



**QAI /QAAM 2011 Conference**  
**“Proven Practices For**  
**Managing and Testing IT**  
**Projects”**



**Proven Development Methodologies**  
**for Managing and Testing IT Projects**

**Brad Eichstadt**  
PMP, CSQA, MCP

10<sup>th</sup> Annual  
Mid-Atlantic  
Software Quality  
Conference

**Date: September 28, 2011**

## Presentation Timeline

- Presentation Overview & Expectations 5 Minutes
- Introductions and Expectations Capture 15 Minutes
- Topic Discussion 60 Minutes
- Break
- Topic Discussion Wrap-up 20 Minutes
- Agile Testing Comparison Exercise 20 Minutes
- Q&A & Expectations Review 10 Minutes

## Objectives

- Welcome everyone.
- In this session, we will have the following objectives:
  - Answering the question, “What development methodologies have proven successful in delivering quality projects?”
  - Reviewing various IT methodology approaches to development, testing, quality reviews, managing project quality tasks within time and constraints and how these approaches impact quality
  - Reviewing how the Project Management Institute (PMI) defines and implements Quality Management as part of development and project management methodologies
  - Completing a Methodology Testing Comparison Exercise for Participants

## Background

- Presenter
  - Brad Eichstadt, CSQA, PMP, MCP
    - TEL: 816-860- E-MAIL: brad.eichstadt@umb.com
    - An IT professional with over 23 years experience in Information Technology.
    - Certified as a Project Management Professional (PMP) and has also achieved certifications as a CSQA (Quality), a CPIM--Certified Production & Inventory Management (1991-2009)—professional (Supply Chain & Quality Control) and as a Mercury Certified Professional (Automated Testing).
    - Is currently a Manager at UMB in the IT Core Systems Development and Support group.
    - Has helped develop and teach PMP exam prep program courses multiple companies.
    - Presented as board member at local Kansas City PMI conference on, “What are the Characteristics of a Successful PMO?”

## Traditional "Project Triangle" Triple Constraint

- Time, Cost, Scope
  - Ideal situation is when a project completes on time, on budget and meets user scope and quality expectations
  - Unfortunately, Many Projects Don't Meet These Goals



- Note, PMI's 4<sup>th</sup> Edition of the PMBOK has moved away from the strict constraints of the project triangle to emphasize balancing the competing project constraints including scope, quality, schedule, budget, resources and risk.

## Which if Either of These Projects was Successful?

- Project 1
  - Project completed on budget
  - Project effectively completed on time (within < 1 month after target date out of 4 year development cycle)
    - Delivery date driven by marketing and financial objectives, not legal requirements
  - Project did not fully meet quality/scope requirements
  
- Project 2
  - Project was over budget by > 30%,
  - Project completion completed ~30-40% late, roughly two to three months beyond original six month plan
    - Delivery date driven by marketing and financial objectives, not legal requirements
  - Project fully met and exceeded quality/scope requirements

## Project 1 – The Titanic

- For several years, White Star had been losing ground to competition. The Executives responded with a campaign to replacing the legacy fleet with three new super ships using the latest technology
  - Testing was compromised
    - To meet launch date, proper quality reviews and testing (i.e. shakedown sea cruises) had been severely cut back from several weeks to just one day
  - Project Managers bowed to arbitrary time pressures
    - Launch date was 100% a marketing driven decision. Advertising had been in place for months. Pressure was there to launch additional sister ships. Any delay was considered unacceptable.
  - Lack of requirements completion
    - Functional requirements (i.e. passenger comfort, service) were rigidly enforced
    - Non-functional requirements (i.e. safety, performance, maintenance, etc.) were compromised to be able to meet the launch date.
  - Scope creep
    - Project sponsor promised last minute operational service level objective requirement just days before sailing. This led to pushing the Titanic far harder than originally planned
  - **RESULT:** 4 years in development, 4 days in operation

## Project 2 - Jaws

- Nickname by the film crew for the shoot was “Flaws”
  - Filming suffered from extensive hardware failures and time delays with the mechanical sharks
  - Project went well over budget
  - Testing effort was not ignored and instead was extremely successful
    - Due to the film's success in advance test screenings, studio executives decided to distribute it in a much wider release than ever before, helping further drive the success of the movie.
  - Project manager did not bow to timeline pressures
    - Spielberg was almost fired when he refused to call a completion to the project before he was done with post-filming changes
    - The additional production and testing time allowed several script improvements be made that later proved to be some of the biggest scenes in the movie
  - Additional money was spent to improve quality even though budget was greatly exceeded
    - Spielberg paid \$3000 of his own money to shoot the floating head scene when Universal denied him additional funds
  - **RESULT:** Project broke all attendance and box office records to that point. Became the basis for summer blockbuster releases

