Fault Tolerance and Safety Assurance

Lecture 8

Outline

- Basic Concepts
- Fault Tolerance via RB and NVP
- Safety Assurance Techniques/Strategies
- Summary and Perspectives

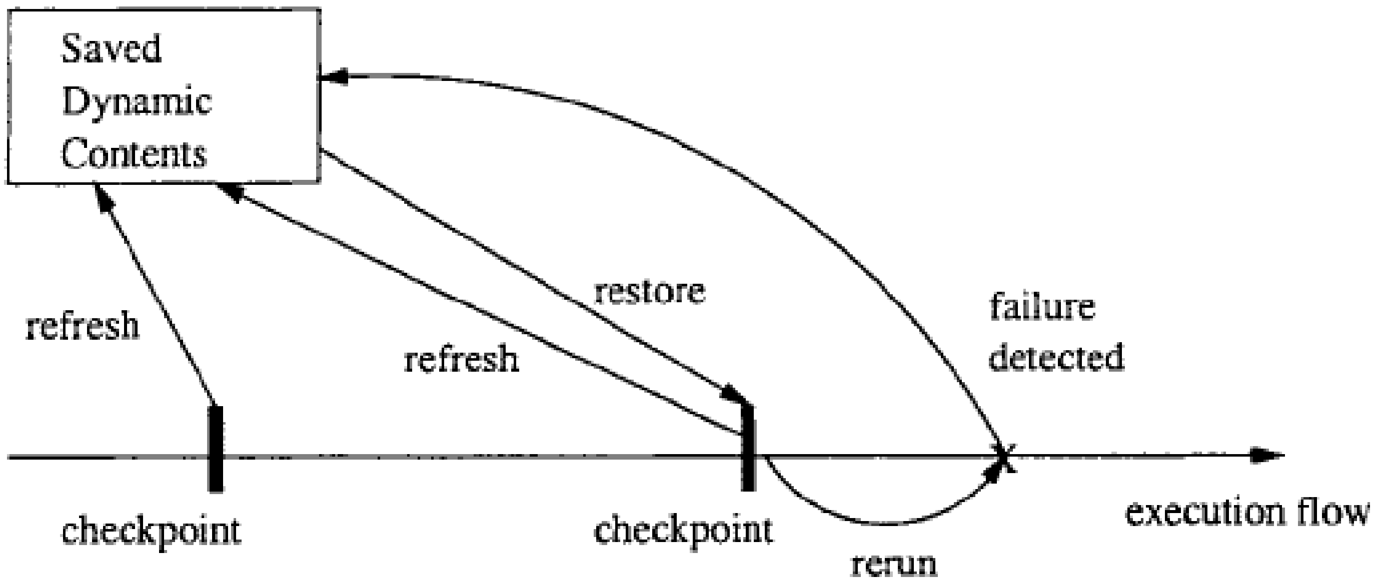
QA Alternatives

- Defect and QA:
  - Defect: error/fault/failure.
  - Defect prevention/removal/containment.
  - Map to major QA activities
- Defect prevention
  - Error source removal & error blocking
- Defect removal: Inspection/testing/etc.
- Defect containment (this lecture)
  - Fault tolerance
    - ✓ local faults ≠> system failures
  - safety assurance - contain failures or weaken failure-accident link

QA and Fault Tolerance

- Fault tolerance as part of QA:
  - Duplication (over time or components) and Backup
  - High cost, high reliability
  - Run-time/dynamic focus
  - FT design and implementation
  - Complementary to other QA activities
- General idea
  - Local faults not lead to system failures
  - Duplication/redundancy used
  - Redo
    - ✓ recovery block (RB)
  - Parallel redundancy
    - ✓ N version programming (NVP)

Fault Tolerance with Recovery Blocks



General idea:

