morris

1/27/2005

Safety-Critical Systems Context

- Safety Engineering well established for electromechanical systems
  - Nuclear, military, space, avionics
  - Goal: prevent/minimize hazards, that might lead to mishaps

- Traditional safety design strategies
  - Design for high reliability (10^{-3} - 10^{-9} failures/hour)
    - Redundancy
  - Simple designs facilitate the enforcement of safety invariants
    - Interlocks/lock-in/lock-out
  - Static analysis to identify hazards (HAZOP, FMEA, FTA)

Why is Software Safety Difficult?

- SW design failures are not random & independent
  - HW reliability models don’t apply
  - Redundancy (N-version programming) may not solve problem if common-mode design failures occur

- SW increases system complexity
  - SW is used because it provides complex functionality that is unattainable or impractical in HW
  - Hazard analysis is more difficult
  - Risk increases

How Do We Design Safe Software?

- Recognize that safety is an emergent system property
  - SW is safe only in the context of the system & environment
  - May require trade-offs with other properties (availability/reliability, safety)

- Aim to prevent hazards during design process

- Detect & mitigate hazards at run-time

- Adapt techniques from Safety/Reliability/Security
  - HAZOP, SFTA, FMEA
  - Software interlocks/lock-in/lock-out
  - Modularization to isolate safety invariants
  - Safety monitors
  - Roll-forward recovery
Bibliography