## **Proposition: Quality/Scope is the Most Important Side of the Constraints Triangle**

- Barring certain exceptions, a project that is over-budget and/or over-time can still be “successful” if it fully meets/exceeds scope/quality requirements for user fitness for use
  - For this to be “successful “requires solid testing and adherence to good quality assurance practices despite cost/time pressures as well as clear communication
- Conversely, it is not be possible for a project to be on cost, on time, not meet scope/quality requirements and yet still be successful
  - If the project has not met scope/quality requirements for user fitness, it simply has not spent enough time/money to develop, test and implement properly

# Importance of Proven Development Methodologies for Managing and Testing IT Projects

## Why do IT Projects Fail/Succeed?

- Chaos Report – Published by Standish Group
  - Original 1994 Report showed only 17% of IT projects succeeded.
  - Latest 2010 summary report shows ~ 1/3 of projects are succeeding, down slightly from earlier reports, but up significantly from the original.  
Why?
  - Per Standish Group the reasons are:
    - Better project management
    - More Iterative development (e.g. new project development methodologies)
    - Emerging web infrastructure

## Why do IT Projects Fail/Succeed?

- Critics of the Standish Group - 2009 & 2010 IEEE ["The Rise and Fall of the Chaos Report Figures"](#)
  - Dispute how Standish Group defines project success
  - Contend that the Standish Report undercounts project success
  - Per these Critics IT Projects Succeed when:
    - They accurately predict budget & schedule (e.g. EQF measures, etc.)
    - Adequately account for the "context" of project. Quote:
      - "There's no way around including more context (or totally different definitions) when assessing successful and challenged projects. However, the Standish definitions don't consider a software development project's context, such as usefulness, profit, and user satisfaction."

## Why do IT Projects Fail/Succeed?

- IFPUG
  - Primary torch bearer of Function Point Analysis standards
  - Maintains FP counting practices manual and defines ISO Function Point analysis standards.
  
  - Per IFPUG practitioners the reasons for project success include:
    - Better measurement/prediction of software functionality estimates
    - Delivery of required function points to meet user functionality

## So What Do these Groups All Agree Upon?

- The project must deliver on planned functionality (i.e. must deliver on Quality)
  - While project groups disagree about how important budget & schedule are, the one thing that all appear to agree upon is that the project must ultimately deliver on what it is supposed to do to be successful.
  - The major project management functions of measurement, prediction and quantification of required functionality are key to delivering successful projects.
- Successful delivery of planned functionality relies heavily on the development methodology selected
  - This holds particularly true for IT development projects

## So What is a Project Methodology?

- Project Management Methodology Definition
  - A project management methodology defines a set of Project Management process groups, their related processes and the related control functions that are consolidated and combined into a functioning and unified whole.  
(PMBOK® Guide Chapter 4 –page 85)

## How Does Better Project Management Help Quality?

- Good Project Management Methodologies...
  - Provide adequate time for requirements capture
  - Emphasize clear scope definition and change management procedures
  - Support strong testing and quality assurance practices
  - Require good tracking and communication on relationship of Scope, Time & Budget
- RESULT: When good project management methodologies are followed correctly, customers have clear expectations on what product they are getting and why and the product is well tested, delivering what is expected.
  - If problems arise during the project, they are identified and communicated earlier, not later, allowing expectations to be adjusted.

## How Do Project Methodologies & Iterative Development Help Quality?

- Good Project Management Methodologies...
  - Recognize that longer projects have inherent risk vs. shorter projects
  - Support scoping and division of projects into manageable work breakdown structure phases which facilitates iterative development
  - Provide ongoing information in a timely fashion to facilitate multiple go/no go points at which the project can be halted if conditions have changed
  - RESULT: When good project management methodologies are followed correctly, projects can be delivered more smoothly in smaller increments over time vs. “big bang” approaches
    - This process is analogous to manufacturing in which smaller lot sizes and more efficient changeovers are leading practices in enhancing quality and product delivery

