SOFTWARE ENGINEERING

SOFTWARE PROJECT MANAGEMENT

Saulius Ragaišis
saulius.ragaisis@mif.vu.lt
CSC2008 SE “Software Project Management”

Learning Objectives:

• Demonstrate through involvement in a team project the central elements of team building and team management.

• Prepare a project plan for a software project that includes estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management.

• Indicate an approach to risk that will help to secure the on-time delivery of software.

• Compare and contrast the different methods and techniques used to assure the quality of a software product.
WHAT IS A PROJECT?
Project characteristics

- Temporary
- Unique products, services, or results
- Progressive elaboration

[PMBOK]
Project vs. Operational work

Common characteristics:
• Performed by people
• Constraint by limited resources
• Planned, executed, and controlled

Differ primarily in that operations are ongoing and repetitive, while projects are temporary and unique. [PMBOK]
WHAT IS PROJECT MANAGEMENT?
Definition

Project management is the application of knowledge, skills, tools and techniques to project activities to meet project requirements. Project management is accomplished through the application and integration of the project management processes of initiating, planning, executing, monitoring and controlling, and closing. [PMBOK]
Triple constraint

- Project scope
- Time
- Cost

Project quality is affected by balancing these three factors.

[PMBOK]
PM process groups

- Initiating process group
- Planning process group
- Executing process group
- Monitoring and controlling process group
- Closing process group

[PMBOK]
PM knowledge areas

• Project **Scope** Management
• Project **Time** Management
• Project **Cost** Management
• Project **Quality** Management
• Project **Human Resource** Management
• Project **Communications** Management
• Project **Risk** Management
• Project **Procurement** Management [PMBOK]
Management scope

• People
• Product
• Process
• Project
People

• **Stakeholders**: senior managers, project (technical) managers, practitioners, customers, end-users, ...

• **Team Leaders**: Motivation, Organization, Ideas and innovation. Problem solving, Managerial identity, Achievement, Influence and team building.

• **Software Team**: Organizational paradigms:
  – Closed
  – Random
  – Open
  – Synchronous
Product

• Software Scope:
  – Context
  – Information objectives
  – Function and performance

• Problem Decomposition
Process

- Melding the Product and the Process
- Process Decomposition
Project

Five-part common-sense approach:
• Start on the right foot.
• Maintain momentum.
• Track progress.
• Make smart decisions.
• Conduct a postmortem analysis.
W\textsuperscript{5}HH principle

- Why is the system being developed?
- What will be done?
- When will it be done?
- Who is responsible for a function?
- Where are they organizationally located?
- How will the job be done technically and managerially?
- How much of each resource is needed?
Critical practices

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

- *Measure* provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process.
- *Measurement* is the act of determining a measure.
- *Metric* is a quantitative measure of the degree to which a system, component, or process possesses a given attribute.
- *Indicator* is a metric or combination of metrics that provides insight into the process, a project, or the product itself.
“Good” metrics

• Simple and computable
• Empirically and intuitively persuasive
• Consistent and objective
• Consistent in the use of units and dimensions
• Programming language independent
• An effective mechanism for high-quality feedback.
Metrics

• Process metrics
• Product metrics

• Project metrics enable software manager to:
  – assess the status of an ongoing project,
  – track potential risks,
  – uncover problem areas before they go “critical”,
  – adjust work flow or tasks,
  – evaluate the project team’s ability to control quality of software work products.
ISO 9126 quality attributes

- **Functionality**: suitability, accuracy, interoperability, compliance, and security.
- **Reliability**: maturity, fault tolerance, recoverability.
- **Usability**: understandability, learnability, operability,
- **Efficiency**: time behavior, resource behavior.
- **Maintainability**: analyzability, changeability, stability.
- **Portability**: adaptability, installability, conformance, replaceability.
Software metrics

- **Size-oriented** (e.g. LOC)
- **Function-oriented** (e.g. FP)
- **Object-oriented** (e.g. number of scenario scripts, key classes, support classes, subsystems)
- **Use-case oriented**
FP metrics. Information Domain Values

Number of

- External Inputs (EIs)
- External Outputs (EOs)
- External Inquiries (EQs)
- Internal Logical Files (ILFs)
- External Interface Files (EIFs)
FP metrics. Functional Complexity

Weighting factors

<table>
<thead>
<tr>
<th>Information Domain Value</th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIs</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>EOs</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>EQs</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>ILFs</td>
<td>7</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>EIFs</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>
FP metrics. Unadjusted FPs

Unadjusted FPs =

EIs * Wf +
EOs * Wf +
EQs * Wf +
ILFs * Wf +
EIFs * Wf
FP metrics. Value Adjustment Factors

1. Data Communications. The data and control information used in the application are sent or received over communication facilities.

2. Distributed Data Processing. Distributed data or processing functions are a characteristic of the application within the application boundary.

3. Performance. Application performance objectives, stated or approved by the user, in either response or throughput, influence (or will influence) the design, development, installation and support of the application.

4. Heavily Used Configuration. A heavily used operational configuration, requiring special design considerations, is a characteristic of the application.

5. Transaction Rate. The transaction rate is high and influences the design, development, installation and support.
6. **On-line Data Entry.** On-line data entry and control information functions are provided in the application.

7. **End -User Efficiency.** The on-line functions provided emphasize a design for end-user efficiency.

8. **On-line Update.** The application provides on-line update for the internal logical files.

9. **Complex Processing.** Complex processing is a characteristic of the application.

