THOMSON
COURSE TECHNOLOGY

Ethics in Information Technology, Second Edition

Chapter 7
Software Development

Objectives

- Why do companies require high-quality software in business systems, industrial process control systems, and consumer products?
- What ethical issues do software manufacturers face in making tradeoffs between project schedules, project costs, and software quality?

Objectives (continued)

- What are the four most common types of software product liability claims, and what actions must plaintiffs and defendants take to be successful?
- What are the essential components of a software development methodology, and what are its benefits?

Objectives (continued)

- How can Capability Maturity Model Integration improve an organization's software development process?
- What is a safety-critical system, and what actions are required during its development?

Strategies to Engineer Quality Software

- High-quality software systems
  - Operate safely and dependably
  - Have a high degree of availability
  - Required to support the fields of
    - Air traffic control
    - Nuclear power
    - Automobile safety
    - Health care
    - Military and defense
    - Space exploration

Strategies to Engineer Quality Software (continued)

- More and more users are demanding high quality software
- Software defect
  - Could cause a system to fail to meet users' needs
  - Impact may be trivial or very serious
  - Patches may contain defects
- Software quality
  - Degree to which software meets the needs of users

Strategies to Engineer Quality Software (continued)

- Quality management
  - How to define, measure, and refine the quality of the development process and products
  - Objective
    - Help developers deliver high-quality systems that meet the needs of users
- Deliverables
  - Products such as:
    - Statements of requirements
    - Flowcharts
    - User documentation

Strategies to Engineer Quality Software (continued)

- Primary cause for poor software quality
  - Developers do not know how to design quality into software
  - Or do not take the time to do so
- Developers must
  - Define and follow a set of rigorous engineering principles
  - Learn from past mistakes
  - Understand the environment in which systems operate
  - Design systems relatively immune to human error

Strategies to Engineer Quality Software (continued)

- Programmers make mistakes in turning design specifications into code
  - About one defect for every 10 lines of code
- Pressure to reduce time-to-market
- First release
  - Organizations avoid buying the first release
  - Or prohibit its use in critical systems
  - Usually has many defects

The Importance of Software Quality

- Business information systems are a set of interrelated components
  - Including
    - Hardware
    - Software
    - Databases
    - Networks
    - People
    - Procedures

The Importance of Software Quality (continued)

- Business information system examples
  - Order-processing system
  - Electronic-funds transfer system
  - Airline's online ticket reservation system
- Decision support system (DSS)
  - Used to improve decision making
- Software for industrial use
- Software controls the operation of many industrial and consumer products

The Importance of Software Quality (continued)

- Mismanaged software can be fatal to a business
- Ethical questions
  - How much effort and money to invest to ensure high-quality software
  - Whether products could cause damage
    - Legal exposure if they did

Legal Overview: Software Product Liability

- Product liability
  - Liability of manufacturers, sellers, lessors, and others for injuries caused by defective products
  - There is no federal product liability law
    - Mainly state law
    - Article 2 of the Uniform Commercial Code
- Strict liability
  - Defendant held responsible for the injury
  - Regardless of negligence or intent

Legal Overview: Software Product Liability (continued)

- Strict liability
  - Plaintiff must prove only that the software product is defective or unreasonably dangerous and that the defect caused the injury
  - No requirement to prove that the manufacturer was careless or negligent
    - Or to prove who caused the defect
  - All parties in the chain of distribution are liable

Legal Overview: Software Product Liability (continued)

- Legal defenses used against strict liability
  - Doctrine of supervening event
  - Government contractor defense
  - Expired statute of limitations
- Negligence
  - A supplier is not held responsible for every product defect that causes a customer or third-party loss
  - Responsibility is limited to defects that could have been detected and corrected through “reasonable” software development practices

Legal Overview: Software Product Liability (continued)

- Negligence
  - Area of great risk for software manufacturers
  - Defense of negligence may include
    - Legal justification for the alleged misconduct
    - Demonstrate that the plaintiffs’ own actions contributed to injuries

Legal Overview: Software Product Liability (continued)

- Warranty
  - Assures buyers or lessees that a product meets certain standards of quality
  - Expressly stated
  - Implied by law
- Breach of warranty claim
  - Plaintiff must have a valid contract that the supplier did not fulfill
  - Can be extremely difficult to prove
    - Because the software supplier writes the warranty