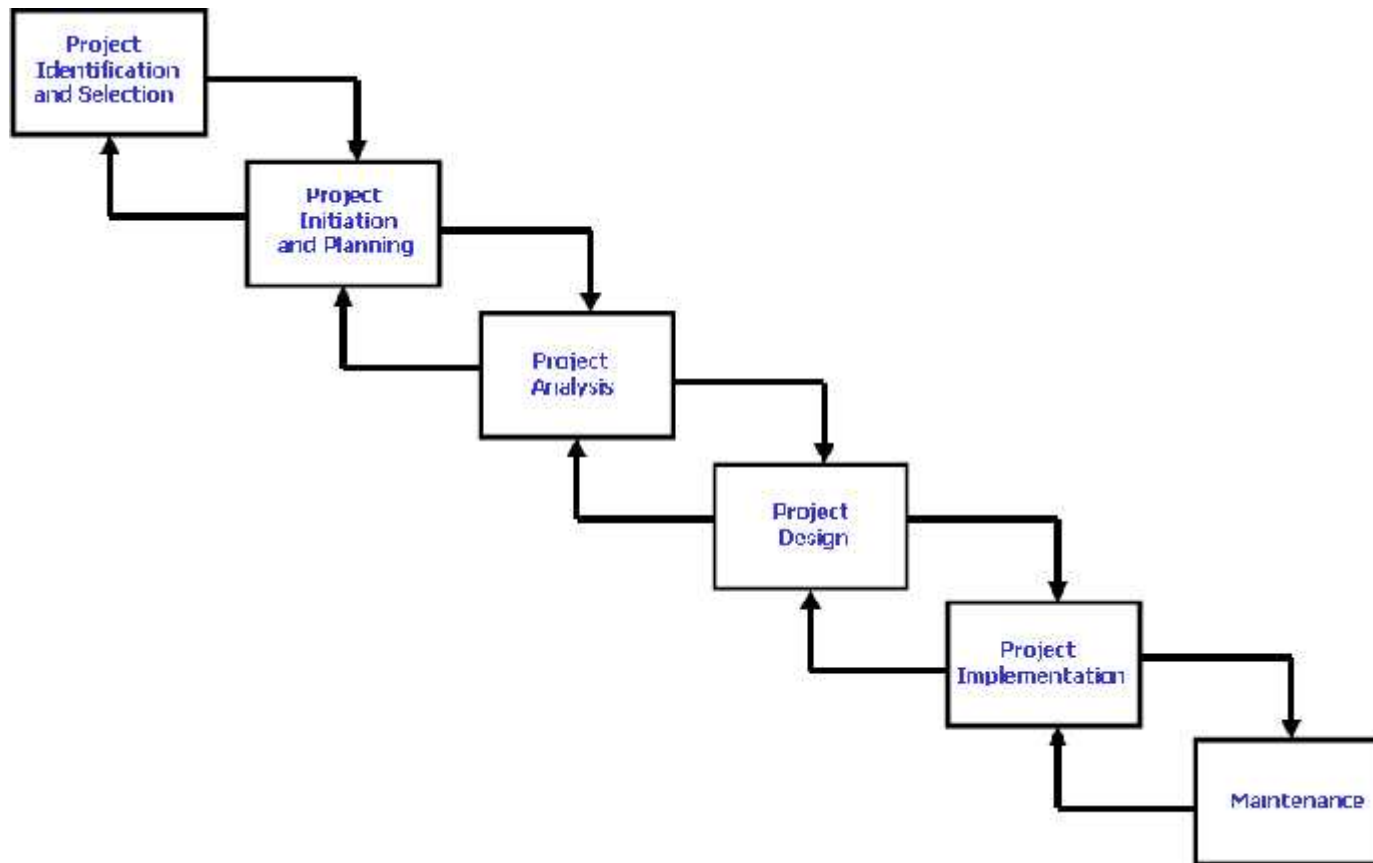
## Multitude of Iterative Project Management Methodologies

- Adaptive Project Framework
- Agile Software Development
  - including Crystal Methods, Dynamic Systems Development Model (DSDM) & Scrum
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Information Technology Infrastructure Library (ITIL)
- Joint Application Development (JAD)
- Lean Development (LD)
- PRINCE2
- Project+
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Spiral
- Systems Development Life Cycle (SDLC)
  - A.k.a. Waterfall, Traditional
- TenStep Project Management Process
- Client Specific