. Periodic checkpointing

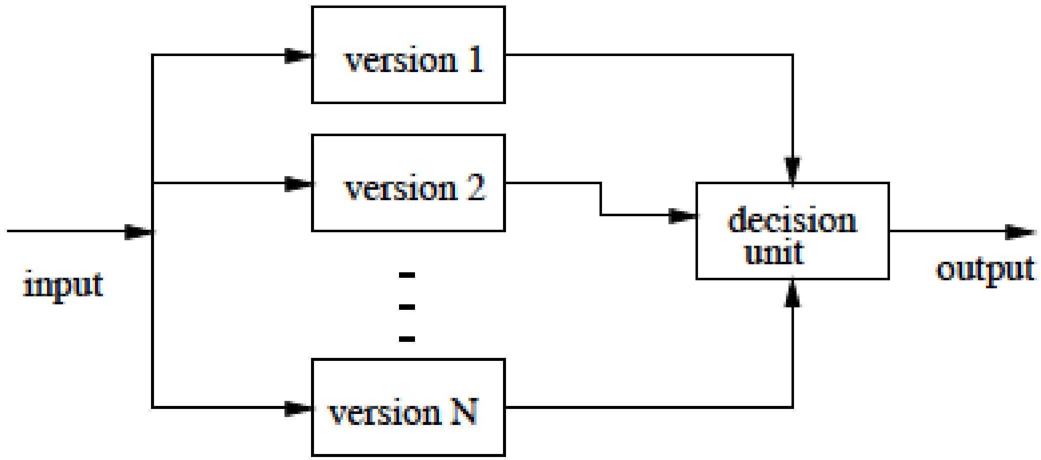
. Problem detection/acceptance test

. Rollback (recovery)

Fault Tolerance with Recovery Blocks 2

- Periodic checkpointing
  - too often: expensive checkpointing
  - too rare: expensive recovery
  - smart/incremental checkpointing
- Problem detection/acceptance test
  - exceptions due to in/external causes
  - periodic vs event-triggered
- Recovery (rollback) from problems:
  - external disturbance: environment?
  - internal faults: tolerate/correct?

Fault Tolerance with N-Version Programming



- FT with NVP:
  - NVP: N-Version Programming
  - Multiple independent versions
  - Dynamic voting/decision => FT.

Fault Tolerance with N-Version Programming 2

- Multiple independent versions
  - Multiple: parallel vs backup?
  - How to ensure independence?
- Support environment:
  - concurrent execution
  - Switching
  - voting/decision algorithms
- Correction/recovery?
  - p-out-of-n reliability
  - in conjunction with RB
  - dynamic vs. off-line correction

FT/NVP: Ensure Independence

- Ways to ensure independence:
  - People diversity:
    - type, background, training, teams, etc.
  - Process variations
  - Technology: methods/tools/PL/etc.
  - End result/product:
    - design diversity: high potential
    - implementation diversity: limited
- Ways to ensure design diversity:
  - People/teams
  - Algorithm/language/data structure
  - Software development methods
  - Tools and environments
  - Testing methods and tools (!)
  - Formal/near-formal specifications

FT/NVP: Development Process

- Programming team independence
  - Assumption: P-team independence => version independence
  - Maximize P-team isolation/independence
  - Mandatory rules (DOs & DON'Ts)
  - Controlled communication (see below)
- Use of coordination team
  - 1 C-team - n P-teams
  - Communication via C-team
    - ✓ not P-team to P-team
    - ✓ protocols and overhead cost
  - Special training for C-team
- NVP-specific process modifications

FT/NVP: Development Phases

- Pre-process training/organization
- Requirement/specification phases:
  - NVP process planning
  - Goals, constraints, and possibilities
  - Diversity as part of requirement
    - relation to and trade-off with others
    - achievable goals under constraints
  - Diversity specification
- Design and coding phases:

  enforce NVP-process/rules/protocols

FT/NVP: Development Phases 2

- Testing phases:
  - Cross-checking by different versions - free oracle!
  - Focus on fault detection/removal
  - Focus on individual versions
- Evaluation/acceptance phases:
  - How N-versions work together?
  - Evidence of diversity/independence?
  - NVP system reliability/dependability?
  - Modeling/simulation/experiments
- Operational phase:
  - Monitoring and quality assurance
  - NVP-process for modification also

FT and Safety

- Extending FT idea for safety:
  - FT: tolerate fault
  - Extend: tolerate failure
  - Safety: accident free
  - Weaken error-fault-failure-accident link
- FT in SSE (software safety engineering):
  - Too expensive for regular systems
  - As hazard reduction technique in SSE
  - Other related SSE techniques:
    - ✓ general redundancy
    - ✓ substitution/choice of modules
    - ✓ barriers and locks
    - ✓ analysis of FT

What Is Safety?

