Chapter 11 – Reliability Engineering

Topics covered

- ✧ Availability and reliability
- ✧ Reliability requirements
- ✧ Fault-tolerant architectures
- ✧ Programming for reliability
- ✧ Reliability measurement

Software reliability

- ✧ Dependable software is expected
- ✧ However, some system failures are accepted.
- ✧ Software systems have high reliability requirements
  - E.g., critical software systems
- ✧ Software engineering techniques for reliability requirements.
  - E.g., medical systems and aerospace systems

Faults, errors and failures

Term	Description
Human error or mistake	Human behavior that results in the introduction of faults into a system. For example, in the wilderness weather system, a programmer might decide that the way to compute the time for the next transmission is to add 1 hour to the current time. This works except when the transmission time is between 23.00 and midnight (midnight is 00.00 in the 24-hour clock).
System fault	A characteristic of a software system that can lead to a system error. The fault is the inclusion of the code to add 1 hour to the time of the last transmission, without a check if the time is greater than or equal to 23.00.
System error	An erroneous system state that can lead to system behavior that is unexpected by system users. The value of transmission time is set incorrectly (to 24.XX rather than 00.XX) when the faulty code is executed.
System failure	An event that occurs at some point in time when the system does not deliver a service as expected by its users. No weather data is transmitted because the time is invalid.

Faults and failures

- ✧ Failures
  - Results of system errors resulted from faults in the system
- ✧ However, faults do not necessarily result in system errors
  - Transient and ‘corrected’ before an error arises.
  - Never be executed.
- ✧ Errors do not necessarily lead to system failures
  - Corrected by detection and recovery
  - Protected by protection facilities.

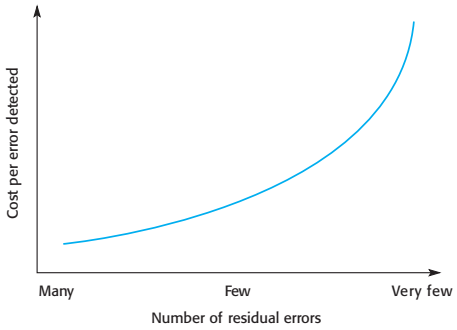
Fault management

- ✧ Fault avoidance
  - Avoid human errors to minimize system faults.
  - Organize development processes to detect and repair faults.
- ✧ Fault detection
  - Verification and validation techniques to remove faults.
- ✧ Fault tolerance
  - Design systems that faults do not cause failures.

Reliability achievement

- Fault avoidance
  - Development technique to minimise the possibility of mistakes or reveal mistakes.
- Fault detection and removal
  - Verification and validation techniques to increase the probability of correcting errors.
- Fault tolerance
  - Run-time techniques to ensure that faults do not cause errors.

The increasing costs of residual fault removal



Availability and reliability

Availability and reliability

- Reliability
  - The probability of failure-free system operation.
- Availability
  - The probability that a system conducts requested services at a point in time.
  - E.g., availability of 0.99.

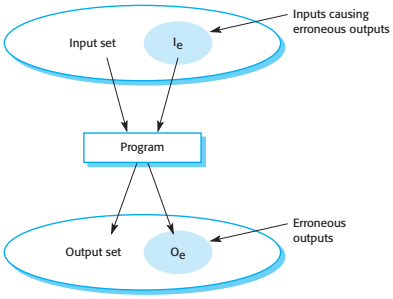
Reliability and specifications

- Reliability
  - Defined formally w.r.t. a system specification
  - A deviation from a specification.
- Incomplete or incorrect specifications
  - A system following specifications may ‘fail’.
- Unfamiliar with specifications
  - Unaware how the system is supposed to behave.
- Perceptions of reliability

Perceptions of reliability

- Not always reflect the user’s reliability perception
  - The assumptions about environments for a system are incorrect
    - Different usage of a system between in an office environment and in a university environment.
  - The consequences of system failures affects the perception of reliability.

