



ILLINOIS INSTITUTE
OF TECHNOLOGY

Transforming Lives. Inventing the Future.

www.iit.edu

SOFTWARE ENGINEERING

CS 487

Prof. Dennis Hood
Computer Science



Week 6

Dependability and Reliability

Lesson Overview

- Dependability and Reliability
- Reading
 - Ch. 10 – Dependable Systems
 - Ch. 11 – Reliability Engineering
- Objectives
 - Understand the concepts of dependability and reliability in the context of systems
 - Examine approaches for developing better systems, where better means dependable and reliable
 - Discuss assessment approaches for measuring these factors

Dependability Considerations

- Repairability
 - The ability to recover from failure
 - Diagnosis, analysis, “surgical” repair, etc.
- Maintainability
 - Economical adaptation to new requirements
- Survivability
 - The ability to withstand “attack”
 - Recognize, resist, and recover
- Error tolerance
 - Avoid or at least tolerate user errors
 - Autocorrect if possible
 - Teach the user along the way

Specification

- Types of specification
 - Risk-driven – avoid hazards
 - Reliability – measurable performance standards
 - Security – authorization and protection
- Formal specification
 - Human communication is complex and error prone
 - Formality seeks to simplify and reduce the opportunity for error
 - Unfortunately formality has had limited effectiveness in practice to date

Risk Management

- Assessment
 - Identify assets requiring protection and value
 - Identify threats and likelihood of occurrence
 - Assess exposure (likelihood X impact)
 - Consider mitigation possibilities and costs
 - Mitigate where feasible
- Life-cycle risk assessment
 - Secondary assessment following system and data architecture decisions

Failure Categories

- Hardware failure
 - Design errors
 - Component failure
- Software failure
 - Requirements issues
 - Design errors
 - Coding defects
- Operational failure
 - User misuse

Safety Critical Systems

- Primary safety-critical systems
 - Embedded system controllers
 - Failure of the controller leads to failure of the system it is controlling
- Secondary safety-critical systems
 - Failure of this system will not directly cause harm
 - However such a failure could lead to harmful situations (e.g., CAD or CASE tools)

The Need for Security

- Openness has many benefits
 - Data sharing
 - Remote user access, etc.
- But also introduces vulnerabilities
 - Unauthorized access
 - Denial of service
 - Exposure of sensitive data, etc.
- Security engineering attempts to develop systems that minimize the exposure
 - Application security is a software engineering problem
 - Infrastructure security is a systems engineering problem

Security Management

- Access
 - User and permission management
 - Restrict users' access
- Deployment
 - Control installation and configuration
 - Patching
- Attacks
 - Monitoring, detection and recovery

Security Concepts

- Asset – system resource that must be protected
- Exposure – potential loss/harm
- Vulnerability – exploitable weakness
- Attack – exploitation of vulnerability
- Threats – circumstances under which attacks can occur
- Control – a protective measure that reduces vulnerability

Design for Security

- Architectural design
 - Protect critical assets
 - Distribute assets to minimize the effects of an attack
 - Analogous to a medieval castle
- Design guidelines
 - Establish and adhere to policies
 - Minimize impact through distribution, redundancy, compartmentalization, etc.
 - Recoverability, failing securely, safe deployment
 - Maintain usability
 - Validate inputs

Design for Deployment

- Include support for viewing and analyzing configurations
- Minimize default privileges to only those that are essential
- Localize configuration settings
- Make it easy to fix vulnerabilities

Dependable Programming

- Control the visibility of information
- Check all inputs for validity
- Handle all exceptions
- Avoid error-prone code constructs
- Provide recovery and restart capabilities
- Check array bounds
- Include timeouts when interfacing with external components
- Name all constants that represent real-world values

Risky Programming

- Unconditional branching (go-to's)
- Floating point numbers
- Pointers
- Dynamic memory allocation
- Parallelism
- Recursion
- Interrupts
- Inheritance
- Aliasing
- Unbounded arrays
- Default input processing

Survivability

- Design to minimize vulnerabilities and their effects,
- but just in case, design to withstand attacks
- This is critical for critical systems
- Multiple strategies
 - Resistance to attacks
 - Recognition of the type of attack
 - Maintain adequate operation during an attack and then recover to full operation