Quality overview

Quality—A Philosophical View

Robert Persig [74] commented on the thing we call *quality:*

Quality . . . you know what it is, yet you don't know what it is. But that's self-contradictory. But some things are better than others, that is, they have more quality. But when you try to say what the quality is, apart from the things that have it, it all goes poof! There's nothing to talk about. But if you can't say what Quality is, how do you know what it is, or how do you know that it even exists? If no one knows what it is, then for all practical purposes it doesn't exist at all. But for all practical purposes it really does exist. What else are the grades based on? Why else would people pay fortunes for some things and throw others in the trash pile? Obviously some things are better than others . . . but what's the betterness? . . . So round and round you go, spinning mental wheels and nowhere finding anyplace to get traction. What the hell is Quality? What is it?

Quality—A Pragmatic View

The *transcendental view argues (like Persig) that quality* is something that you immediately recognize, but cannot explicitly define.

The *user view sees quality in terms of an end-user's* specific goals. If a product meets those goals, it exhibits quality.

The *manufacturer's view defines quality in terms of the* original specification of the product. If the product conforms to the spec, it exhibits quality.

The *product view suggests that quality can be tied to* inherent characteristics (e.g., functions and features) of a product.

Finally, the *value-based view measures quality based on* how much a customer is willing to pay for a product. In reality, quality encompasses all of these views and more.

Software Quality

Software quality can be defined as:

*An **effective software process** applied in a manner that creates a **useful product** that provides measurable **value** for those who produce it and those who use it.*

- J Bessin, "The Business Value of Quality" (IBM DeveloperWorks)

Effective Software Process

- An *effective software process* establishes the infrastructure that supports any effort at building a high quality software product.
- The management aspects of process create the checks and balances that help avoid project chaos—a key contributor to poor quality.
- Software engineering practices allow the developer to analyze the problem and design a solid solution—both critical to building high quality software.
- Finally, umbrella activities such as change management and technical reviews have as much to do with quality as any other part of software engineering practice.

Useful Product

- A *useful product* delivers the content, functions, and features that the end-user desires
- But as important, it delivers these assets in a reliable, error free way.
- A useful product always satisfies those requirements that have been explicitly stated by stakeholders.
- In addition, it satisfies a set of implicit requirements (e.g., ease of use) that are expected of all high quality software.

Adding Value

- By *adding value for both the producer and user* of a software product, high quality software provides benefits for the software organization and the end-user community.
- The software organization gains added value because high quality software requires less maintenance effort, fewer bug fixes, and reduced customer support.
- The user community gains added value because the application provides a useful capability in a way that expedites some business process.
- The end result is:
  - (1) greater software product revenue,
  - (2) better profitability when an application supports a business process, and/or
  - (3) improved availability of information that is crucial for the business.

Quality Dimensions

as per David Garvin[87]

Performance Quality. Does the software deliver all content, functions, and features that are specified as part of the requirements model in a way that provides value to the end-user?

Feature quality. Does the software provide features that surprise and delight first-time end-users?

Reliability. Does the software deliver all features and capability without failure? Is it available when it is needed? Does it deliver functionality that is error free?

Conformance. Does the software conform to local and external software standards that are relevant to the application? Does it conform to de facto design and coding conventions? For example, does the user interface conform to accepted design rules for menu selection or data input?

Quality Dimensions (contd…)

as per David Garvin[87]

Durability. Can the software be maintained (changed) or corrected (debugged) without the inadvertent generation of unintended side effects? Will changes cause the error rate or reliability to degrade with time?

Serviceability. Can the software be maintained (changed) or corrected (debugged) in an acceptably short time period. Can support staff acquire all information they need to make changes or correct defects?

Aesthetics. Most of us would agree that an aesthetic entity has a certain elegance, a unique flow, and an obvious "presence" that are hard to quantify but evident nonetheless.

Perception. In some situations, you have a set of prejudices that will influence your perception of quality.

Cost of Quality

Prevention costs:

Quality planning
Formal technical reviews
Test equipment
Training

Appraisal Costs:

Technical reviews
Data Collection
Testing & Debugging

Internal failure costs:

Rework
Repair
failure mode analysis

External failure costs:

complaint resolution
product return and replacement
help line support
warranty work

Quality and Risk

"People bet their jobs, their comforts, their safety, their entertainment, their decisions, and their very lives on computer software. It better be right."