## Methodology Approach, Benefits, Drawbacks

- We are going to explore some common methodologies in more detail
  - Description – What is the methodology and what are its core values with respect to requirements and quality
  - Approach – Comparison of various Approach Attributes
  - Benefits – Comparison of what the perceived benefits are for each methodology
  - Drawbacks – Comparison of what the perceived drawbacks/ criticisms are for each methodology

## Systems Development Life Cycle (SDLC) - Approach



## Systems Development Life Cycle (SDLC) - Approach

- A.K.A “Waterfall”. SDLC is a multi-phased and process-oriented approach comprised of standardized tasks to complete during each phase of the project by both the IT team and the business units
- The first widely accepted IT development methodology adopted by most organizations
- The core values of the SDLC approach are:
  - Each phase in the SDLC has a discrete deliverable (document) and a milestone (phase review)
  - Highly planned and structured approach focused on the development of high-quality systems to meet the exacting needs of the user

## Systems Development Life Cycle (SDLC) - Benefits

- Commonly perceived benefits of SDLC:
  - Allows for departmentalization and managerial control of resources
  - Deadlines for each phase and task of development (i.e. a “roadmap”)
    - In theory, leads to the project being delivered on time because each phase has been planned in detail
  - Includes phase milestones and deliverables to ensure the project team does not move to the next phase without first obtaining approval
  - Includes sequentially ordered phases which can utilize an iterative approach in case steps need to be repeated
  - High Systems Quality
    - Detailed approach ensures that “the system that was specified in the analysis phase is actually delivered in the implementation phase”
  - Approach is familiar to many/most business users

## Systems Development Life Cycle (SDLC) - Criticisms

- Common criticisms of the SDLC approach include:
  - Methodology does not deliver quality projects due to aging of requirements, long lead teams and difficulty in correcting bugs once discovered
    - Project “freezes” project requirements too early in the systems development
    - Does not embrace the inevitable changes and revisions that become necessary with most projects
    - Once an application is in the testing stage, it is very difficult to go back and change something that was not thought of in the concept stage
  - Immediately requires “updates” to new system that just went in
  - Long requirements and development timeframes increase risk to project delivery

## Adaptive Project Framework - Description

- The fundamental concept underlying the adaptive project framework (APF) is that scope is variable within specified time and cost constraints (i.e. timeboxing)
- APF attempts to maximize quality & business value by adjusting scope at each iteration
- The core values of the APF approach are:
  - Client-focused
  - Client-driven
  - Incremental results early and often
  - Continuous question and introspection
  - Change is progress to a better solution
  - Don't speculate on the future

## Agile - Description

- Agile is a conceptual framework for undertaking software development projects
  - Several variations exist (e.g. Crystal Methods, Dynamic Systems Development Model, Scrum, Velocity Tracking, Agile Unified Processes)
- Agile approaches typically try to reduce risk and improve quality by developing software that can be completed in short iterations lasting one to four weeks (i.e. timeboxing)
- The core values of the Agile approach are:
  - Emphasizes real-time communication
  - Prefers face-to-face over written documents
  - Centrally located teams include all people necessary to finish the software
  - Projects are reviewed on a weekly basis by business unit funding the project to evaluate progress

## Rapid Application Development (RAD) - Description

- The primary focus of RAD is to “speed up” systems development and reduce the amount of firm resources allocated to IS projects by fully integrating the business worker into requirements and approval processes
- RAD was not specifically focused or designed to improve quality
- The core values of the RAD approach are:
  - IT & business unit team members work as an independent team (RAD Team)
  - “General” specification developed first, prototype, then refine
  - Relies heavily on automated development tools like: Prototyping tools, code generators, CASE tools
  - Extremely iterative

## Extreme Programming (XP) - Description

- The primary focus of XP is to take core practices derived from generally accepted best practices and implement them to extremes.
  - Often considered an Agile Methodology
- XP attempts to lower the cost of software changes and minimize customer impact of software requirements changes (i.e. decrease impacts to quality)
- The core values of the XP approach are:
  - Developers and end-users communication is “good” for increasing fitness for use
  - Simpler code is more likely to work and increases quality
  - Desk checks/code reviews are good so pair up
  - Learning is good (i.e. Test early, test often)