- Safety: The property of being accident-free for (embedded) software systems.
  - Accident: failures with severe consequences
  - Hazard: condition for accident
  - Special case of reliability
  - Specialized techniques
- Software safety engineering (SSE):
  - Hazard identification/analysis techniques
  - Hazard resolution alternatives
  - Safety and risk assessment
  - Qualitative focus
  - Safety and process improvement

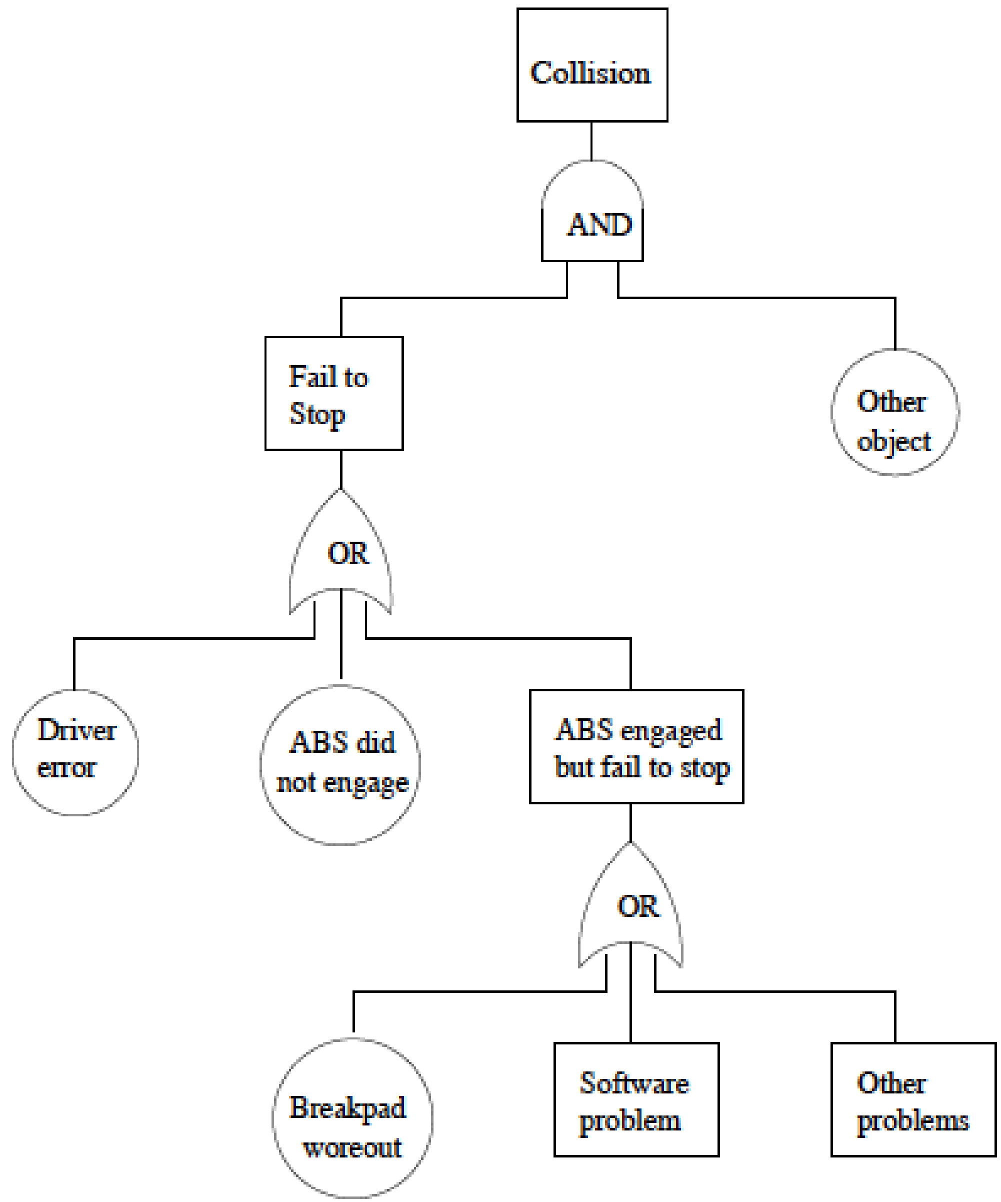
Safety Analysis & Improvement

- Hazard analysis:
  - Hazard: condition for accident
  - Fault trees: (static) logical conditions
  - Event trees: dynamic sequences
  - Combined and other analyses
  - Generally qualitative
  - Related: accident analysis and risk assessment
- Hazard resolution
  - Hazard elimination
  - Hazard reduction
  - Hazard control
  - Related: damage reduction

Hazard Analysis: FTA

- Fault tree idea:
  - Top event (accident)
  - Intermediate events/conditions
  - Basic or primary events/conditions
  - Logical connections
  - Form a tree structure
- Elements of a fault tree:
  - Nodes: conditions and sub-conditions