Example:

Throughout the month of November, 2000 at a hospital in Panama, 28 patients received massive overdoses of gamma rays during treatment for a variety of cancers. In the months that followed, five of these patients died from radiation poisoning and 15 others developed serious complications. What caused this tragedy? A software package, developed by a U.S. company, was modified by hospital technicians to compute modified doses of radiation for each patient.

Negligence and Liability

The story is all too common. A governmental or corporate entity hires a major software developer or consulting company to analyze requirements and then design and construct a software-based "system" to support some major activity.

The system might support a major corporate function (e.g., pension management) or some governmental function (e.g., healthcare administration or homeland security).

Work begins with the best of intentions on both sides, but by the time the system is delivered, things have gone bad.

The system is late, fails to deliver desired features and functions, is error-prone, and does not meet with customer approval.

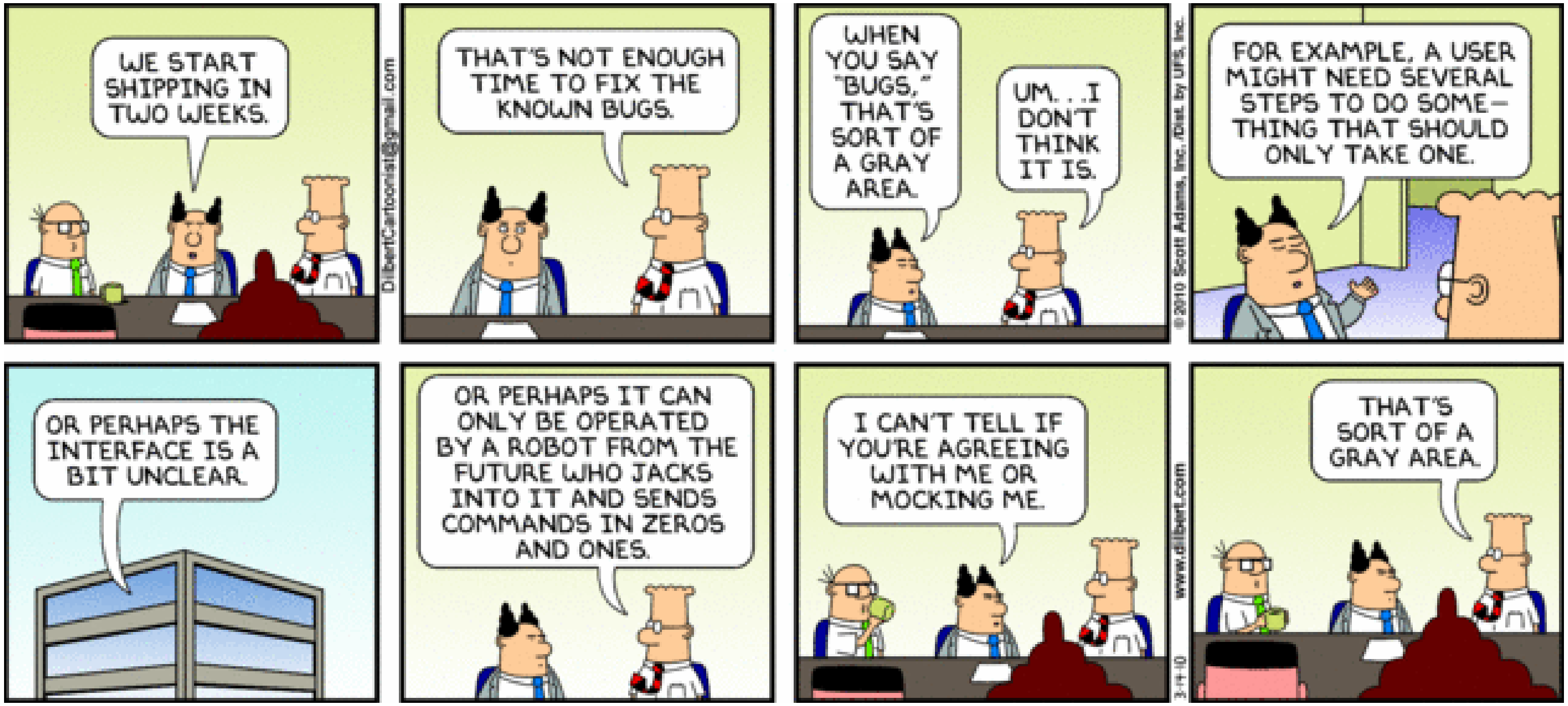
Litigation ensues.

