Ethics in Information Technology, Fourth Edition

Chapter 7

Software Development

Objectives

- As you read this chapter, consider the following questions:
  - Why do companies require high-quality software in business systems, industrial process control systems, and consumer products?
  - What potential ethical issues do software manufacturers face in making trade-offs between project schedules, project costs, and software quality?
  - What are the four most common types of software product liability claims?

Objectives (cont'd.)

- What are the essential components of a software development methodology, and what are the benefits of using such a methodology?
- How can the Capability Maturity Model Integration improve an organization's software development process?
- What is a safety-critical system, and what special actions are required during its development?

Strategies for Engineering Quality Software

- High-quality software systems:
  - Perform quickly and efficiently
  - Operate safely and reliably
  - Meet their users' needs
  - Are required to support the fields of:
    - Air traffic control
    - Nuclear power
    - Automobile safety
    - Health care
    - Military and defense
    - Space exploration

Strategies for Engineering Quality Software (cont'd.)

- Increased demand for high-quality software
- Software defect
  - Could cause a system to fail to meet users' needs
  - Impact may be trivial or very serious
  - Subtle and undetectable or glaringly obvious
- Software quality
  - Degree to which software meets the needs of users

Strategies for Engineering Quality Software (cont'd.)

- Quality management
  - Defines, measures, and refines the quality of the development process and products developed
  - Objective
    - Help developers deliver high-quality systems that meet the needs of users
- Deliverables are products such as:
  - Statements of requirements
  - Flowcharts
  - User documentation

Strategies for Engineering Quality Software (cont'd.)

- Primary cause for poor software quality:
  - Many developers do not know how to design quality into software from the start
  - Or do not take the time to do so
- Developers must:
  - Define and follow rigorous engineering principles
  - Learn from past mistakes
  - Understand systems' operating environment
  - Design systems relatively immune to human error

Strategies for Engineering Quality Software (cont'd.)

- Programmers make mistakes in turning design specifications into code
  - About one defect for every 7-10 lines of code
- Extreme pressure to reduce time to market
  - Driven by need to:
    - Deliver new functionality
    - Begin generating revenue to recover costs
    - Meet quarterly earnings forecasts
  - Resources and time to ensure quality are often cut

Strategies for Engineering Quality Software (cont'd.)

- Ethical dilemma: how much additional cost and effort should be expended to ensure products and services meet customers' expectations?
- First release of software
  - Organizations avoid buying the first release
  - Or prohibit its use in critical systems
  - Usually has many defects
- Established software products can also falter:
  - When operating conditions change

The Importance of Software Quality

- Business information systems
  - Set of interrelated components including:
    - Hardware
    - Software
    - Databases
    - Networks
    - People
    - Procedures
  - Collect and process data and disseminate the output

The Importance of Software Quality (cont'd.)

- Business information system examples
  - Manufacturer's order-processing system
  - Bank's electronic-funds transfer system
  - Airline's online ticket reservation system
- Decision support system (DSS)
  - Used to improve decision making
- Software is used to control industrial processes
- Software controls the operation of many industrial and consumer products

The Importance of Software Quality (cont'd.)

- 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 and what the legal exposure would be if they did

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

Software Product Liability (cont'd.)

- 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
    - Manufacturer
    - Subcontractors
    - Distributors

Software Product Liability (cont'd.)

- Legal defenses used against strict liability
  - Doctrine of supervening event
  - Government contractor defense
  - Expired statute of limitations
- Negligence
  - Failure to do what a reasonable person would do, or doing something that a reasonable person would not do
  - Responsibility is limited to defects that could have been detected and corrected through "reasonable" software development practices

Software Product Liability (cont'd.)

- Negligence
  - Area of great risk for software manufacturers
  - Defense of negligence may include:
    - Legal justification for the alleged misconduct
    - Demonstration that the plaintiffs' own actions contributed to injuries (contributory negligence)

Software Product Liability (cont'd.)

- Warranty
  - Assures buyers or lessees that a product meets certain standards of quality
  - May be expressly stated or implied by law
- Breach of warranty claim
  - When the product fails to meet the terms of its warranty
  - 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 to limit liability

Software Product Liability (cont'd.)

- 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 (cont'd.)

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

Software Development Process (cont'd.)

- Easier and cheaper to avoid software problems at the beginning than to attempt to fix damages after the fact
  - Cost to identify and remove a defect in an early stage can be up to 100 times less than removing a defect in distributed software
  - Identify and remove errors early in the development process
    - Cost-saving measure
    - Most efficient way to improve software quality

Software Development Process (cont'd.)