## Approach Comparisons

General Approach Comparison	Iterative Requirements	Timeboxed	Requires Embedded Client Involvement	Independent/Centrally Located Teams	Business "Owns" Project	Emphasizes Written Documentation	Real Time Communication	Frequent Testing	Relies on Automated Tools	Prototyping
Methodology										
- SDLC (aka Waterfall, Traditional)	No	No	No	No	No	Yes	No	No	No	Sometimes
- Adaptive Project Framework	Yes (Varies)	Yes (Varies)	Yes	No	No	No	No	Yes	No	No
- Agile	No	Yes (Weeks)	Yes	Yes	Yes	No	Yes	Yes	Yes	No
- Rapid Application Development	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	Yes
- XP (Extreme Programming)	Yes (Days)	Yes (Days)	Yes	No	No	No	Yes	Yes (Immediate)	No	No

## Perceived Benefits

Perceived Benefits Comparison	Delivers Requirements	Ties Requirements to Company Goals	Improved Speed to Delivery	Reduce Project Risk	Emphasizes Continuous Improvement	Suitable for Large Projects	Formal Timeline Communication with Upper Management	Suitable for Outsourcing/Offshoring	Business User Familiarity
<b>Methodology</b>									
- SDLC (aka Waterfall, Traditional)	Varies	No	No	No	No	Yes	Yes	Yes	Yes
- Adaptive Project Framework	Yes	Yes	No	Yes	Yes	No	No	No	No
- Agile	Yes	Yes	Yes	Yes	Yes	No	No	No	No
- Rapid Application Development	Yes	No	Yes	No	Yes	Yes	No	No	Yes
- XP (Extreme Programming)	Yes	Yes	Yes	Yes	Yes	No	No	No	No

## Perceived Criticisms

<b>Perceived Criticisms Comparison</b>	<b>Hard to Change Requirements</b>	<b>Not Suitable for "Hard" Date Projects</b>	<b>Not Suitable for "Hard" Requirements Projects</b>	<b>Lack of Documentation</b>	<b>Projects Incorrectly Killed</b>	<b>Developer Skills/ Familiarity</b>
<b>Methodology</b>						
- SDLC (aka Waterfall, Traditional)	Yes	No	Yes	No	No	Yes
- Adaptive Project Framework	No	Yes	No	Yes	No	No
- Agile	No	Yes	No	Yes	Yes	No
- Rapid Application Development	No	No	No	Yes	No	Yes
- XP (Extreme Programming)	No	Yes	No	Yes	Yes	No (Pairing)

# Proven Agile Development & Testing Methods

## Most Agile Methodologies Include the Following Traits

- Iterative development
- Responsive to changing requirements
- Focus on delivering Business Value
- Many opportunities to re-prioritize and re-evaluate
- 'Travel light'
- Favour conversations over documentation and formal process
- Customer (Product Owner) is part of the team
- Everything has an opportunity cost, so don't do anything that doesn't add value!

## **Most Agile Testing Methodologies Also Include The Following Traits**

- Empowered customer rep either on-site or accessible to the team, provides constant feedback on quality and priorities during development and testing
- Running, tested features developed in order of business priority
- Test-Driven throughout development, not just in “testing phase”
- High emphasis on testing early, testing often & testing fast
- Restrospectives
  - The team reviews and improves its own processes regularly (quality improvement cycle)

## Agile Requirements Tools

- Agile Theme
  - A theme is a collection (group) of user stories.
- Agile Epic
  - Epic is just a large user story. It is possible to break up epic to several user stories. It is generally required to break up epics to facilitate testing
- Agile User Story
  - a user story is one or more sentences in the everyday or business language of the end user that captures what the user wants to achieve (wiki). Most common requirements tool for Agile
- Agile Task
  - A small portion of a user story. Generally something that can be completed in a couple of hours or less

## Agile Estimation (Sizing) Tools

- Hours
  - Agile Teams **only estimate Tasks in hours.**
- Story Points
  - Story Points use Fibonacci series (relative size comparison) to estimate larger pieces of work. Story Points **can measure things than can't easily be measures in hours** – e.g. complexity
- T-shirt Sizing
  - Used to estimate larger requirements

## Agile Test Management – Test Manager Role

- Manage effectiveness of the test effort
- Manage efficiency of the test effort
- Ensure compliance with required standards
- Ensure testing capabilities remain current
- Ensure independence of testing efforts
- Provide overall test QA
  - How does this reconcile with Agile Team responsibility for development and testing.