Quality and Security

"Software security relates entirely and completely to quality. You must think about security, reliability, availability, dependability—at the beginning, in the design, architecture, test, and coding phases, all through the software life cycle [process]. Even people aware of the software security problem have focused on late lifecycle stuff. The earlier you find the software problem, the better. And there are two kinds of software problems. One is bugs, which are implementation problems. The other is software flaws—architectural problems in the design. People pay too much attention to bugs and not enough on flaws."

Gary McGraw (ComputerWorld)

Hurdle to Quality


WE START SHIPPING IN TWO WEEKS.
THAT'S NOT ENOUGH TIME TO FIX THE KNOWN BUGS.
WHEN YOU SAY "BUGS," THAT'S SORT OF A GRAY AREA.
UM. . .I DON'T THINK IT IS.
FOR EXAMPLE, A USER MIGHT NEED SEVERAL STEPS TO DO SOME-THING THAT SHOULD ONLY TAKE ONE.
OR PERHAPS THE INTERFACE IS A BIT UNCLEAR.
OR PERHAPS IT CAN ONLY BE OPERATED BY A ROBOT FROM THE FUTURE WHO JACKS INTO IT AND SENDS COMMANDS IN ZEROS AND ONES.
I CAN'T TELL IF YOU'RE AGREEING WITH ME OR MOCKING ME.
THAT'S SORT OF A GRAY AREA.
DilbertCartoonist@gmail.com
© 2010 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com
3-14-10

Confusion About Quality

Many words (ending with *ility*) are cited for Quality:

Augmentability	Compatibility
Expandability	Flexibility
Interoperability	Maintainability
Manageability	Modifiability
Operability	Portability
Reliability	Scalability
Survivability	Understandability
Usability	Testability
Traceability	Verifiability

Capers Jones., Software Engineering Best Practices, TMH ©2010

Method in Chaos – Analysis of Vista in 2009

When Vista is judged against the list, it can be seen how abstract and difficult to apply the list to commercial software

Quality Trait	Observation for Vista	Quality Trait	Observation for Vista
Augmentability	Difficult to apply	Compatibility	Poor
Expandability	Fairly good	Flexibility	Difficult to apply
Interoperability	Difficult to apply	Maintainability	Probably Poor
Manageability	Difficult to apply	Modifiability	Probably Poor
Operability	Difficult to apply	Portability	Poor
Reliability	Improving	Scalability	Marginal
Survivability	Difficult to apply	Understandability	Poor
Usability	Questionable	Testability	Poor, Complex
Traceability	Poor, Complex	Verifiability	Difficult to apply

Capers Jones., Software Engineering Best Practices, TMH ©2010

Achieving Software Quality

Critical success factors:

Software Engineering Methods

Project Management Techniques

Quality Control

Quality Assurance

Achieving Software Quality

Software Engineering Methods

- Understand the problem to be solved
- Create design that conforms to the problem
- Design and Software exhibit quality dimensions

Project Management Techniques

- Use estimation for achievable delivery dates
- Understand schedule dependencies and avoid short cuts
- Conduct risk planning

Achieving Software Quality

Quality Control

- Reviews
- Inspection
- Testing
- Measurements and Feedback

Quality Assurance

- Establish infrastructure for software engineering
- Audit effectiveness and completeness of quality control

