Information Technology Engineering

Mohammad Hossein Manshaei

manshaei@gmail.com

1401

Module A.5

Ethical Decision in Software Development

Reference:

Ethics in Information Technology

6th Edition

George W. Reynolds

Chapter 7: Ethical Decision in 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?

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

High 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

Software Quality: Defect and Patches

- 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: Case Studies

- In April 2003, **Sallie Mae** (the largest U.S. student loan company) a miscalculation of the monthly payments on their loans (ultimately, over 1 million loans were affected by the error)

- In September 2004, **The Los Angeles air traffic control system** shut down for part of a day in September 2004, because of a software glitch in the control system

Software Quality: Case Studies

- In December 2005, a critical software bug was detected that affected **39 different Symantec software products**. The bug could have been exploited to load unauthorized code onto a computer and to potentially take control of the computer

- **Windows Vista** became the subject of much criticism for issues relating to performance, security, and privacy. Microsoft began allowing PC manufacturers to offer a "**downgrade**" option to buyers who purchased machines with Vista but who wanted to switch to Windows XP.

Software Quality: Case Studies

- The Nest thermostat is a clever device that enables users to monitor and adjust their thermostats using their smartphones. However, during a recent cold spell, **a software glitch caused the devices to shut down or go offline for many customers**. Temperatures in their homes plunged over night, threatening to freeze pipes and causing potentially serious health issues for the elderly and ill.

- Flaws in the software used to **analyze fMRI data** were recently uncovered by scientists who studied the results of many different brain studies over the last 15 years. According to the new report, the software flaw frequently caused false positives, suggesting brain activity where there was none. This has raised considerable controversy between critics who have long said fMRI is nothing more than high-tech pseudo medicine and brain-imaging researchers who claim that the software problems

Software Quality: Case Studies

- **Skype suffered a massive outage** in August 2007. According to a Skype spokesperson, a "previously unseen software bug within the network resource algorithm" caused the problem

- Software problems in **the automated baggage sorting system at Heathrow airport** caused the system to go offline for almost two days in February 2008. 6,000 passengers experienced delays, flight cancellations, and frustration. The breakdown reportedly occurred during a software upgrade, despite pretesting of the software.

Software Quality: Definition and Management

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

Strategies for Engineering Quality Software

- **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

Software Quality: First Release Issues

- Programmers make mistakes in turning design specifications into code
- Pressure to **reduce time-to-market**
- When forced to choose between adding more user features and doing more testing, most software companies decide in favor of more features.

- **First release**
  - Organizations avoid buying the first release
  - Or prohibit its use in critical systems
  - Usually has many defects

Mistakes in Defining User Requirements and Turning them into Codes

- **Coverity** (a software testing firm) performed a sophisticated scan of **17.5 million lines of code** from the widely used open source packages Linux, Apache HTTP, MySQL, and Perl/PHP/Python, which are used to run millions of web servers around the world.
  - The study found just **0.290 defects per thousand lines** of code.
- Separate analysis found that about **10 to 20 defects per 1000 lines** of code are identified during **in-house testing of Microsoft's applications**.
- By the time those applications are released to the public, that error rate is in the range of **0.5 defects per 1000 lines of codes.**
- That means the **Microsoft Windows 10** operating system (which contains an estimated 50 million lines of code) may have included close to **25,000 defects** at the time it was released.
- Thus, critical software used daily by workers worldwide likely contains tens of thousands of defects.

Case Study: Microsoft

- The **Microsoft Vista** operating system took over five years to develop and contains more than **50 million lines of code**.
- Assume that the Microsoft software developers produced code at the accuracy rate mentioned above.
- Even if 99.9 percent of the defects were identified and fixed before the product was released to the public, there would still be **about one bug per 10,000 lines of code, or roughly 5,000 bugs in Windows Vista.**

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

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: A Few Examples

1. **Business information system examples**
   - Order-processing system
   - Electronic-funds transfer system
   - Airline's online ticket reservation system
2. **Decision support system (DSS) used to improve decision making**
   - Forecast customer demand
   - Recommend stocks
   - Schedule shift workers
3. **Software for industrial use**
   - Steel manufacturers
4. **Software controls the operation of many industrial and consumer products**

The Importance of Software Quality

- 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

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

Legal Overview:
Software 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

Legal Overview: Software Product Liability

- **Strict liability**
  - **Defendant** held responsible for the injury Regardless of negligence or intent
  - **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