## Successful Agile Testing Generally Encompasses

- Continuous Integration of Testing Throughout Development
- Frequent check-ins of workable code (tested every time)
- Mature Test Automation (necessary to facilitate ongoing regression testing)

## Common Agile Test Strategies

- Test Driven Development
- Defect Driven Testing
- Blitz Attack

## Test Driven Development - Process

- Develop a test case that fails
- Write code sufficient to pass test case
- Repeat above steps until test case passes

## Test Driven Development - Characteristics

- Driven by the program design
- Creates a solid product
- Requires maturity in understanding overall program design and requirements (a.k.a. "enlightenment")

## Defect Driven Testing - Process

- Find a bug
- Add a test for the bug (to all programs)
- Recode and improve

## Defect Driven Testing - Characteristics

- Incrementally corrects code
- Test suite improves in depth and complexity
- Well-suited for optimization/retrofit of existing code

## Blitz Testing – Process & Characteristics

- Comprehensive across entire product
- Considered to have a breadth of testing, not detailed depth.
- Requires use of standard templates

## Common Agile Test Planning Pitfalls

- Bad estimation. Developers and testers underestimate the effort and resources required for testing. Consequently, they miss deadlines or deliver a partially tested system to the client because of lack of resources (e.g. instead of Done Done stories we have rework stories)
- Untestable requirements. Ambiguous or undefined requirements which renders them impossible or difficult to test. Note, this should be prevented through proper story development, but often is not.
- Insufficient test coverage. The number of test cases developed cannot/do not test the full functionality of the system. Often tied to the bad estimation and lack of resources as noted above. Note, can be exacerbated by poor story development which strives for independent testing, but may miss inter-dependent tests.

## Test Planning – Pitfalls Continued

- Testing too late. Testing too late during the development process (yes, it happens in Agile too) leaves little time to adjust when tests find major defects.
- Lack of contingency planning. There is no contingency or slack space in the testing plan to deal with major defects identified during testing.
- Failure to maintain automated tests (i.e. they “rust”)



## Common Agile Test Configuration Pitfalls

- Ignoring exceptions (aka happy path or Prozac path testing). Developers and testers sometimes bias testing toward regular or "happy path" cases. Testing such as this can overlook system exceptions, leading to a system that works only when everything goes right (hopefully most of the time), but not all the time.
  - Variation including: "Stress-easy button" testing. Failing to test stories to a sufficient stress level. When testing does not place the system under sufficiently high levels of stress, it fails to investigate system breakpoints, which therefore remain unknown.
- Non robust test data. The test data fails to include the range of all possible data values—that is, it omits extreme or boundary values.
- Environmental mismatch. Testing the system in an environment that is not the same as the environment on which it will be installed doesn't tell you about how it will work in the real world.

## Common Agile Testing Structural Pitfalls

- Configuration mismanagement. In some cases, a software release contains components that have not been tested at all or were not tested with the released versions of the other components. Developers can't ensure that the component will work as intended.
- Non-independent testing. When the development team carries out testing, it can lack the objectivity of an independent testing team.

## Summary

- Project success, including quality, can be improved by good project management
- Projects can be “successful” even when faced with certain obstacles
  - Key is meeting/exceeding user expectations
- Traditional methodology approaches frequently face challenges in meeting user expectations
- Many new project management methodologies have been developed to overcome deficiencies in traditional methodologies
  - New methodologies have their own deficiencies which must be monitored
- PMI Approach fundamentally recognizes, supports and complements good quality practices

## Questions



## Contact Information

- **Brad Eichstadt**, Manager,
- UMB Bank
- (816) 860-
- [Brad.Eichstadt@UMB.com](mailto:Brad.Eichstadt@UMB.com)



# Appendix A: Project Management Institute (PMI) Approach to Quality

## PMI – Approach Recognizes Quality Link

- **Quality Management:**

- Modern quality management and project management complement each other. For example, both disciplines recognize the importance of:
  - Customer satisfaction
  - Prevention over inspection
  - Management responsibility
  - Continuous Improvement

