CS5030

Dependability Properties

Learning objectives

- On completing this lecture and associated reading, you should
  - Be aware of the properties / dimensions of system dependability
  - Be aware of the processes and strategies for achieving these properties
  - Be aware of the metrics that can be used to measure some of these properties
  - Be aware of safety-critical systems

Dependability - recap

- Properties
  - Availability, reliability, safety, security and resilience
- Threats
  - Failures, errors and faults
- Means
  - Fault prevention / avoidance, tolerance, detection and removal

Software reliability

- Users generally expect all software to be dependable
- For non-critical applications, some failures may be acceptable
- However, critical systems have very high reliability requirements
  - Medical systems
  - Telecommunications and power systems
  - Aerospace systems
- Particular software engineering techniques for reliability may be needed in these cases

Faults, errors and failures

- Failures are usually a result of system errors that are derived from faults in the system

- However, faults do not necessarily result in system errors
  - The erroneous system state resulting from the fault may be transient and corrected before an error arises
  - The faulty code may never be executed

- Errors do not necessarily lead to system failures
  - The error can be corrected by built-in error detection and recovery
  - The failure can be protected against by built-in protection facilities

Achieving reliability

- Fault avoidance
  - Development techniques that either minimise the possibility of mistakes or trap mistakes before they result in the introduction of system faults
- Fault detection and removal
  - Verification and validation techniques that increase the probability of detecting and correcting errors before the system goes into service
- Fault tolerance
  - Run-time techniques to ensure that system faults do not result in system errors and/or that system errors do not lead to system failures

Reliability in practice

- Removing X% of the faults in a system will not necessarily improve the reliability by X%
- Program defects may be in rarely executed sections of the code so may never be encountered by users
  - Removing these does not affect the perceived reliability
- Users adapt their behaviour to avoid system features that may fail for them
- A program with known faults may therefore still be perceived as reliable by its users

Reliability metrics (1)

- Reliability metrics are units of measurement of system reliability

- System reliability is measured by
  - counting the number of operational failures and,
  - where appropriate, relating these to the demands made on the system and the time that the system has been operational

- A long-term measurement programme is required to assess the reliability of critical systems

Reliability metrics (2)

- Probability of failure on demand
- Rate of occurrence of failures / mean time to failure
- Availability

Probability of failure on demand (POFOD)

- Probability that the system will fail when a service request is made
  - Useful when demands for service are intermittent and relatively infrequent
- Appropriate for protection systems where services are demanded occasionally and where there are serious consequence if the service is not delivered
- Relevant for many safety-critical systems with exception management components
  - Emergency shutdown system in a chemical plant

Rate of occurrence of failures (ROCOF)

- Reflects the rate of occurrence of failure in the system
- ROCOF of 0.002 means 2 failures are likely in each 1000 operational time units
  - For e.g., 2 failures per 1000 hours of operation
- Relevant for systems where the system has to process a large number of similar requests in a short time
  - Credit card processing system, airline booking system
- Reciprocal of ROCOF is Mean time to Failure (MTTF)
  - Relevant for systems with long transactions - where system processing takes a long time (e.g. CAD systems)
  - MTTF should be longer than expected transaction length

Availability

- Measure of the fraction of the time that the system is available for use
- Takes repair and restart time into account
- Availability of 0.998 means software is available for 998 out of 1000 time units
- Relevant for non-stop, continuously running systems
  - For eg, telephone switching systems, railway signalling systems

Availability

- Availability is usually expressed as a percentage of the time that the system is available to deliver services e.g. 99.95%

- However, this does not take into account two factors:
  - Number of users affected by the service outage
    - Loss of service in the middle of the night is less important for many systems than loss of service during peak usage periods
  - Length of the outage
    - The longer the outage, the more the disruption
    - Several short outages are less likely to be disruptive than one long outage

Requirements

- Non-functional reliability requirements
  - Specifications of required reliability and availability using a metric
- Safety-critical systems have used quantitative reliability and availability specification for many years
  - Less common for business-critical systems
- More companies now demand 24/7 service from their systems
  - Need to be precise about their reliability and availability expectations

Fault tolerance

- In critical situations, software systems must be fault tolerant
  - where there are high availability requirements or
  - where system failure costs are very high
- Fault tolerance means that the system can continue in operation in spite of software failure
- Even if the system has been proved to conform to its specification, it must also be fault tolerant
  - There may be specification errors or the validation may be incorrect

Fault tolerant architectures

- Fault-tolerant architectures are used in situations where fault tolerance is essential
  - generally based on redundancy and diversity