- **Legal defenses used against strict liability**
  1. Doctrine of supervening event
  2. Government contractor defense
  3. Expired statute of limitations

Legal Overview: Software Product Liability

- **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
  - 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

- **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 to attempt to limit their liability

Warranty: Standard to be Met

- The goods must be fit for the ordinary purpose for which they are used.
- The goods must be adequately contained, packaged, and labeled.
- The goods must be of an even kind, quality, and quantity within each unit.
- The goods must conform to any promise or affirmation of fact made on the container or label.
- The quality of the goods must pass without objection in the trade.
- The goods must meet a fair average or middle range of quality.

Legal Overview: Software Product Liability

- **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

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

Software Development Methodologies

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

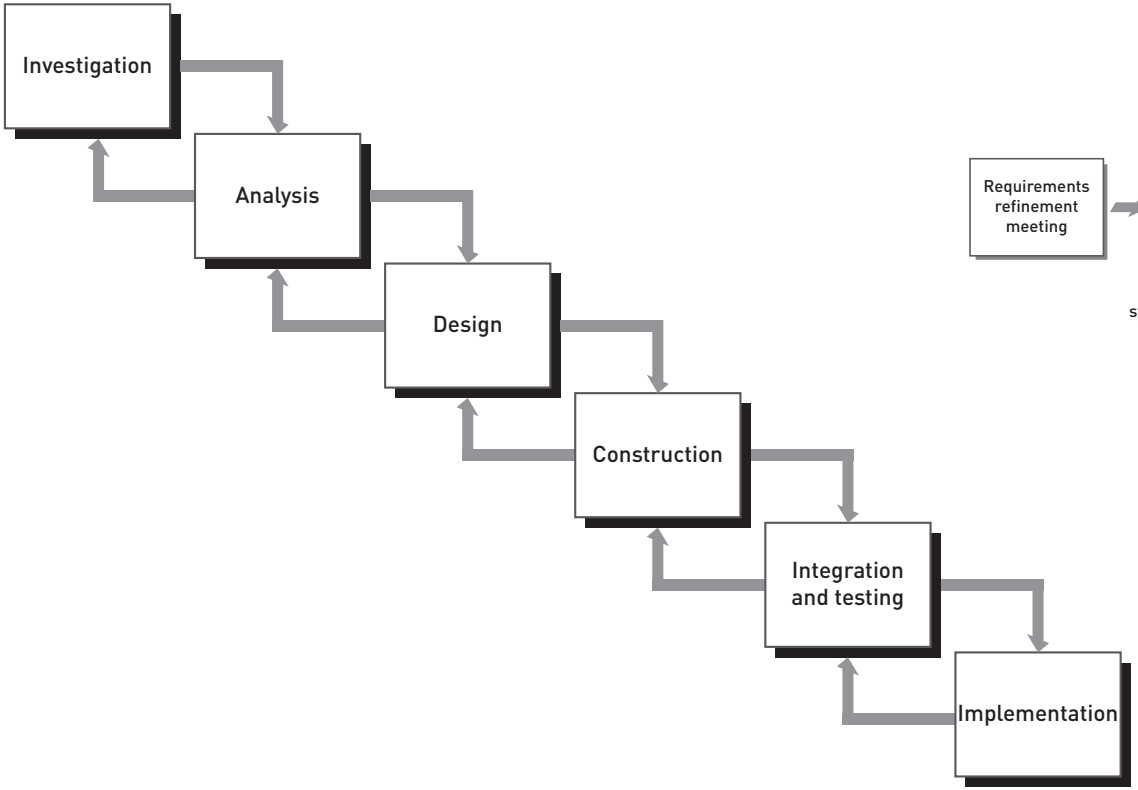
Software Development Methodology

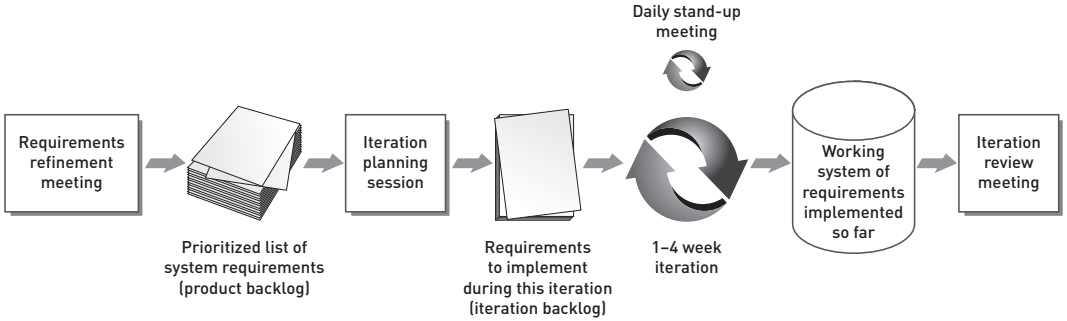
1. Work process
2. Controlled and orderly progress
3. Defines activities and individual and group responsibilities
4. Recommends specific techniques for accomplishing various activities