Project Management

As per PMBOK(Project Management Book of Knowledge) published by PMI

The Project Management Context

- **Project Phases and Project Life Cycle :** Each project is unique endeavour
  - Phases can involve a degree of uncertainty
  - Each project phase is marked by completion of one or more deliverables
  - A deliverable is a tangible, verifiable work product
  - The project life cycle serves to define the beginning and the end of a project
- **Project Stakeholders :**
  - Are individuals and organizations actively involved in the project,
  - Whose interests are impacted by project execution and project completion. (Project manager, Customer, Performing organization, Sponsor)
- **Organizational Influences :** Project is influenced by the
  - Organizational Systems,
  - Cultures,
  - Style and Structure of Organization that set-up the project
- **Key General Management Skills**
  - Leading, Communicating, Negotiating, Problem Solving…
- **Socioeconomic Influences**
  - Standards and Regulations, Internationalization, Cultural influence

The Project Management Knowledge Areas

- ***Project Integration Management***
  - Ensure that various elements of the project are properly coordinated and integrated
  - **Processes:** Project Plan Development, Project Plan Execution, Overall Change Control

- ***Project Scope Management***
  - Ensure that the project includes all the work required,
  - And only the work required, to complete the project successfully
  - **Processes:** Initiation, Scope Planning, Scope Definition, Scope Verification, Scope Change Control

- ***Project Time Management***
  - Ensure timely completion of the project
  - **Processes:** Activity Definition, Activity Sequencing, Activity Duration Estimating, Schedule Development, Schedule Control

The Project Management Knowledge Areas (contd)

- ***Project Cost Management***
  - Ensure that the project is complete within the approved budget
  - **Processes:** Resource Planning,
    Cost Estimating, Cost Budgeting, Cost Control

- ***Project Quality Management***
  - Ensure that the project will satisfy the requirements
  - **Processes:** Quality Planning, Quality Assurance,
    Quality Control

- ***Project Communication Management***
  - Ensure timely and appropriate generation, collection, storage
  - And ultimate disposition of project information
    - **Processes:**
      Communications Planning,Information Distribution,
      Performance Reporting, Administrative Closure

The Project Management Knowledge Areas *(contd)*

- ***Project Risk Management***
  - Concerned with identifying, analyzing, and responding to project risk.
  - Maximizing the results of positive events
  - Minimizing the consequences of negative events
  - **Processes:** Risk Identification, Risk Quantification, Risk Response Development, Risk Response Control
- ***Project Procurement Management***
  - Acquire goods and services from outside the performing organization
  - **Processes:** Procurement Planning, Solicitation Planning, Solicitation, Source Selection, Contract Administration, Contract Close-out
- ***Project Human Resources Management***
  - Make the most effective use of people involved with the project
  - **Processes:** Organizational Planning, Staff Acquisition, Team Development
- ***Project Stakeholder Management***
  - Understand stakeholders and manage their expectations
  - **Processes:** Identify Stakeholders, Plan Stakeholder Management, Manage Stakeholder Engagement and Control Stakeholder Engagement

PMBOK

Project Processes

	Process Groups				
Knowledge Areas	**Initiating Process group**	**Planning Process group**	**Executing Process group**	**Monitoring & Controlling Process group**	**Closing Process group**
Project Integration Management	Develop Project Charter	Develop Project Management Plan	Direct and Manage work	Monitoring & Controlling project work Perform Integrated Change Control	Close Project or Phase
Project Scope Management		Plan Scope Management Collect Requirement Define Scope Create WBS		Validate Scope Control Scope	
Project Time Management		Plan Schedule Management Define Activities Sequence Activities Estimate Activity Resources Estimate Activity Durations Develop Schedule		Control Schedule	
Project Cost Management		Plan Cost Management Estimate Costs Determine Budget		Control Costs	
Project Quality Management		Plan Quality Management	Perform Quality Assurance	Perform Quality Control	
Project Human Resource Management		Plan Human Resource Management	Acquire Project Team Develop Project Team Manage Project Team		
Project Communication Management		Plan Communications Management	Manage Communications	Control Communications	
Project Risk Management		Plan Risk Management Identify Risks Performance Qualitative Risk Analysis Perform Quantitative Risk Analysis Plan Risk Analysis		Control Risks	
Project Procurement Management		Plan Procurements Management	Conduct Procuments	Control Procurements	Close procuments
Project Stakeholder Management	Identify Stakeholders	Plan Stakeholder Management	Manage Stakeholder Engagement	Control Stakeholder Engagement	

PMBOK

Managing Software Projects

(As per Pressman)

