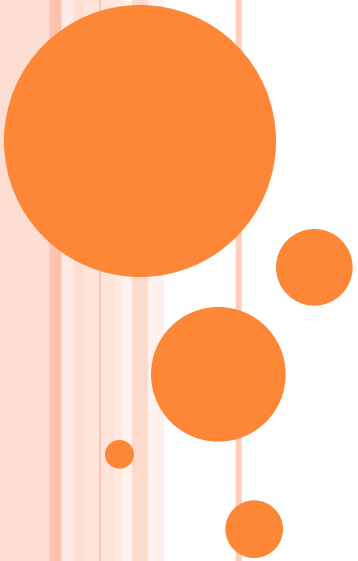
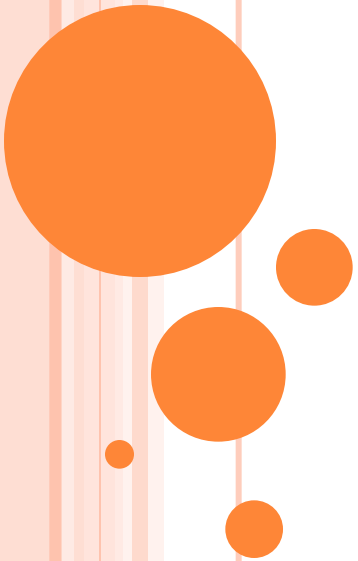
# Software Engineering

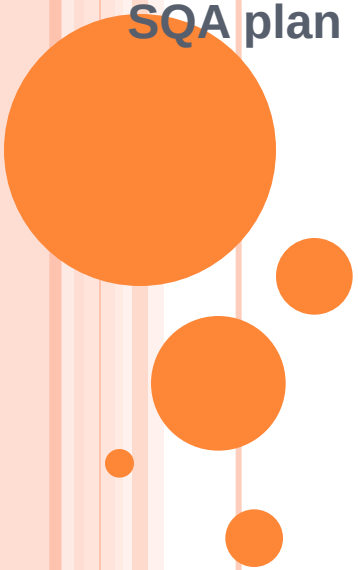


# Quality Management



# Topics Covered

- Quality concepts
- Software quality assurance
- Software reviews
- Statistical software quality assurance
- Software reliability, availability, and safety
- SQA plan



# What is Quality Management

- Also called software quality assurance (SQA)
- Serves as an umbrella activity that is applied throughout the software process
- Involves doing the software development correctly versus doing it over again
- Reduces the amount of rework, which results in lower costs and improved time to market
- Encompasses
  - A software quality assurance process
  - Specific quality assurance and quality control tasks (including formal technical reviews and a multi-tiered testing strategy)
  - Effective software engineering practices (methods and tools)
  - Control of all software work products and the changes made to them
  - A procedure to ensure compliance with software development standards
  - Measurement and reporting mechanisms

# Quality Defined

- Defined as a characteristic or attribute of something
- Refers to measurable characteristics that we can compare to known standards
- In software it involves such measures as cyclomatic complexity, cohesion, coupling, function points, and source lines of code
- Includes variation control
  - A software development organization should strive to minimize the variation between the predicted and the actual values for cost, schedule, and resources
  - They should make sure their testing program covers a known percentage of the software from one release to another
  - One goal is to ensure that the variance in the number of bugs is also minimized from one release to another

# Quality Defined (continued)

- Two kinds of quality are sought out
  - Quality of design
    - The characteristic that designers specify for an item
    - This encompasses requirements, specifications, and the design of the system
  - Quality of conformance (i.e., implementation)
    - The degree to which the design specifications are followed during manufacturing
    - This focuses on how well the implementation follows the design and how well the resulting system meets its requirements
- Quality also can be looked at in terms of user satisfaction

  User satisfaction = compliant product
  + good quality
  + delivery within budget and schedule

# Quality Control

- Involves a series of inspections, reviews, and tests used throughout the software process
- Ensures that each work product meets the requirements placed on it
- Includes a feedback loop to the process that created the work product
  - This is essential in minimizing the errors produced
- Combines measurement and feedback in order to adjust the process when product specifications are not met
- Requires all work products to have defined, measurable specifications to which practitioners may compare to the output of each process