5. Offers guidelines for managing the quality of software during various stages of development

Software Development Process

- It is **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

Waterfall vs Agile System Development Models

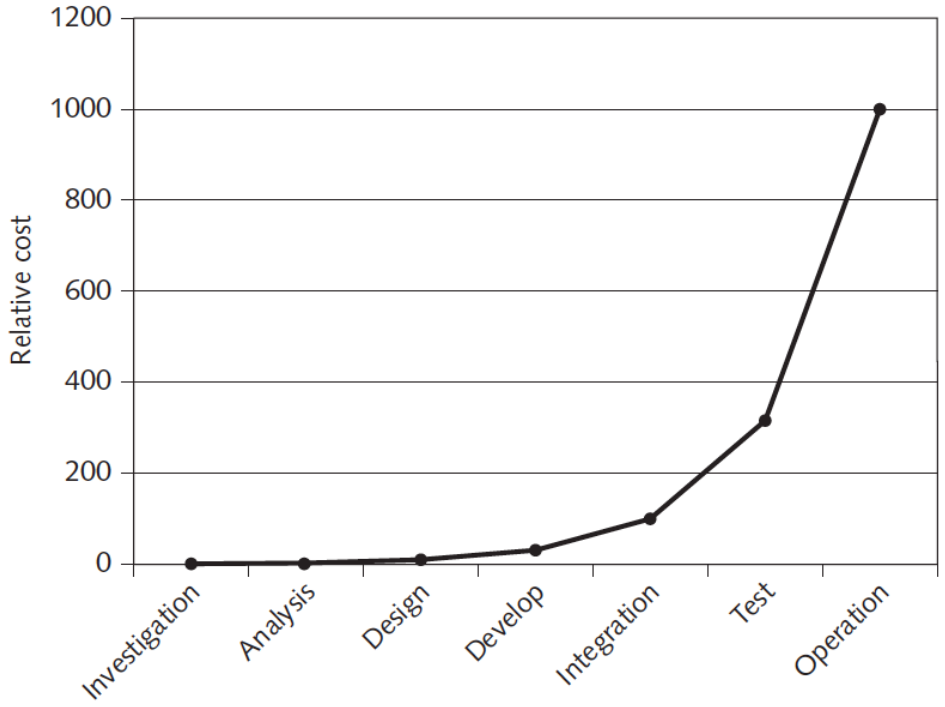




Waterfall vs Agile System Development Models

Waterfall		Agile	
Pros	**Cons**	**Pros**	**Cons**
Formal review at end of each stage allows maximum management control.	Often, users' needs go unstated or are miscommunicated or misunderstood. Users may end up with a system that meets those needs as understood by the developers; however, this might not be what the users really needed.	For appropriate projects, this approach puts an application into production sooner.	It is an intense process that takes considerable time and effort on the part of project members and can result in burnout for system developers and other project participants.
Structured processes produce many intermediate products that can be used to measure progress toward developing the system.	Users can't easily review intermediate products and evaluate whether a particular product will lead to a system that meets their business requirements.	Forces teamwork and lots of interaction between users and project stakeholders so that users are more likely to get a system that meets their needs.	Requires stakeholders and users to spend more time working together on the project.