    - ✓ terminal vs. no terminal
  - Logical relations among sub-conditions
    - ✓ AND, OR, NOT
  - Other types/extensions possible

Hazard Analysis: FTA Example



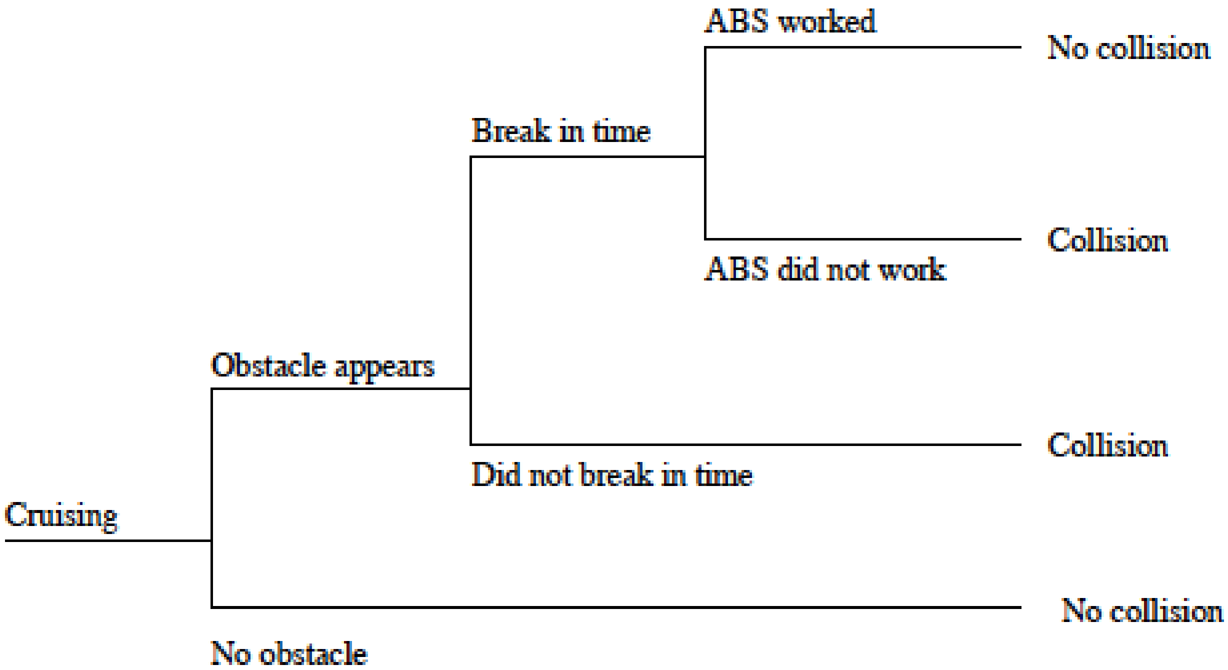
Hazard Analysis: FTA

- **FTA construction:**
  - Starts with top event/accident
  - Decomposition of events or conditions
  - Stop when further development not required or not possible (atomic)
  - Focus on controllable events/elements
- **Using FTA:**
  - Hazard identification
    - ✓ *logical* composition
    - ✓ (vs. *temporal* composition in ETA)
  - Hazard resolution (more later)
    - ✓ component replacement etc.
    - ✓ focused safety verification
    - ✓ negate logical relation

Hazard Analysis: ETA

- ETA: Why?
  - FTA: focus on static analysis
    - (static) logical conditions
  - Dynamic aspect of accidents
  - Timing and temporal relations
  - Real-time control systems
- Search space/strategy concerns:
  - Contrast ETA with FTA:
    - FTA: backward search
    - ETA: forward search
  - May yield different path/info.
  - ETA provide additional info.