- PMBOK® Guide Chapter 8 – Project Quality Management page 181

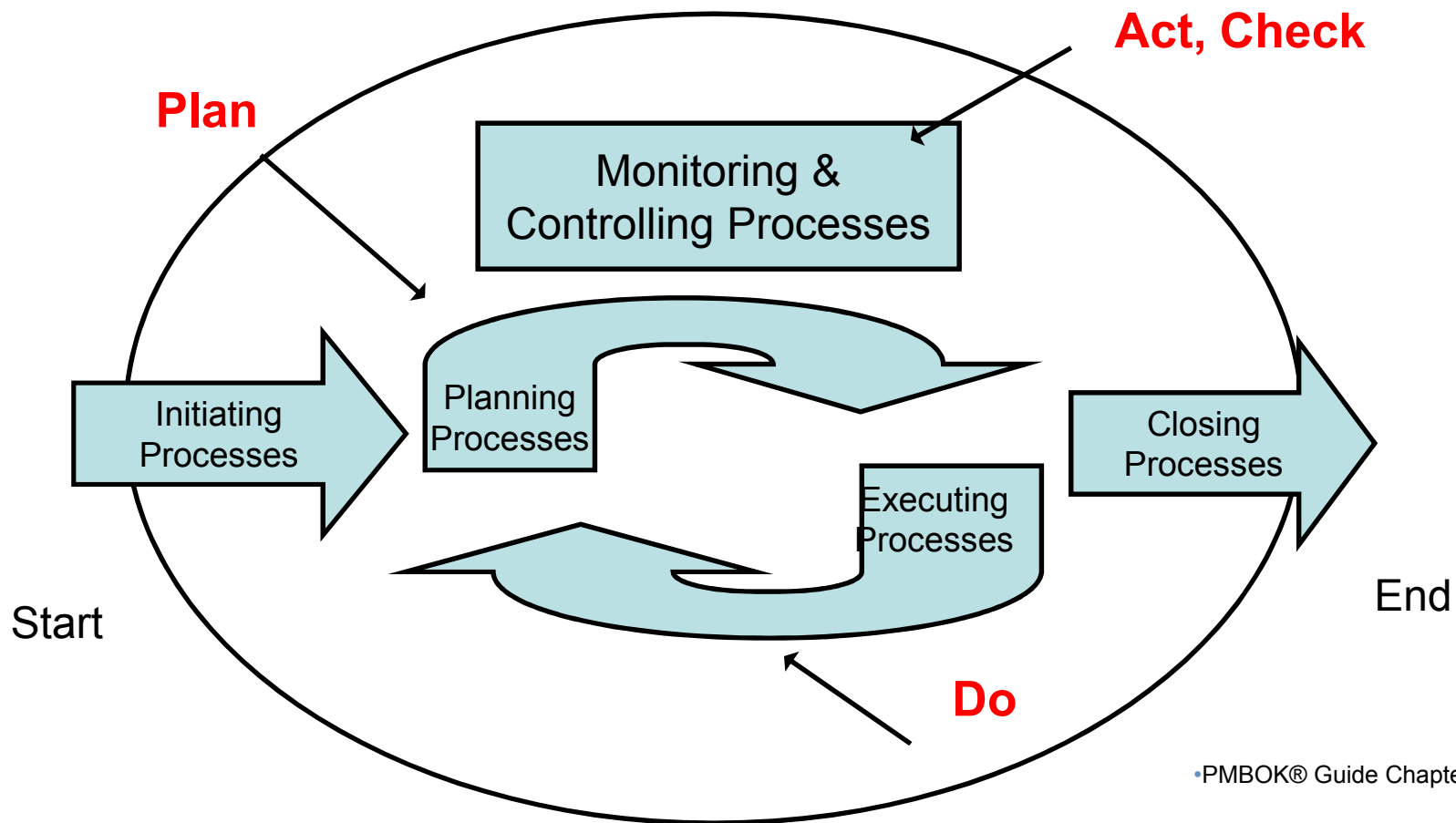
## PMI – Uses Quality Organizational Approach

- **An underlying concept for the interaction among project management processes is the plan-do-act-check cycle (Shewhart/Deming ASQ Handbook, pages 13-14, American Society for Quality, 1999)**

- PMBOK® Guide Chapter 3 – Project Management Processes for a Project - page 38-39)

## PMI – Uses Quality Organizational Approach

Project Management Process Groups Mapped to the Plan-Do-Check-Act Cycle



•PMBOK® Guide Chapter 3 –page 40)