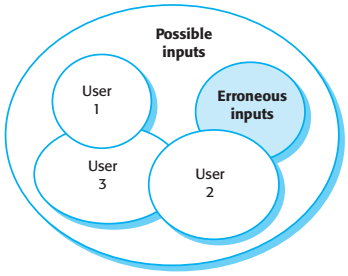
A system as an input/output mapping



Availability perception

- Expressed as a percentage of the time
  - Available to conduct services.
- Two factors not considered:
  - The number of users affected by unavailable systems.
  - The length of system failed or unavailable period.

Software usage patterns



Reliability in use

- Reliability not improved by X% by removing faults with X%
- Program defects rarely executed
  - Not encountered by users.
  - Not affect the perceived reliability.
- Users' operation patterns to avoid system features.
- Software systems with known faults
  - Considered reliable systems by users.

Reliability requirements

System reliability requirements

- Functional reliability requirements
  - Define system and software functions
  - Avoid, detect or tolerate faults
  - Not lead to system failure.
- Software reliability requirements
  - Cope with hardware failure or operator error.
- Non-functional reliability requirements
  - A measurable system attribute specified quantitatively.
  - E.g., the number of failures and the available time.

Reliability metrics

- Units of measurement of system reliability.
- Counting the number of operational failures and the period length that the system has been operational.
- Assess the reliability (e.g., critical systems)
  - Long-term measurement techniques
- Metrics
  - Probability of failure on demand
  - Rate of occurrence of failures

Probability of failure on demand (POFOD)

- The probability of the system failure when a service request is made.
  - Useful when demands for service are relatively infrequent.
- Implement appropriate protection systems
  - Demand services occasionally.
  - Serious consequence due to failed services.
- Develop for safety-critical systems
  - E.g., emergency shutdown system in a chemical plant.

Rate of fault occurrence (ROCOF)

- System failure occurrence rate
- ROCOF of 0.002
  - 2 failures are likely in each 1000 operational time units
- Reliable systems needed
  - Systems perform a number of similar requests in a short time
  - E.g., credit card processing system.
- Reciprocal of ROCOF is Mean time to Failure (MTTF)
  - Systems with long transactions
  - System processing takes a long time. MTTF should be longer than expected transaction length.

Availability

- The time that a software system is available
  - Repair and restart time considered
- Availability of 0.998
  - Software is available for 998 out of 1000 time units.
- Continuously running systems
  - E.g., railway signalling systems.

Availability specification

Availability	Explanation
0.9	The system is available for 90% of the time. This means that, in a 24-hour period (1,440 minutes), the system will be unavailable for 144 minutes. (1440 * 10%)
0.99	In a 24-hour period, the system is unavailable for 14.4 minutes.
0.999	The system is unavailable for 84 seconds in a 24-hour period.
0.9999	The system is unavailable for 8.4 seconds in a 24-hour period. Roughly, one minute per week.

Non-functional reliability requirements

- Non-functional reliability requirements
  - Reliability specifications using one of the reliability metrics
  - POFOD: probability of fault on demand
  - ROCOF: rate of occurrence of Fault
  - AVAIL: availability
- Used for many years in safety-critical systems
  - Uncommon for business critical systems.
- Need precise measurement about reliability and availability expectations.

Benefits of reliability specification

- Help to clarify stakeholders' needs.
- Provide a measurement basis for system tests.
- Improve the reliability by different design strategies.
- Evidence of including required reliability implementations.

Specifying reliability requirements

- Availability and reliability requirements for different types of failure.
  - Low probability of high-cost failures
- Availability and reliability requirements for different types of system service.
  - Tolerate failures in less critical services.
- A high level of reliability required.
  - Other mechanisms for reliable system services.