Project Management Concepts

- The Management Spectrum
  - The People
  - The Product
  - The Process
  - The Project

The Management Spectrum

- Effective software project management focuses on these items (in this order)
  - The people
    - Deals with the cultivation of motivated, highly skilled people
    - Consists of the stakeholders, the team leaders, and the software team
  - The product
    - Product objectives and scope should be established before a project can be planned
  - The process
    - The software process provides the framework from which a comprehensive plan for software development can be established
  - The project
    - Planning and controlling a software project is done for one primary reason…it is the only known way to manage complexity
    - In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns

People

The People: The Stakeholders

- Five categories of stakeholders
  - **Senior managers** – define business issues that often have significant influence on the project
  - **Project (technical) managers** – plan, motivate, organize, and control the practitioners who do the work
  - **Practitioners** – deliver the technical skills that are necessary to engineer a product or application
  - **Customers** – specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome
  - **End users** – interact with the software once it is released for production use

The People: Team Leaders

- Competent practitioners <u>often fail</u> to make good team leaders; they just don't have the right people skills
- Qualities to look for in a team leader, as per Jerry Weinberg
  - **Motivation** – the ability to encourage technical people to produce to their best ability
  - **Organization** – the ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product
  - **Ideas or innovation** – the ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application
- Successful Team leaders use problem-solving management style
  - Concentrate on understanding the problem to be solved
  - Manage the flow of ideas
  - Let everyone on the team know, by words and actions, that quality counts and that it will not be compromised

The People: Team Leaders - Traits

- According to Edgemon, key leadership traits of an effective project manager are
  - **Problem solving** – diagnose, structure a solution, apply lessons learned, remain flexible
  - **Managerial identity** – take charge of the project, have confidence to assume control, have assurance to allow good people to do their jobs
  - **Achievement** – reward initiative, demonstrate that controlled risk taking will not be punished
  - **Influence and team building** – be able to “read” people, understand verbal and nonverbal signals, be able to react to signals, remain under control in high-stress situations

The People: The Software Team

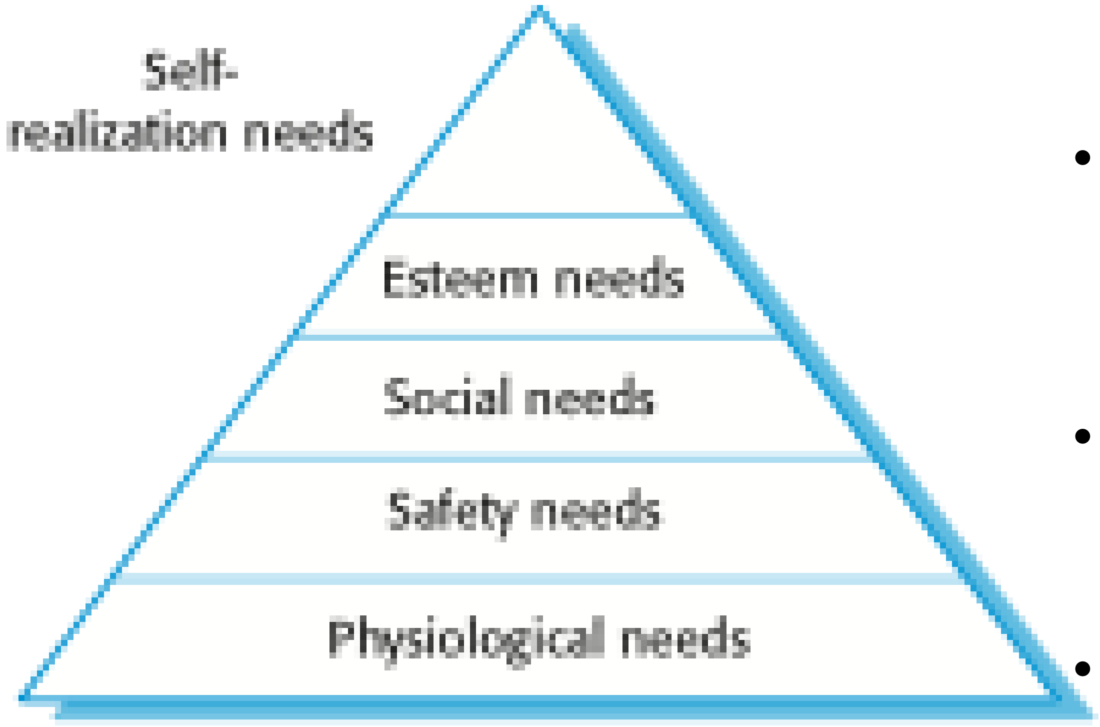
- Seven project factors to consider when structuring a software development team, as per Mantei
  - The difficulty of the problem to be solved
  - The size of the resultant program(s) in source lines of code
  - The time that the team will stay together
  - The degree to which the problem can be modularized
  - The required quality and reliability of the system to be built
  - The rigidity of the delivery date
  - The degree of sociability (communication) required for the project