- Examples of such situations:
  - Flight control systems, where system failure could threaten the safety of passengers
  - Reactor systems where failure of a control system could lead to a chemical or nuclear emergency
  - Telecommunication systems, where there is a need for 24/7 availability

Implementation strategies (1)

- Hardware fault tolerance
  - For eg, triple modular redundancy
- Architecture fault tolerance
  - Protection systems
  - Self-monitoring architectures

Implementation strategies (2)

- Software fault tolerance
  - N-version programming
  - Software diversity
    - Programming languages, teams, design methods, tools, algorithms
  - Good practices in dependable programming

Challenges in achieving fault tolerance

- Different teams trained in the same way may make the same errors
- Errors in specifications used by all the teams
- Cost implications of additional measures

Safety

- Property of a system that reflects the system’s ability to operate, normally or abnormally, without danger of causing human injury or death and without damage to the system’s environment

- Software safety is an important consideration
  - Most devices whose failure is critical now incorporate software-based control systems

Safety-critical systems (1)

- Systems where it is essential that system operation is always safe
  - The system should never cause damage to people or its environment even though there is potential for damage

- Examples
  - Control and monitoring systems in aircraft
  - Process control systems in chemical manufacture
  - Automobile control systems such as braking and engine management systems

Safety-critical systems (2)

- The system may be software-controlled so that decisions made by the software and subsequent actions are safety-critical
  - Software behaviour will be directly related to the overall safety of the system

- Software is extensively used for checking and monitoring other safety-critical components in a system
  - For example, all aircraft engine components are monitored by software looking for early indications of component failure
  - This software is safety-critical because, if it fails, other components may fail and cause an accident

Hazards

- Situations or events that can lead to an accident
  - Stuck valve in reactor control system
  - Incorrect computation by software in navigation system
  - Failure to detect possible allergy in medication prescribing system

- Hazards do not inevitably result in accidents
  - Accident prevention actions can be taken

Achieving safety

- Hazard avoidance
  - The system is designed so that some classes of hazard simply cannot arise

- Hazard detection and removal
  - The system is designed so that hazards are detected and removed before they result in an accident

- Damage limitation
  - The system includes protection features that minimise the damage that may result from an accident

Safety specifications

- The goal of safety requirements engineering is to identify protection requirements to ensure that system failures do not cause injury or death or environmental damage
- Safety requirements may be ‘shall not’ requirements
  - They define situations and events that should never occur
- Functional safety requirements define:
  - Checking and recovery features that should be included in a system
  - Features that provide protection against system failures and external attacks

Safety engineering processes

- Safety engineering processes are based on reliability engineering processes
  - Plan-based approach with reviews and checks at each stage in the process
  - General goal of fault avoidance and fault detection
  - Must also include safety reviews and explicit identification and tracking of hazards
- Regulators may require evidence that safety engineering processes have been used in system development

Safety and agility

- Agile methods are not usually used for safety-critical systems engineering
  - Extensive process and product documentation is needed for system regulation
    - Contradicts the agile focus on the software itself
  - A detailed safety analysis of a complete system specification is important
    - Contradicts the interleaved development of a system specification and program

- Some agile techniques such as test-driven development may be used

Formal methods for safety

- Formal methods can be used when a mathematical specification of the system is produced
- The ultimate static verification technique that may be used at different stages in the development process
- Arguments for and against their use

Security engineering

- Includes
  - the tools, techniques and methods
  - to support the development and maintenance of systems
  - that can resist malicious attacks intended to damage a computer-based system or its data

- A sub-field of the broader field of computer security

Security (1)

- A system property that reflects the system’s ability to protect itself from accidental or deliberate external attack

- Security is important
  - Most systems are networked so that external access to the system through the network is possible

- Security is an essential pre-requisite for availability, reliability and safety

Security (2)

- If a networked system is insecure then statements about its reliability and its safety may not hold

- These statements depend on the executing system and the developed system being the same
  - However, intrusion can change the executing system and/or its data
  - Therefore, the reliability and safety assurance may no longer be valid

Dimensions of security

- Confidentiality
  - Information in a system may be disclosed or made accessible to people or programs that are not authorised to have access to that information
- Integrity
  - Information in a system may be damaged or corrupted making it inconsistent or unreliable
- Availability
  - Access to a system or its data that is normally available may not be possible

Levels of security

- Infrastructure security
  - concerned with maintaining the security of all systems and networks that provide an infrastructure and a set of shared services to the organisation
- Application security
  - concerned with the security of individual application systems or related groups of systems
- Operational security
  - concerned with the secure operation and use of the organisation’s systems