Legal Overview: Software Product Liability (continued)

- Intentional misrepresentation
  - Seller or lessor either misrepresents the quality of a product
  - Or conceals a defect in it
  - Forms of representation
    - Advertising
    - Salespersons' comments
    - Invoices
    - Shipping labels

Software Development Process

- Large software project roles
  - System analysts
  - Programmers
  - Architects
  - Database specialists
  - Project managers
  - Documentation specialists
  - Trainers
  - Testers

Software Development Process (continued)

- Software development methodology
  - Work process
  - Controlled and orderly progress
  - Defines activities and individual and group responsibilities
  - Recommends specific techniques for accomplishing various activities
  - Offers guidelines for managing the quality of software during various stages of development

Software Development Process (continued)

- Safer and cheaper to avoid software problems at the beginning than to attempt to fix damages after the fact
  - Identify and remove errors early in the development process
    - Cost-saving measure
    - Most efficient way to improve software quality

Software Development Process (continued)

- Effective methodology
  - Reduces the number of software errors that might occur
  - If an organization follows widely accepted development methods, negligence on its part is harder to prove
- Software quality assurance (QA) refers to methods within the development cycle
  - Guarantee reliable operation of product
  - Ideally applied at each stage throughout the development cycle

Software Development Process (continued)

- Dynamic testing
  - Black-box testing
    - Tester has no knowledge of code
  - White-box testing
    - Testing all possible logic paths through the software unit
    - With thorough knowledge of the logic
    - Make each program statement execute at least once

Software Development Process (continued)

- Static testing
  - Static analyzers are run against the new code
  - Looks for suspicious patterns in programs that might indicate a defect
- Integration testing
  - After successful unit testing
  - Software units are combined into an integrated subsystem
  - Ensures that all linkages among various subsystems work successfully

Software Development Process (continued)

- System testing
  - After successful integration testing
  - Various subsystems are combined
  - Tests the entire system as a complete entity
- User acceptance testing
  - Independent testing
  - Performed by trained end users
  - Ensures that the system operates as they expect

Capability Maturity Model Integration for Software

- Process improvement approach
- Defined by the Software Engineering Institute
  - At Carnegie Mellon University in Pittsburgh
- Defines essential elements of effective processes
- General enough to evaluate and improve almost any process
- Frequently used to assess software development practices

Capability Maturity Model Integration for Software (continued)

- Defines five levels of software development maturity
- Identifies issues most critical to software quality and process improvement
- Organization conducts an assessment of its software development practices
  - Determines where they fit in the capability model
  - Identifies areas for improvement
    - Action plans are needed to upgrade the development process

Capability Maturity Model Integration for Software (continued)

- Maturity level increases
  - Organization improves its ability to deliver good software on time and on budget

CMMI Maturity Levels

TABLE 7-1 CMMI maturity levels

Maturity level	Definition	Percentage of organizations at this level (as of August 2005)
Initial	Process unpredictable, poorly controlled, and reactive	5%
Managed	Process characterized for projects and is often reactive	36%
Defined	Process characterized for the organization and is proactive	29%
Quantitatively managed	Process measured and controlled	5%
Optimizing	Focus is on continuous process improvement	25%

Source: Capability Maturity Model Integration (CMMI) Overview, Carnegie Mellon University, www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview05.pdf, October 29, 2005.

Key Issues in Software Development

- Consequences of software defects in certain systems can be deadly
  - Companies must take special precautions

Development of Safety-Critical Systems

- Safety-critical system
  - Failure may cause injury or death
  - Examples
    - Automobile's antilock brakes
    - Nuclear power plant reactors
    - Airplane navigation
    - Roller coasters
    - Elevators
    - Medical devices

Development of Safety-Critical Systems (continued)

- Key assumption
  - Safety will *not* automatically result from following the organization's standard development methodology
- Must go through a more rigorous and time-consuming development process than other kinds of software
- All tasks require
  - Additional steps
  - More thorough documentation
  - More checking and rechecking

Development of Safety-Critical Systems (continued)

- Project safety engineer
  - Explicit responsibility for the system's safety
  - Uses a logging and monitoring system
    - To track hazards from the project's start to finish
- Hazard log
  - Used at each stage of the software development process
  - Assesses how it has accounted for detected hazards