Cost to Identify and Remove a Defect



Software Development Process

- **Effective methodology**
  1. Reduces the number of software errors that might occur
  2. 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**

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

Software Testing

- **Dynamic Software Testing:** Entering test data and comparing the results to the expected results
  - **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 Testing

- **Static testing**
  - Static analyzers are run against the new code
  - Looks for suspicious patterns in programs that might indicate a defect
- **Unit Testing**
  - Involves testing individual components of code (subroutines, modules, and programs) to verify that each unit performs as intended.
- **Integration testing**
  - After successful unit testing
  - Software units are combined into an integrated subsystem
  - Ensures that all linkages among various subsystems work successfully

Software Testing

- **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

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

Capability Maturity Model Integration (CMMI) for Software

- Process improvement approach: a set of guidelines for 22 process areas related to system development and maintenance.
- 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
- **CMMI-Development** used to assess and improve software development practices

Capability Maturity Model Integration (CMMI) for Software

- **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

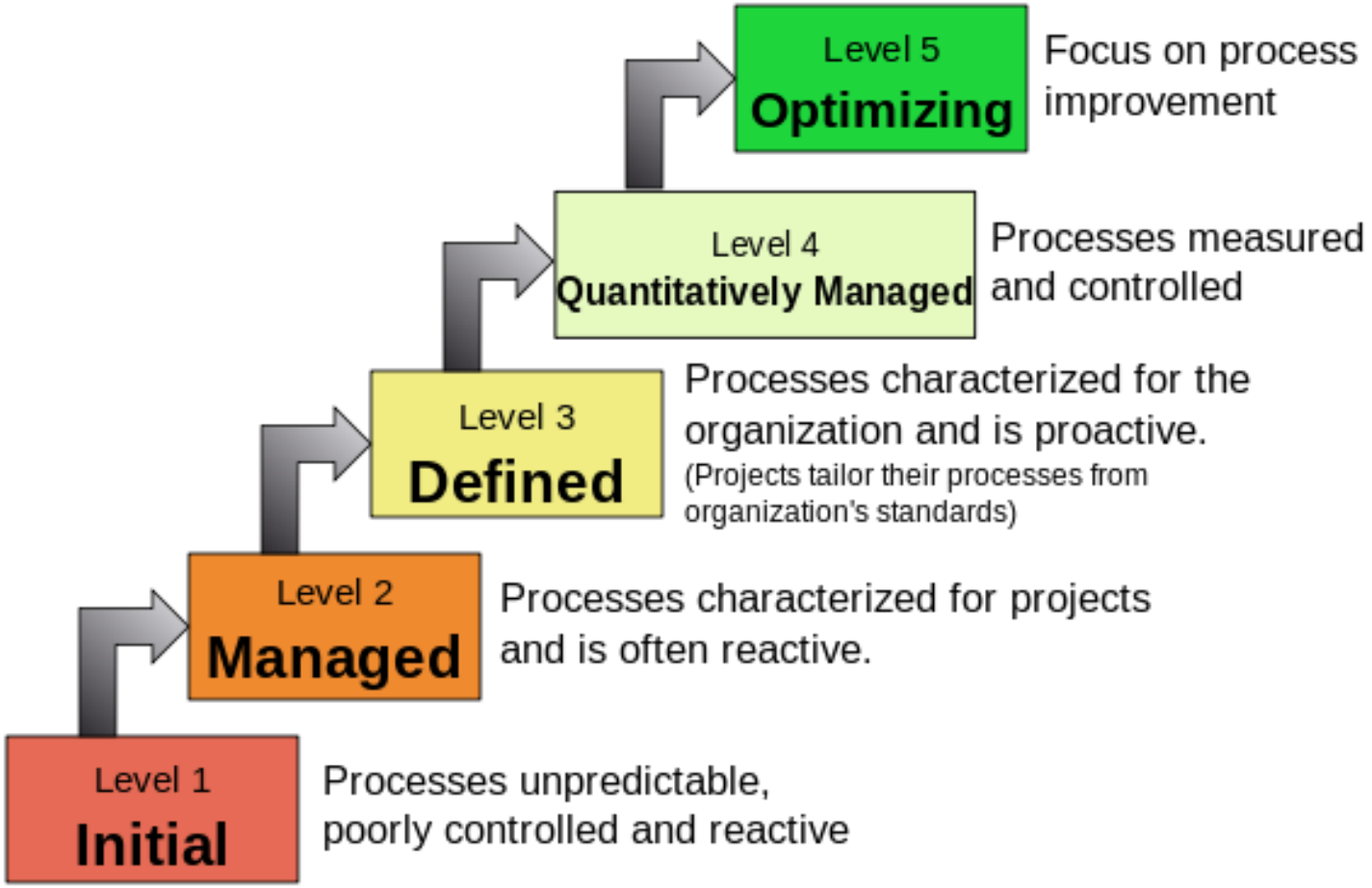
Capability Maturity Model Integration (CMMI) for Software

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

Maturity level	Description
Initial	Process is ad hoc and chaotic; organization tends to overcommit 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; changes are based on a quantitative understanding of its business objectives and performance needs.