The People: The Software Team – Paradigms

- Four <u>organizational paradigms</u> for software development teams
  - **Closed paradigm** – *traditional hierarchy of authority*; works well when producing software similar to past efforts; members are less likely to be innovative
  - **Random paradigm** – *depends on individual initiative of team members*; works well for projects requiring innovation or technological breakthrough; members may struggle when orderly performance is required
  - **Open paradigm** – *hybrid of the closed and random paradigm*; works well for solving complex problems; requires collaboration, communication, and consensus among members
  - **Synchronous paradigm** – *organizes team members based on the natural pieces of the problem*; members have little communication outside of their subgroups

– Constantine

Maslow’s Human needs hierarchy



- In software development groups, basic physiological and safety needs are not an issue.
- Social
  - Provide communal facilities;
  - Allow informal communications e.g. via social networking
- Esteem
  - Recognition of achievements;
  - Appropriate rewards.
- Self-realization
  - Responsibility;
  - Training - people want to learn more.

Sommerville, I., Software Engineering, Pearson Education, 7th Ed., 2005

The People: The Software Team - Problems

- Five factors that cause <u>team toxicity</u> (i.e., a toxic team environment)
  - A frenzied work atmosphere
  - High frustration that causes friction among team members
  - A fragmented or poorly coordinated software process
  - An unclear definition of roles on the software team
  - Continuous and repeated exposure to failure

- How to avoid these problems
  - Give the team access to all information required to do the job
  - Do not modify major goals and objectives, once they are defined, unless absolutely necessary
  - Give the team as much responsibility for decision making as possible
  - Let the team recommend its own process model
  - Let the team establish its own mechanisms for accountability (i.e., reviews)
  - Establish team-based techniques for feedback and problem solving

The People: Challenges

- Key characteristics of modern software make projects fail
  - Scale,
  - Uncertainty,
  - Interoperability

- To better ensure success
  - Establish effective methods for coordinating the people who do the work
  - Establish methods of formal and informal communication among team members

Agile Teams

- Team members must have trust in one another.
- The distribution of skills must be appropriate to the problem.
- Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.
- Team is “self-organizing”
  - An adaptive team structure
  - Uses elements of Constantine’s random, open, and synchronous paradigms
  - Significant autonomy

Product

The Product

- The scope of the software development must be established and bounded
  - **Context** – How does the software to be built fit into a larger system, product, or business context, and what constraints are imposed as a result of the context?
  - **Information objectives** – What customer-visible data objects are produced as output from the software? What data objects are required for input?
  - **Function and performance** – What functions does the software perform to transform input data into output? Are there any special performance characteristics to be addressed?
- Software project scope must be unambiguous and understandable at both the managerial and technical levels

Problem Decomposition

- Sometimes called *partitioning* or *problem elaboration*
- Once scope is defined …
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects

  or
  - It is decomposed into a set of problem classes
- Decomposition process continues until all functions or problem classes have been defined

Process

The Process

- Each project needs a process to be set up
  - The project manager must decide which process model is most appropriate based on
    - The customers who have requested the product and the people who will do the work
    - The characteristics of the product itself
    - The project environment in which the software team works
  - Once a process model is selected, a preliminary project plan is established based on the process framework activities
  - Process decomposition then begins
  - The result is a complete plan reflecting the work tasks required to populate the framework activities
- Project planning begins as a blending of the product and the process based on the various framework activities