Levels of security and focus

- Application security is a software engineering problem where the system is *designed* to resist attacks

- Infrastructure security is a systems management problem where the infrastructure is *configured* to resist attacks

- Operational security is primarily a human and social issue
  - Users sometimes take insecure actions for ease of use

Types of security threats

- Interception threats allow an attacker to gain access to an asset
- Interruption threats allow an attacker to make part of the system unavailable
- Modification threats allow an attacker to tamper with a system asset
- Fabrication threats allow an attacker to insert false information into a system

Security assurance (1)

- Vulnerability avoidance
  - The system is designed so that vulnerabilities do not occur
  - For example, if there is no external network connection then external attack is impossible

- Attack detection and elimination
  - The system is designed so that attacks on vulnerabilities are detected and neutralised before they result in an exposure
  - For example, virus checkers find and remove viruses before they infect a system

Security assurance (2)

- Exposure limitation and recovery
  - The system is designed so that the adverse consequences of a successful attack are minimised
  - For example, a backup policy allows damaged information to be restored

Security and dependability

- Security and reliability
  - Example problem: corrupted data
- Security and availability
  - Example problem: denial of service attack
- Security and safety
  - Example problem: corrupted code or data
- Security and resilience
  - Example problem: a cyberattack on a networked system

Security and organisations

- Security is expensive and it is important that security decisions are made in a cost-effective way
- Organisations use a risk-based approach to support security decision making
  - Should have a defined security policy based on security risk analysis
- Security risk analysis is a business rather than a technical process

Organisational security policies

- Security policies should set out information access strategies that should apply across the organisation
- The purpose of security policies is to inform everyone in an organisation about security
  - So these should not be long and detailed technical documents
- From a security engineering perspective, the security policy defines, in broad terms, the security goals of the organisation
- The security engineering process is concerned with implementing these goals

Security policies

- Organisations should have security policies on
  - the assets that must be protected
  - the level of protection that is required for different types of asset
  - the responsibilities of individual users, managers and the organisation
  - existing security procedures and technologies that should be maintained

Secure system design

- Security should be designed into a system
  - It is very difficult to make an insecure system secure after it has been designed and implemented
  - Should be accounted for during all stages of the development process

- Architectural design

- Good practices during development

Design compromises for security

- Adding security features to a system to enhance its security affects other attributes of the system

- Performance
  - Additional security checks slow down a system so its response time or throughput may be affected

- Usability
  - Security measures may require users to remember information or require additional interactions to complete a transaction
  - Can make the system less usable and frustrate system users

Architecture design for security

- Two fundamental issues have to be considered when designing an architecture for security
  - Protection
    - How should the system be organised so that critical assets can be protected against external attack?
  - Distribution
    - How should system assets be distributed so that the effects of a successful attack are minimised?

- These are potentially conflicting
  - If assets are distributed, then they are more expensive to protect
  - If assets are protected, then usability and performance requirements may be compromised

Protection levels

- Platform-level protection
  - Top-level controls on the platform on which a system runs
- Application-level protection
  - Specific protection mechanisms built into the application itself such as additional password protection
- Record-level protection
  - Protection that is invoked when access to specific information is requested
- These lead to a layered protection architecture

Design guidelines for security

- Security decisions should be based on an explicit policy
- Avoid single point of failure
- Fail securely
- Consider balance between security and usability
- Reduce risks with redundancy and diversity
- Design for deployment and recoverability
- Compartmentalise assets
- …

Key points

- Software reliability can be achieved by fault avoidance, detection and removal and tolerance
- Reliability requirements can be defined quantitatively in the system requirements specification
- Reliability metrics include probability of failure on demand, rate of occurrence of failure and availability
- Redundancy and diversity can be used in different levels to achieve fault tolerance

Key points

- Safety-critical systems are systems whose failure can lead to human injury or death
- A hazard-driven approach is used to understand the safety requirements for safety-critical systems
- It is important to have a well-defined, certified process for safety-critical systems development
  - This should include the identification and monitoring of potential hazards
- Static analysis and formal methods can be used identify potential errors

Key points

- Security engineering is concerned with how to develop systems that can resist malicious attacks
- Security threats can be threats to confidentiality, integrity or availability of a system and/or its data
- Key issues in designing a secure systems architecture include organising the system structure to protect key assets and distributing the system assets to minimise the losses from a successful attack
- Security validation is difficult because security requirements state what should not happen in a system, rather than what should