Source: Used with permission from Carnegie Mellon University.

Characteristics of the Maturity levels



CMMI Maturity Levels

Maturity level	Percent of 5,159 organizations surveyed
Not provided	3.3%
Initial	0.8%
Managed	22.1%
Defined	66.8%
Quantitatively managed	1.6%
Optimizing	6.5%

CMMI Results in 11 Organizations

- A 33 percent decrease in the cost to fix defects
- A reduction in the number of defects found, from 6.6 to 2.1 per thousand lines of code
- A 30 percent increase in productivity
- An increase in project-schedule milestones met, from 50 percent to 95 percent

Partial List of Organizations using CMMI

Accenture	Intel
Boeing	Lockheed Martin
DynCorp	Reuters
Federal Aviation Administration	Samsung
General Dynamics	Tata Consultancy
Honeywell	U.S. Army, Air Force, and Navy
IBM Global Services	Wipro

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

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

- **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

- **Appoint System 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

- **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

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

Risk Management

- **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

Risk Management

- **Risk:** Quantified by three elements:
  1. A risk event
  2. The probability of the event happening
  3. The impact (positive or negative) on the business outcome if the risk does actually occur.
- The **annualized rate of occurrence (ARO)** is an estimate of the probability that this event will occur over the course of a year.
- The **single loss expectancy (SLE)** is the estimated loss that would be incurred if the event happens.
- The **annualized loss expectancy (ALE)** is the estimated loss from this risk over the course of a year.
- The annual loss expectancy: **ARO X SLE = ALE**

Risk Management

- **Risk management:** The process of identifying, monitoring, and limiting risks to a level that an organization is willing to accept.
- **Residual Risk:** The level of risk that remains after managing risk.
- Strategies for addressing a particular risk include the following:
  1. Acceptance
  2. Avoidance
  3. Mitigation
  4. Redundancy
  5. Transference

1. Acceptance

- When the cost of avoiding a risk outweighs the potential losss of a risk, an organization will likely accept the risk.

- A decision to accept a risk can be extremely difficult and controversial when dealing with safety-critical systems because making that determination involves forming personal judgments about:
  1. The value of human life
  2. Assessing potential liability in case of an accident
  3. Evaluating the potential impact on the surrounding natural environment
  4. Estimating the system's costs and benefits

2. Avoidance

- Eliminate the vulnerability that gives rise to a particular risk in order to avoid the risk altogether
- The most effective solution, but often not possible due to organizational requirements and factors beyond an organization's control.

3. Mitigation

- The reduction in either the likelihood or the impact of the occurrence of a risk
- **N-version programming** is an approach to minimizing the impact of software errors by independently implementing the same set of user requirements N times

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

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

4. Redundancy

- Provision of multiple interchangeable components to perform a single function
- In order to cope with failures and errors

5. Transference

- A common way to accomplish risk transference is for an individual or an organization to purchase insurance, such as auto or business liability insurance.
- Another way to transfer risk is to outsource the risk by contracting with a third party to manage the risk.

Development of Safety-Critical Systems

- 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

- **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

Contents

- ➢ Software Quality and Why It is Important
  - ◇ Case Studies
  - ◇ The Importance of Software Quality
  - ◇ Software Product Liability
- ➢ Strategies for Developing Quality Software
  - ◇ Software Development Methodologies
  - ◇ Software Testing
  - ◇ Capability Maturity Model Integration (CMMI)
  - ◇ Development of Safety-Critical Systems
  - ◇ Risk Management
  - ◇ Quality Management Standards

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

ISO 9000 Series of Standards

- ISO 9001: Design, development, production, installation, servicing
- ISO 9002: Production, installation, servicing
- ISO 9003: Inspection and testing
- ISO 9000-3: Development, supply, and maintenance of software
- ISO 9004: Quality management and quality systems elements

Quality Management Standards

- **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**

Issue	Severity	Occurrence	Criticality	Detection	Risk priority
#1	3	4	12	9	108
#2	9	4	36	2	72
#3	4	5	20	4	80

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

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?		

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