- Effective methodology protects from legal liability
  - Reduces the number of software errors
  - 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
  - Are applied at each stage in the development cycle
  - Include testing before the product ships

Software Development Process (cont'd.)

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

Software Development Process (cont'd.)

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

Software Development Process (cont'd.)

- System testing
  - Occurs 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

- 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 (cont'd.)

- 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 defined to upgrade the development process

Capability Maturity Model Integration (cont'd.)

- Maturity level increases
  - Organization improves its ability to deliver good software on time and on budget
- CMMI-Development
  - Set of guidelines for 22 process areas related to systems development
  - Organizations that do these 22 things well will have an outstanding software development and maintenance process

Capability Maturity Model Integration (cont'd.)

TABLE 7-1 Definition of CMMI maturity levels

Maturity level	Description
Initial	Process is ad hoc and chaotic; organization tends to over commit and processes are often abandoned during times of crisis
Managed	Projects employ processes and skilled people; status of work products is visible to management at defined points
Defined	Processes are well defined and understood and are described in standards, procedures, tools, and methods; processes are consistent across the organization
Quantitatively managed	Quantitative objectives for quality and process performance are established and are used as criteria in managing projects; specific measures of process performance are collected and statistically analyzed
Optimizing	Organization continually improves its processes based on a quantitative understanding of its business objectives and performance needs

Source Line: Used with permission from Carnegie Mellon University.

Key Issues in Software Development

- Consequences of software defects in certain systems can be deadly
  - Companies must take special precautions
- Ethical decisions involve a trade-off between quality and cost, ease of use, and time to market

Development of Safety-Critical Systems

- Safety-critical system
  - A system whose 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 (cont'd.)

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

Development of Safety-Critical Systems (cont'd.)

- 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 to assess how project team has accounted for detected hazards

Development of Safety-Critical Systems (cont'd.)

- 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 issues
  - Ethical dilemmas re: increased time and expense
  - Deciding when QA staff has performed enough testing

Development of Safety-Critical Systems (cont'd.)

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

Development of Safety-Critical Systems (cont'd.)

- Redundancy
  - Provision of multiple interchangeable components to perform a single function
  - Used to cope with failures and errors
  - During times of widespread disaster, lack of sufficient redundant can lead to major problems

Development of Safety-Critical Systems (cont'd.)

- 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 (cont'd.)

- N-version programming (cont'd.)
  - 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 can be:
    - Written by programmers from two different companies
    - Run on different hardware devices
  - Rationale
    - Both systems are highly unlikely to fail at the same time under the same conditions

Development of Safety-Critical Systems (cont'd.)

- Decide what level of risk is acceptable
  - Difficult and controversial decision
  - Make system modifications if level of risk is judged to be too great
- Mitigate the consequences of failure
  - Devise emergency procedures and evacuation plans
- Decide whether to recall a product:
  - When data indicates a problem

Development of Safety-Critical Systems (cont'd.)

- 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
  - Should leave the operator little room for erroneous judgment
  - Poor design of a system interface can greatly increase risk

Quality Management Standards

- ISO 9001 family of standards
  - 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 (cont'd.)

- Failure mode and effects analysis (FMEA)
  - Technique used to evaluate reliability and determine the effect of system and equipment failures
  - Failures are classified by:
    - Impact on a project's success
    - Personnel safety
    - Equipment safety
    - Customer satisfaction and safety
  - Goal
    - Identify potential design and process failures early in a project

TABLE 7-4 Manager's checklist for improving software quality

Question	Yes	No
Has senior management made a commitment to develop quality software?		
Have you used CMMI to evaluate your organization's software development process?		
Has your company 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?		
Are both static and dynamic software testing methods used?		
Are white-box testing and black-box testing methods 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: a project safety engineer, hazard logs, safety reviews, formal configuration management systems, rigorous documentation, risk analysis processes, and the FMEA technique?		

Source Line: Course Technology/Cengage Learning.

Summary

- Demand for high-quality software is increasing
- Developers are under extreme pressure to reduce time to market of products
- Software product liability claims are frequently based on:
  - Strict liability
  - Negligence
  - Breach of warranty
  - Misrepresentation

Summary (cont'd.)

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

Summary (cont'd.)

- ISO 9001 standard is a guide to quality products, services, and management
- Failure mode and effects analysis (FMEA) is an important technique used to develop ISO 9001-compliant quality systems