Introduction to Software Fault Tolerance Techniques and Implementation

Presented By:
Hoda Banki
Contents:

• Introduction
• Types of faults
• Dependability concept classification
• Error recovery
• Types of redundancy
Introduction:

- **Safe** and **reliable** software operation is a significant requirement
  - for instance: aircraft, medical devices, nuclear safety
  - appliance-type applications: automobiles, washing machines, temperature control

- Examples about faults in softwares:
  - software on space shuttle Endeavor rounded near-zero values to zero
  - Problems in Airbus A320
Introduction to Software Fault Tolerance Techniques and Implementation

Introduction (...)

Faults:

• Hardware Faults:
  - physical faults
  - can be characterized
  - predicted

• Software Faults:
  - logical faults
  - difficult to visualize, classify, detect, and correct

• Software faults may be traced to:
  - incorrect requirements
  - implementation (software design, coding) not satisfy requirements
A few definition:

Cycle:

... failure ➞ fault ➞ error ➞ failure ➞ Fault ...

• software fault tolerance prevents failures by tolerating faults

☑ Fault tolerance techniques:

Chapter 2 : various means of structuring redundancy
Chapter 3 : Some programming methods
Chapter 4 : design diverse techniques (Recovery)
Chapter 5 : data diverse techniques
Introduction to Software Fault Tolerance Techniques and Implementation

- **Background and basics**
  (Chapters 1, 2, 3)

- **Basic**
  - Basic and classic techniques
    (Sections 4.1, 4.2, 4.7.1, 5.1, 5.2)
  - Basic decision mechanisms
    (Sections 7.1.1–7.1.3, 7.1.7, 7.2)

- **Advanced**
  - More complex techniques
    (Sections 4.3–4.6, 4.7.2–4.7.4, 5.3, Chapter 6)
  - Advanced decision mechanisms
    (Sections 7.1.4–7.1.6, 7.1.8)

Alternate path

---

6
Means to achieve dependable software:

- Dependability
  - Means
    - Construction
    - Validation
      - Fault avoidance
      - Fault tolerance
      - Fault removal
      - Fault forecasting
  - Attributes
    - Availability
    - Reliability
    - Safety
    - Confidentiality
    - Integrity
    - Maintainability
  - Impairments
    - Faults
    - Errors
    - Failures
Means:

• **Fault avoidance or prevention:** to avoid or prevent fault occurrence

• **Fault removal:** to detect the existence of faults and eliminate them

• **Fault/failure forecasting:** to estimate the presence of faults and the occurrence and consequences of failures

• **Fault tolerance:** to provide service complying with the specification in spite of faults
1) System Requirements Specification:

- behavior specified in the requirements is not the expected or desired system behavior
- software requirements specification lies at the intersection between software engineering and system engineering

2) Structured Design and Programming Methods:

- introduce structure to the design to reduce the complexity and interdependency of components
- reduces overall complexity of the software, making it easier to implement and reduces the introduction of faults
3) Formal Methods:

- requirements specifications are developed mathematically
- same size as the program
- difficult to construct
- harder to understand than the program itself
- prone to error
- suitable for small components that are highly critical

4) Software Reuse:

- savings in development cost
- has been well exercised is less likely to fail
Fault removal:

1) Software Testing:
   - Difficulties: **cost** and **complexity** of exhaustive testing using all possible input sets
   - can show the presence, but not the absence of faults

2) Formal Inspection:
   - Has been widely implemented in industry
   - Focuses on: examining **source code** to find faults correcting the faults verifying the corrections
   - prior to the testing phase
3) Formal Design Proofs:

- closely related to formal methods
- attempts to achieve mathematical proof of correctness for programs
- costly and complex technique
Fault/failure forecasting:

1) Reliability Estimation:
   - determines **current** software reliability
   - applying statistical inference techniques to failure data

2) Reliability Prediction:
   - determines **future** software reliability
   - different techniques, depending on software development stage
Fault tolerance:

- When a fault occurs, provide mechanisms to prevent system failure
- Provide protection against errors in translating the requirements and algorithms into a programming language
- Do not provide explicit protection against errors in specifying the requirements
Error recovery:

Error detection: an erroneous state is identified

Error diagnosis: the damage caused by the error is assessed
the cause of the error is determined

Error containment/isolation: error is prevented from propagating

Error recovery: the erroneous state is substituted with an error-free state
Types of recovery:

1) Backward Recovery:
   - restoring or rolling back the system to a previously saved state
   - checkpointing: saving previous state
   - system state is restored to the last saved state
   - operations continue or restart from that state
Backward recovery:

- **Checkpoint**
  - Fault detection
  - Fault detected

- **Rollback**
  - Restore checkpoint
  - Recovery point

Fault tolerated
Advantage of backward recovery:

- handle **unpredictable** errors caused by design faults
- has a uniform pattern of error detection and recovery
- only knowledge required the relevant prior state is error-free
- particularly suited to recovery of transient faults
Disadvantage of backward recovery:

- requires significant resources (time, computation, stable storage)
- requires the system be halted temporarily
- domino effect may occur
Forward recovery:

- finding a new state from which the system can continue operation uses redundancy
- redundant software processes are executed in parallel
- uses voting schema to select correct result
**Advantage of forward recovery:**

- efficient in terms of the overhead (time and memory)
- suitable for **real-time** applications
- suitable for predicted faults

**Disadvantage of forward recovery:**

- it must be tailored to each situation or program
- can only remove predictable errors
- requires knowledge of the error

✓ *Forward recovery is primarily used when there is no time for backward recovery*
Types of redundancy:

1) hardware redundancy:
   - includes replicated and supplementary hardware added to the system to support fault tolerance
   - Hardware faults are typically random, due to component aging and environmental effects

2) Software redundancy:
   - called program, modular, or functional redundancy
   - Software faults arise from specification and design errors or implementation (coding) mistakes
Software redundancy:

- Software errors cannot be detected by simple replication of identical software units
- Introduce diversity into the software replicas

Various views of redundant software:

(a)

(b)
Introduction to Software Fault Tolerance Techniques and Implementation

(c) HW\textsubscript{1}, HW\textsubscript{2}, ..., HW\textsubscript{n+1} connected to each SW\textsubscript{1}, SW\textsubscript{2}, ..., SW\textsubscript{n}.

(d) SW\textsubscript{1}, SW\textsubscript{2}, ..., SW\textsubscript{n} versus SW\textsubscript{1}, SW\textsubscript{2}, SW\textsubscript{n}.

HW = Hardware, SW = Software.
**Temporal redundancy:**

- repeating an execution using the same software and hardware resources involved in the initial, failed execution
- can overcome transient faults

**Advantage of temporal redundancy:**

- it does not require redundant hardware or software
- requires the availability of additional time to reexecute the failed process
Thanks for listening