Hazard Analysis: ETA Example



Hazard Analysis: ETA

- **Event trees:**
  - Temporal/cause-effect diagram
  - (Primary) event and consequences
  - Stages and (simple) propagation
    - ✓ not exact time interval
    - ✓ logical stages and decisions
- **Event tree analysis (ETA):**
  - Recreate accident sequence/scenario
  - Critical path analysis
  - Used in hazard resolution (more later)
    - ✓ esp. in hazard reduction/control
    - ✓ e.g. creating barriers
    - ✓ isolation and containment

Hazard Elimination

- Hazard sources identification => elimination

  (Some specific faults prevented or removed.)
- Traditional QA (but with hazard focus):
  - Fault prevention activities:
    - education/process/technology/etc
    - formal specification & verification
  - Fault removal activities:
    - rigorous testing/inspection/analyses
- “Safe" design: More specialized techniques:
  - Substitution, simplification, decoupling.
  - Human error elimination.
  - Hazardous material/conditions↓.

Hazard Reduction

- Hazard identification => reduction

  (Some specific system failures prevented or tolerated.)
- Traditional QA (but with hazard focus):
  - Fault tolerance
  - Other redundancy
- “Safe" design: More specialized techniques:
  - Creating hazard barriers
  - Safety margins and safety constraints
  - Locking devices
  - Reducing hazard likelihood
  - Minimizing failure probability
  - Mostly “passive" or “reactive"

Hazard Control

- Hazard identification => control
  - Key: failure severity reduction.
  - Post-failure actions.
  - Failure-accident link weakened.
  - Traditional QA: not much, but good design principles may help.
- “Safe" design: More specialized techniques:
  - Isolation and containment
  - Fail-safe design & hazard scope ↓
  - Protection system
  - More “active" than “passive”
  - Similar techniques to hazard reduction,
    - ✓ but focus on post-failure severity ↓
    - ✓ vs. pre-failure hazard likelihood ↓.

Accident Analysis & Damage Control

- Accident analysis:
  - Accident scenario recreation/analysis
    - ✓ possible accidents and damage areas
  - Generally simpler than hazard analysis
  - Based on good domain knowledge

    (not much software specifics involved)
- Damage reduction or damage control
  - Post-accident vs. pre-accident hazard resolution
  - Accident severity reduced
  - Escape route
  - Safe abandonment of material/product/etc.
  - Device for limiting damages