ATM reliability specification

- Key concerns
  - ATMs conduct services as requested
  - Record customer transactions
  - ATM systems are available when required.
- Database transaction mechanisms make a correction of transaction problems

ATM availability specification

- System services
  - The customer account database service;
  - 'withdraw cash', 'provide account information', etc.
- Specify a high level of availability in database service.
  - Database availability: 0.9999, between 7 am and 11pm.
  - A downtime of less than 1 minute per week.

ATM availability specification

- Key reliability issues depends on mechanical reliability.
- A lower level of software availability is acceptable.
- The overall availability
  - Specify availability with 0.999
  - A machine might be unavailable for between 1 and 2 minutes each day.

Insulin pump reliability specification

- Probability of failure (POFOD) metric.
- Transient failures
  - Repaired by user actions, such as, recalibration of the machine.
  - Low POFOD is acceptable. A failure occurs in every 500 demands.
- Permanent failures
  - Re-installed by the manufacturer.
  - Occur no more than once per year.
  - POFOD < 0.00002.

Functional reliability requirements

- ✧ Checking requirements
  - Identify incorrect data before it leads to a failure.
- ✧ Recovery requirements
  - Help the system recover from a failure.
- ✧ Redundancy requirements
  - Specify redundant system features.
- ✧ Process requirements
  - Specify software development processes.

Examples of functional reliability requirements

RR1: A pre-defined range for all operator inputs shall be defined and the system shall check that all operator inputs fall within this pre-defined range. (Checking)

RR2: Copies of the patient database shall be maintained on two separate servers that are not housed in the same building. (Recovery, redundancy)

RR3: N-version programming shall be used to implement the braking control system. (Redundancy)

RR4: The system must be implemented in a safe subset of Ada and checked using static analysis. (Process)

Fault-tolerant architectures

Fault tolerance

- ✧ Fault tolerant in critical situations.
- ✧ Fault tolerance required
  - High availability requirements
  - Failure costs are very high.
- ✧ Fault tolerance
  - Able to continue in operation despite software failures.
- ✧ Fault tolerant required against incorrect validation or specification errors, although a system is proved to conform to its specification

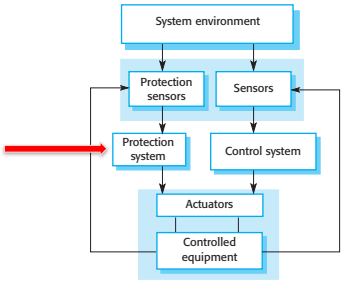
Fault-tolerant system architectures

- ✧ Fault-tolerant systems architectures
  - Fault tolerance is essential based on redundancy and diversity.
- ✧ Examples of situations for dependable architectures:
  - Flight control systems for safety of passengers
  - Reactor systems for a chemical or nuclear emergency
  - Telecommunication systems for 24/7 availability.

Protection systems

- ✧ A specialized system
  - Associated with other control system.
  - Take emergency action to deal with failures.
  - E.g., System to stop a train or system to shut down a reactor
- ✧ Monitor the controlled system and the environment.
- ✧ Take emergency action to shut down the system and avoid a catastrophe.

Protection system architecture



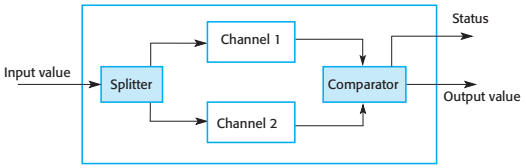
Protection system functionality

- ✧ Protection systems for redundancy
  - Control capabilities to replicate in the control software.
- ✧ Diverse protection systems
  - Different technology used in the control software.
- ✧ Need to expend in validation and dependability assurance.
- ✧ A low probability of failure for the protection system.

Self-monitoring architectures

- ✧ Multi-channel architectures
  - System monitoring its own operations
  - Take action if inconsistencies are discovered
- ✧ The same computation is carried out on each channel
  - Compare the results
  - Producing identical results assumes correct system operation
  - A failure exception is reported when different results arise.

Self-monitoring architecture



The first half of the chapter