# Quality Assurance Functions

- Consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control activities
- Provides management personnel with data that provides insight into the quality of the products
- Alerts management personnel to quality problems so that they can apply the necessary resources to resolve quality issues

# The Cost of Quality

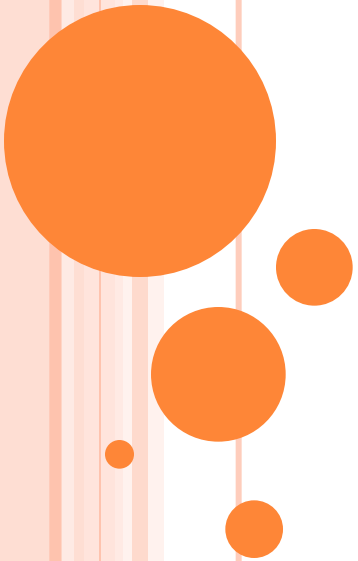
- Includes all costs incurred in the pursuit of quality or in performing quality-related activities
- Is studied to
  - Provide a baseline for the current cost of quality
  - Identify opportunities for reducing the cost of quality
  - Provide a normalized basis of comparison (which is usually dollars)
- Involves various kinds of quality costs (See next slide)
- Increases dramatically as the activities progress from
  - Prevention → Detection → Internal failure → External failure

"It takes less time to do a thing right than to explain why you did it wrong." Longfellow

# Kinds of Quality Costs

- Prevention costs
  - Quality planning, formal technical reviews, test equipment, training
- Appraisal costs
  - Inspections, equipment calibration and maintenance, testing
- Failure costs – subdivided into internal failure costs and external failure costs
  - Internal failure costs
    - Incurred when an error is detected in a product prior to shipment
    - Include rework, repair, and failure mode analysis
  - External failure costs
    - Involves defects found after the product has been shipped
    - Include complaint resolution, product return and replacement, help line support, and warranty work

# Software Quality Assurance



# Software Quality Defined

Definition: "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software"



(More on next slide)

# Software Quality Defined (continued)

- This definition emphasizes three points
  - Software requirements are the foundation from which quality is measured; lack of conformance to requirements is lack of quality
  - Specified standards define a set of development criteria that guide the manner in which software is engineered; if the criteria are not followed, lack of quality will almost surely result
  - A set of implicit requirements often goes unmentioned; if software fails to meet implicit requirements, software quality is suspect
- Software quality is no longer the sole responsibility of the programmer
  - It extends to software engineers, project managers, customers, salespeople, and the SQA group
  - Software engineers apply solid technical methods and measures, conduct formal technical reviews, and perform well-planned software testing

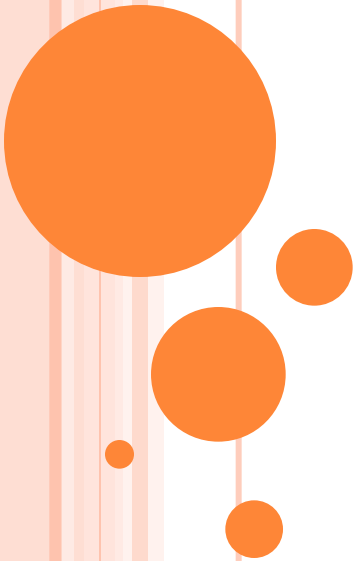
# The SQA Group

- Serves as the customer's in-house representative
- Assists the software team in achieving a high-quality product
- Views the software from the customer's point of view
  - Does the software adequately meet quality factors?
  - Has software development been conducted according to pre-established standards?
  - Have technical disciplines properly performed their roles as part of the SQA activity?
- Performs a set of of activities that address quality assurance planning, oversight, record keeping, analysis, and reporting (See next slide)