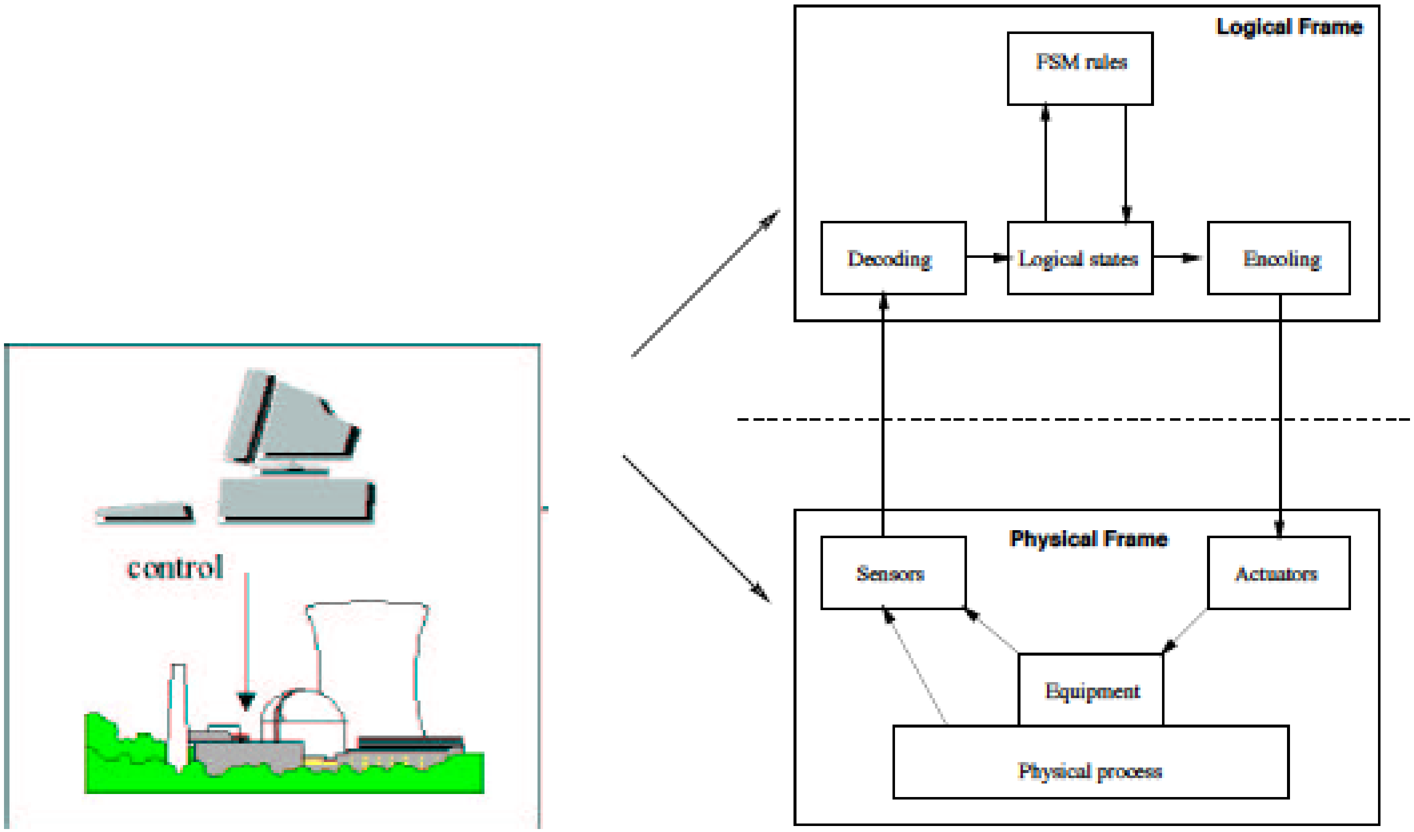
Application in Heterogeneous systems

- Heterogeneous or embedded system required high dependability and safety
- Fault tolerance and failure containment techniques are generally suitable for such systems
- System reliability
  - deals with hardware and communication/interaction problems
  - is the probability of failure-free operations for the whole system for a given time period or under a given set of usage scenarios
- System dependability
  - broader concept
  - includes reliability, fault tolerance, safety, etc., all related to how likely or how much a system can be depended upon

TFM: Two-Frame-Model

- TFM: Two-Frame-Model
  - Physical frame
  - Logical frame
  - Sensors: physical => logical
  - Actuators: logical => physical
- TFM characteristics and comparison:
  - Interaction between the two frames
  - Nondeterministic state transitions and encoding/decoding functions
  - Focuses on symmetry/consistency between the two frames.

TFM Example



. physical frame: nuclear reactor
. logical frame: computer controller

Usage of TFM

- Failure/hazard sources and scenarios:
  - Hardware/equipment failures.
  - Software failures.
  - Communication/interface failures.
  - Focus on last one, based on empirical evidence.
- Causes of communication/interface hazards:
  - Inconsistency between frames.
  - Sources of inconsistencies
  - Use of prescriptive specifications (PS)
  - Automatic checking of PS for hazard prevention

Summary

- Software fault tolerance:
  - Duplication and redundancy.
  - Techniques: RB, NVP, and variations.
  - Cost and effectiveness concerns.
- SSE: Augment S/w Eng.
  - Analysis to identify hazard
  - Design for safety
  - Safety constraints and verification
  - Cost and application concerns.


Q
A