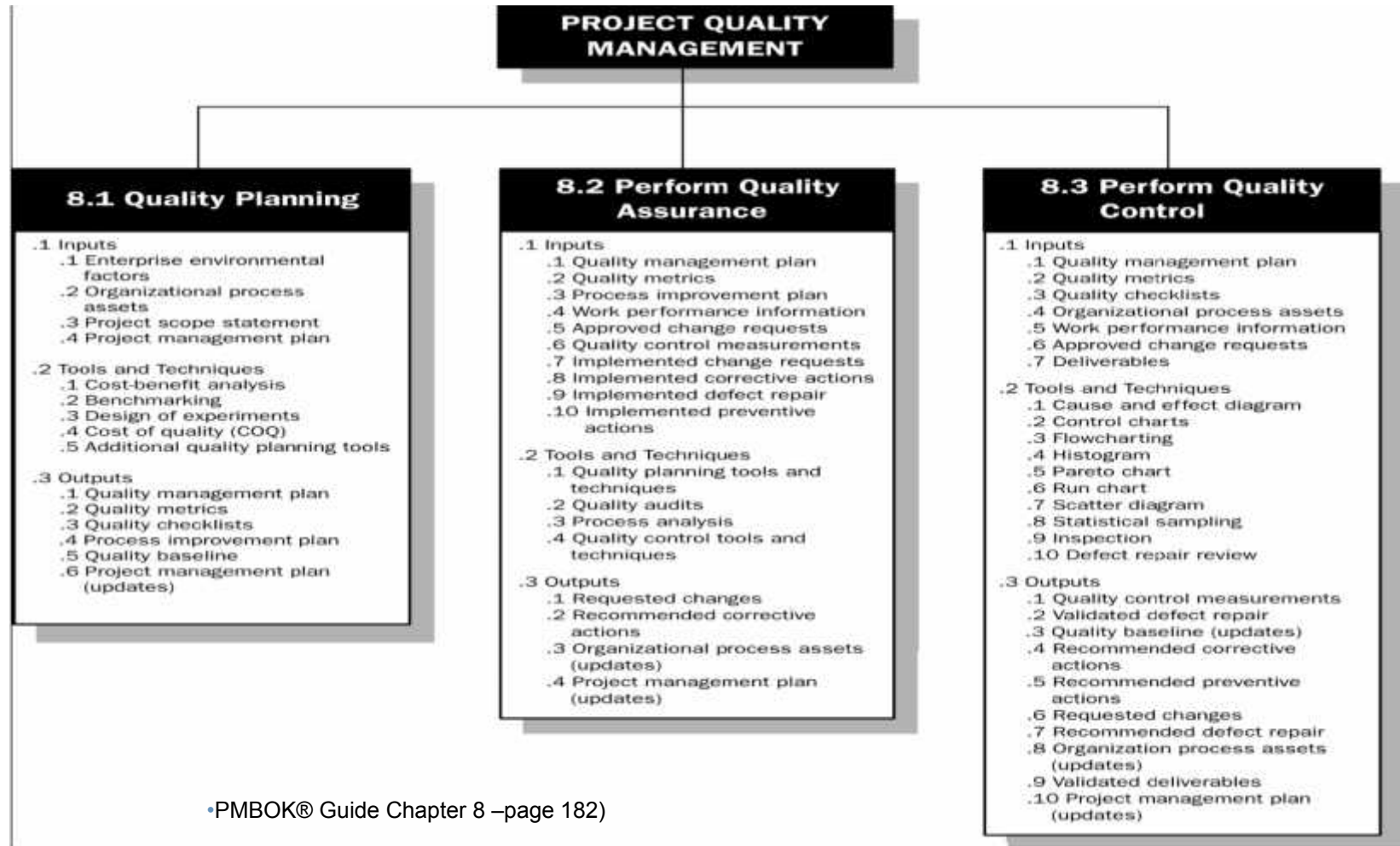
## PMI – Approach is Comprehensive

- Section 3: The Project Management Knowledge Areas
  - Project Quality Management (Guide, Chapter 8, Page 179)
    - Includes all the activities of the performing organization that determine quality policies, objectives, and responsibilities so that the project will satisfy the needs for which it was undertaken
    - It implements the quality management system through the policy, procedures, and processes of quality planning, quality assurance, and quality control, with continuous process improvement activities conducted throughout, as appropriate

## **PMI – Approach is Comprehensive**

- Section 3: The Project Management Knowledge Areas
  - Includes following quality management tools & techniques
    - Total Quality Management (TQM)
    - Six Sigma
    - Failure Mode and Effect Analysis,
    - Design Reviews
    - Voice of the Customer
    - Cost of Quality (COQ)
    - Continuous Improvement

## PMI – Approach is Detailed



•PMBOK® Guide Chapter 8 –page 182)