# SQA Activities

- Prepares an SQA plan for a project
- Participates in the development of the project's software process description
- Reviews software engineering activities to verify compliance with the defined software process
- Audits designated software work products to verify compliance with those defined as part of the  software process
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure
- Records any noncompliance and reports to senior management
- Coordinates the control and management of change
- Helps to collect and analyze software metrics

# Software Reviews



# Purpose of Reviews

- Serve as a filter for the software process
- Are applied at various points during the software process
- Uncover errors that can then be removed
- Purify the software analysis, design, coding, and testing activities
- Catch large classes of errors that escape the originator more than other practitioners
- Include the formal technical review (also called a walkthrough or inspection)
  - Acts as the most effective SQA filter
  - Conducted by software engineers for software engineers
  - Effectively uncovers errors and improves software quality
  - Has been shown to be up to 75% effective in uncovering design flaws (which constitute 50-65% of all errors in software)
- Require the software engineers to expend time and effort, and the organization to cover the costs

# Formal Technical Review (FTR)

- Objectives
  - To uncover errors in function, logic, or implementation for any representation of the software
  - To verify that the software under review meets its requirements
  - To ensure that the software has been represented according to predefined standards
  - To achieve software that is developed in a uniform manner
  - To make projects more manageable
- Serves as a training ground for junior software engineers to observe different approaches to software analysis, design, and construction
- Promotes backup and continuity because a number of people become familiar with other parts of the software
- May sometimes be a sample-driven review
  - Project managers must quantify those work products that are the primary targets for formal technical reviews
  - The sample of products that are reviewed must be representative of the products as a whole

# The FTR Meeting

- Has the following constraints
  - From 3-5 people should be involved
  - Advance preparation (i.e., reading) should occur for each participant but should require no more than two hours a piece and involve only a small subset of components
  - The duration of the meeting should be less than two hours
- Focuses on a specific work product (a software requirements specification, a detailed design, a source code listing)
- Activities before the meeting
  - The producer informs the project manager that a work product is complete and ready for review
  - The project manager contacts a review leader, who evaluates the product for readiness, generates copies of product materials, and distributes them to the reviewers for advance preparation
  - Each reviewer spends one to two hours reviewing the product and making notes before the actual review meeting
  - The review leader establishes an agenda for the review meeting and schedules the time and location



(More on next slide)

# The FTR Meeting (continued)

- Activities during the meeting
  - The meeting is attended by the review leader, all reviewers, and the producer
  - One of the reviewers also serves as the recorder for all issues and decisions concerning the product
  - After a brief introduction by the review leader, the producer proceeds to "walk through" the work product while reviewers ask questions and raise issues
  - The recorder notes any valid problems or errors that are discovered; no time or effort is spent in this meeting to solve any of these problems or errors
- Activities at the conclusion of the meeting
  - All attendees must decide whether to
    - Accept the product without further modification
    - Reject the product due to severe errors (After these errors are corrected, another review will then occur)
    - Accept the product provisionally (Minor errors need to be corrected but no additional review is required)
  - All attendees then complete a sign-off in which they indicate that they took part in the review and that they concur with the findings



(Move on to next slide)

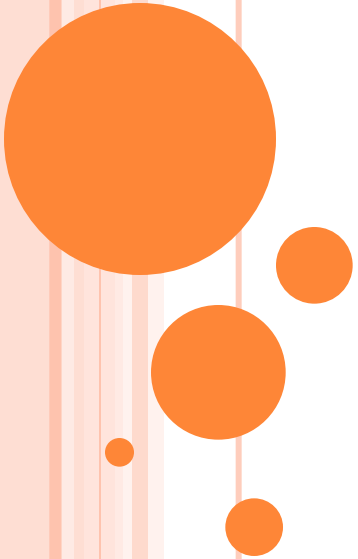
# The FTR Meeting (continued)

- Activities following the meeting
  - The recorder produces a list of review issues that
    - Identifies problem areas within the product
    - Serves as an action item checklist to guide the producer in making corrections
  - The recorder includes the list in an FTR summary report
    - This one to two-page report describes what was reviewed, who reviewed it, and what were the findings and conclusions
  - The review leader follows up on the findings to ensure that the producer makes the requested corrections