Development of Safety-Critical Systems (continued)

- Safety reviews
  - Held throughout the development process
- Robust configuration management system
  - Tracks all safety-related documentation
- Formal documentation required
  - Including verification reviews and signatures
- Key issue
  - Deciding when QA staff has performed enough testing

Development of Safety-Critical Systems (continued)

- Risk
  - Probability of an undesirable event occurring times the magnitude of the event's consequences if it does happen
  - Consequences include
    - Damage to property
    - Loss of money
    - Injury to people
    - Death

Development of Safety-Critical Systems (continued)

- Redundancy
  - Provision of multiple interchangeable components to perform a single function
  - In order to cope with failures and errors
- N-version programming
  - Form of redundancy
  - Involves the execution of a series of program instructions simultaneously by two different systems
  - Uses different algorithms to execute instructions that accomplish the same result

Development of Safety-Critical Systems (continued)

- N-version programming
  - Results from the two systems are compared
  - If a difference is found, another algorithm is executed to determine which system yielded the correct result
  - Instructions for the two systems are:
    - Written by programmers from two different companies
    - Run on different hardware devices
  - Both systems are highly unlikely to fail at the same time under the same conditions

Development of Safety-Critical Systems (continued)

- Decide what level of risk is acceptable
  - Controversial
  - If the level of risk in a design is judged to be too great, make system modifications
- Mitigate the consequences of failure
  - By devising emergency procedures and evacuation plans
- Recall product
  - When data indicates a problem

Development of Safety-Critical Systems (continued)

- Reliability
  - Probability of a component or system performing without failure over its product life
- Human interface
  - Important and difficult area of safety-critical system design
  - Leave the operator little room for erroneous judgment

Quality Management Standards

- ISO 9000 standard
  - Guide to quality products, services, and management
  - Organization must submit to an examination by an external assessor
  - Requirements:
    - Written procedures for everything it does
    - Follow those procedures
    - Prove to the auditor the organization fulfilled the first two requirements

Quality Management Standards (continued)

- Failure mode and effects analysis (FMEA)
  - Used to evaluate reliability
  - Determine the effect of system and equipment failures
  - Goal:
    - Identify potential design and process failures early in a project

Quality Management Standards (continued)

- Failure mode and effects analysis (FMEA)
  - Failure mode
    - Describes how a product or process could fail
  - Effect
    - Adverse consequence that a customer might experience
  - Seldom is a one-to-one relationship between cause and effect

Quality Management Standards

- DO-178B/EUROCCAE ED-128
  - Evaluation standard for the international aviation community
  - Developed by Radio Technical Commission for Aeronautics (RTCA)

Manager's Checklist for Improving Software Quality

TABLE 7-2 Manager's checklist for improving software quality

Questions	Yes	No
Has senior management made a commitment to quality software?	___	___
Have you used CMMI to evaluate your organization's software development process?	___	___
Have you adopted a standard software development methodology?	___	___
Does the methodology place a heavy emphasis on quality management and address how to define, measure, and refine the quality of the software development process and its products?	___	___
Are software project managers and team members trained in the use of this methodology?	___	___
Are software project managers and team members held accountable for following this methodology?	___	___
Is a strong effort made to identify and remove errors as early as possible in the software development process?	___	___
In the testing of software, are both static and dynamic testing used?	___	___
Are white-box testing and black-box testing used?	___	___
Has an honest assessment been made to determine if the software being developed is safety-critical?	___	___
If the software is safety-critical, are additional tools and methods employed, and do they include the following: project safety engineer, hazard logs, safety reviews, formal configuration management systems, rigorous documentation, risk analysis processes, and the FMEA technique?	___	___

Summary

- More and more users are demanding high quality software
- Software product liability claims are frequently based on
  - Strict liability
  - Negligence
  - Breach of warranty
  - Misrepresentation

Summary (continued)

- Software development methodology
  - Defines activities in the software development process
  - Defines individual and group responsibilities
  - Recommends specific techniques
  - Offers guidelines for managing the quality of products
- CMMI
  - Defines five levels of software development maturity
- Safety-critical system
  - Failure may cause injury or death