The Process

- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones

Project

The Project: A Common Sense Approach

- Start on the right foot
  - Understand the problem; set realistic objectives and expectations; form a good team
- Maintain momentum
  - Provide incentives to reduce turnover of people; emphasize quality in every task; have senior management stay out of the team's way
- Track progress
  - Track the completion of work products; collect software process and project measures; assess progress against expected averages
- Make smart decisions
  - Keep it simple; use COTS or existing software before writing new code; follow standard approaches; identify and avoid risks; always allocate more time than you think you need to do complex or risky tasks
- Conduct a post mortem analysis
  - Track lessons learned for each project; compare planned and actual schedules; collect and analyze software project metrics; get feedback from teams members and customers; record findings in written form

– John Reel

The Project: 10 Signs that it is in Jeopardy

(as per John Reel)

- Software people don't understand their customer's needs
- The product scope is poorly defined
- Changes are managed poorly
- The chosen technology changes
- Business needs change (or are poorly defined)
- Deadlines are unrealistic
- Users are resistant
- Sponsorship is lost (or was never properly obtained)
- The project team lacks people with appropriate skills
- Managers (and practitioners) avoid best practices and lessons learned

The Project: The W^{5}HH Principle

A series of questions that lead to a definition of key project characteristics and the resultant project plan

- **Why** is the system being developed?
  - Assesses the validity of business reasons and justifications
- **What** will be done?
  - Establishes the task set required for the project
- **When** will it be done?
  - Establishes a project schedule
- **Who** is responsible for a function?
  - Defines the role and responsibility of each team member
- **Where** are they organizationally located?
  - Notes the organizational location of team members, customers, and other stakeholders
- **How** will the job be done technically and managerially?
  - Establishes the management and technical strategy for the project
- **How** much of each resource is needed?
  - Establishes estimates based on the answers to the previous questions

Barry Boehm [Boe96]

Critical Practices

(as per US Dept of Defence)

- Formal risk management
- Empirical cost and schedule estimation
- Metrics-based project management
- Earned value tracking
- Defect tracking against quality targets
- People aware project management

Project Management Performance

Following table lists how well key project management functions are performed based on observations within about 150 companies. The scoring range is from -10 for very poor performance to +10 for excellent performance

Project Management Function	**Score**	**Definition**
Reporting “red flag” problems	-9.5	Very poor
Defect removal efficiency measurements	-9.0	Very poor
Benchmarks at project completion	-8.5	Very poor
Requirements change estimating	-8.0	Very poor
Postmortems at project completion	-8.0	Very poor
Quality estimating	-7.0	Very poor
Productivity measurements	-6.0	Poor
Risk estimating	-3.0	Poor
Process improvement tracking	-2.0	Poor
Schedule estimating	1.0	Marginal
Initial application sizing	2.0	Marginal
Status and progress tracking	2.0	Marginal
Cost estimating	3.0	Fair
Value estimating	4.0	Fair
Quality measurements	4.0	Fair
Process improvement planning	4.0	Fair
Quality and defect tracking	5.0	Good
Software assessments	6.0	Good
Cost tracking	7.0	Very good
Earned value tracking	8.0	Very good
Average	-0.8	Poor

Capers Jones., Software Engineering Best Practices, TMH ©2010

Summary – Software Quality

- Software quality is the concern of every software process stakeholder.
- If a software team stresses quality in all software engineering activities, it reduces the amount of rework that must be done. This results in lower costs and improved time-to-market.
- To achieve high quality software, four elements must be present:
  - proven software engineering process and practice,
  - solid project management,
  - comprehensive quality control, and
  - the presence of a quality assurance infrastructure.
- Quality software meets its customer's needs, performs accurately and reliably, and provides value to all who use it.

Summary of Software Project Management

•Project management involves the planning, monitoring, and control of people, process, and events that occur during software development.

•Everyone manages, but the scope of each person's management activities varies according his or her role in the project.

•Software needs to be managed because it is a complex undertaking with a long duration time.

•Managers must focus on the fours P's to be successful (people, product, process, and project).

•A project plan is a document that defines the four P's in such a way as to ensure a cost effective, high quality software product.

•The way to be sure that a project plan worked correctly is by observing that a high quality product was delivered on time and under budget.

Thank You