# FTR Guidelines

1) Review the product, not the producer
2) Set an agenda and maintain it
3) Limit debate and rebuttal; conduct in-depth discussions off-line
4) Enunciate problem areas, but don't attempt to solve the problem noted
5) Take written notes; utilize a wall board to capture comments
6) Limit the number of participants and insist upon advance preparation
7) Develop a checklist for each product in order to structure and focus the review
8) Allocate resources and schedule time for FTRs
9) Conduct meaningful training for all reviewers
10) Review your earlier reviews to improve the overall review process

# Statistical Software Quality Assurance



# Process Steps

1) Collect and categorize information (i.e., causes) about software defects that occur
2) Attempt to trace each defect to its underlying cause (e.g., nonconformance to specifications, design error, violation of standards, poor communication with the customer)
3) Using the Pareto principle (80% of defects can be traced to 20% of all causes), isolate the 20%

# A Sample of Possible Causes for Defects

- Incomplete or erroneous specifications
- Misinterpretation of customer communication
- Intentional deviation from specifications
- Violation of programming standards
- Errors in data representation
- Inconsistent component interface
- Errors in design logic
- Incomplete or erroneous testing
- Inaccurate or incomplete documentation
- Errors in programming language translation of design
- Ambiguous or inconsistent human/computer interface

# Six Sigma

- Popularized by Motorola in the 1980s
- Is the most widely used strategy for statistical quality assurance
- Uses data and statistical analysis to measure and improve a company's operational performance
- Identifies and eliminates defects in manufacturing and service-related processes
- The "Six Sigma" refers to six standard deviations (3.4 defects per a million occurrences)



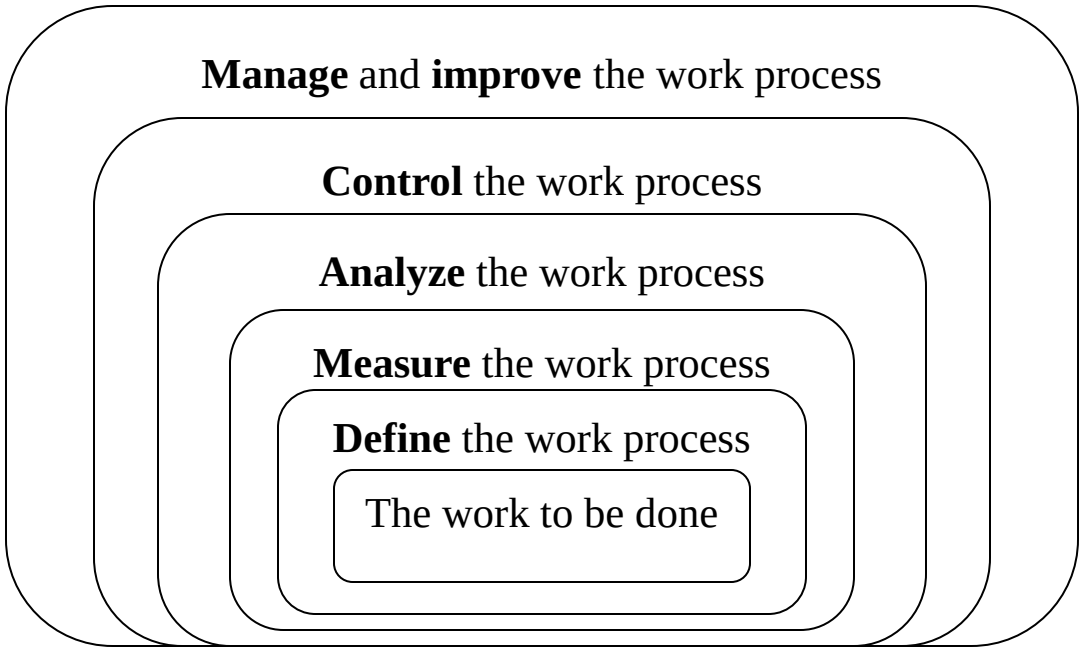
(More on next slide)

# Six Sigma (continued)

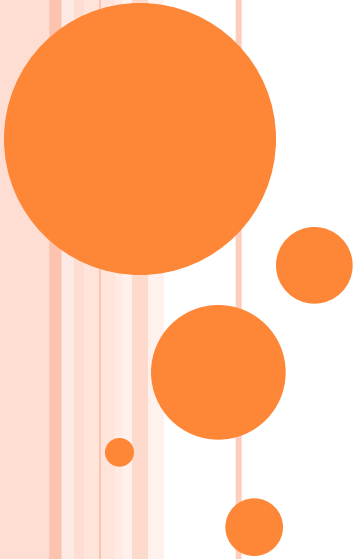
- Three core steps
  - Define customer requirements, deliverables, and project goals via well-defined methods of customer communication
  - Measure the existing process and its output to determine current quality performance (collect defect metrics)
  - Analyze defect metrics and determine the vital few causes (the 20%)
- Two additional steps are added for existing processes (and can be done in parallel)
  - Improve the process by eliminating the root causes of defects
  - Control the process to ensure that future work does not reintroduce the causes of defects

# Six Sigma (continued)

- All of these steps need to be performed so that you can manage the process to accomplish something
- You cannot effectively manage and improve a process until you first do these steps (in this order):



# Software Reliability, Availability, and Safety



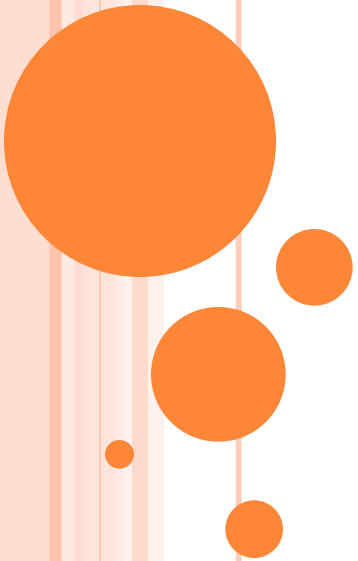
# Reliability and Availability

- Software failure
  - Defined: Nonconformance to software requirements
  - Given a set of valid requirements, all software failures can be traced to design or implementation problems (i.e., nothing wears out like it does in hardware)
- Software reliability
  - Defined: The probability of failure-free operation of a software application in a specified environment for a specified time
  - Estimated using historical and development data
  - A simple measure is `MTBF = MTTF + MTTR = Uptime + Downtime`
  - Example:
    - `MTBF = 68 days + 3 days = 71 days`
    - `Failures per 100 days = (1/71) * 100 = 1.4`
- Software availability
  - Defined: The probability that a software application is operating according to requirements at a given point in time
  - `Availability = [MTTF/ (MTTF + MTTR)] * 100%`
  - Example:
    - `Avail. = [68 days / (68 days + 3 days)] * 100 % =  96%`

# Software Safety

- Focuses on identification and assessment of potential hazards to software operation
- It differs from software reliability
  - Software reliability uses statistical analysis to determine the likelihood that a software failure will occur; however, the failure may not necessarily result in a hazard or mishap
  - Software safety examines the ways in which failures result in conditions that can lead to a hazard or mishap; it identifies faults that may lead to failures
- Software failures are evaluated in the context of an entire computer-based system and its environment through the process of fault tree analysis or hazard analysis

# SQA Plan



# Purpose and Layout

- Provides a road map for instituting software quality assurance in an organization
- Developed by the SQA group to serve as a template for SQA activities that are instituted for each software project in an organization
- Structured as follows:
  - The purpose and scope of the plan
  - A description of all software engineering work products that fall within the purview of SQA
  - All applicable standards and practices that are applied during the software process
  - SQA actions and tasks (including reviews and audits) and their placement throughout the software process
  - The tools and methods that support SQA actions and tasks
  - Methods for assembling, safeguarding, and maintaining all SQA-related records
  - Organizational roles and responsibilities relative to product quality