10. **Reusability.** The application and the code in the application have been specifically designed, developed and supported to be usable in other applications.
FP metrics. Value Adjustment Factors (3)

11. Installation Ease. Conversion and installation ease are characteristics of the application. A conversion and installation plan and/or conversion tools were provided and tested during the system test phase.

12. Operational Ease. Operational ease is a characteristic of the application. Effective start-up, backup and recovery procedures were provided and tested during the system test phase.

13. Multiple Sites. The application has been specifically designed, developed and supported to be installed at multiple sites for multiple organizations.

14. Facilitate Change. The application has been specifically designed, developed and supported to facilitate change.
FP metrics. Value Adjustment Factors (4)

For each value adjustment factor (F), give a rating on a scale of 0 to 5

0 - No influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 - Essential
FP metrics. Adjusted FPs

Adjusted FPs =

Unadjusted FPs * [0.65 + 0.01 * \( \sum F_i \)]

where \( F_i \) are Value Adjustment Factors.
Defect Removal Efficiency

\[
\text{DRE} = \frac{E}{E + D}
\]

where \( E \) is the number of errors found before the delivery of the software to the end-user, and \( D \) is the number of defects found after delivery.
Estimation options

• Delay estimation until late in the project (obviously we can achieve 100% accurate estimates after the project is complete).
• Base estimates on similar projects that have already been completed.
• Use relatively simple decomposition techniques to generate project cost and efforts estimates.
• Use one or more empirical models for software cost end efforts estimation.
Software Sizing

- LOCs
- FPs
- “Fuzzy logic”
- Standard component
- Change sizing
Expected-value for the size

\[ S = \frac{(S_{opt} + 4S_m + S_{pess})}{6} \]

where

- \( S_{opt} \) is optimistic,
- \( S_m \) is most likely, and
- \( S_{pess} \) is pessimistic estimation.
Proxy-based estimating

• PROxy-Based Estimating (PROBE) is an estimating process used in the Personal Software Process (PSP) to estimate size and effort.

• PROBE is based on the idea that if an engineer is building a component similar to one he built previously, then it will take about the same effort as it did in the past.
Individual engineers use a “database” to keep track of the size and effort of all of the work that they do, developing a history of the effort they have put into their past projects, broken into individual components. Each component in the database is assigned a type (“calculation,” “data,” “logic,” etc.) and a size (from “very small” to “very large”). When a new project must be estimated, it is broken down into tasks that correspond to these types and sizes. A formula based on linear regression is used to calculate the estimate for each task.
Process-based estimation

• Project’s estimation is based on the process that will be used.
• Delineation of software functions obtained from the project scope.
• Estimations for each of framework activities must be performed for each function.
Empirical estimation models

Wallston-Felix model:
\[ E = 5.2 \times \text{KLOC}^{0.91} \]

Boehm simple model:
\[ E = 3.2 \times \text{KLOC}^{1.05} \]

Bailey-Basili model:
\[ E = 5.5 + 0.73 \times \text{KLOC}^{1.16} \]

Albrecht and Gaffney model:
\[ E = -91.4 + 0.355 \times \text{FP} \]

Kemerer model:
\[ E = -37 + 0.96 \times \text{FP} \]

where \( E \) is effort in person-month.
COCOMO models

• CONstructive COst Models introduced by Barry Boehm.

• Address the following areas:
  – Application composition model
  – Early stage design model
  – Post-architecture stage model

• First, sizing information (like FP). Then adjustments based on project, team, and customer specific.
Software Equation

\[ E = \left[ \text{LOC} \times B^{0.333}/ P \right]^3 \times \left(1 / t^4\right) \]

where

- \( E \) - effort in person-months or person-years,
- \( t \) – project duration in months or years,
- \( B \) – special skills factor (increases slowly as the need for integration, testing, quality assurance, documentation, and management skills grows; for small (5-15 KLOC) programs \( B=0.16 \), for programs >70 KLOC \( B=0.39 \)),
- \( P \) – productivity parameter (reflects overall process maturity and management practices, the level of programming language used, the software environment, the skills and experience of the software team, and the complexity of the application).
Reconciling estimates

• The various estimation methods encountered result in multiple estimates which must be reconciled.

• The goal of a reconciliation process is to produce a single estimate of effort, project duration, or cost.

• “Complicated methods might not yield a more accurate estimate, particularly when developers can incorporate their own intuition into the estimate.” -- Philip Johnson, et al.
Make/buy decision

Expected cost\(_{Path}\) = \(\sum\) (Path probability\(_i\)) \(\times\) (Path cost\(_i\))
PROJECT SCHEDULING
Causes of delivery late

• Changing customer requirements that are not reflected in schedule changes.
• An unrealistic deadline established by someone outside the software engineering group.
• An honest underestimate of effort/resources.
• Risks that were not considered.
• Technical difficulties. Human difficulties.
• Miscommunication among project staff.
• A failure by project management to recognize that the project is failing behind the schedule and a lack of action to correct the problem.
Software project scheduling

• Creating a network of software engineering tasks enabling you to get the job done on time

• Responsibility is assigned for each task, progress is tracked, and the network is adapted to changes in the process as they occur

• “I love deadlines. I like the whooshing sound they make as they fly by.” - Douglas Adams
Basic principles

• Compartmentalization
• Interdependency
• Time allocation
• Effort validation
• Defined responsibilities
• Defined outcomes
• Defined milestones
Relationship between people and effort

• Common myth still believed by many software managers:
  “If we fall behind schedule we can always add more programmers to catch up.”

• Smith’s law: Adding more people to a late software project always makes it later.
Elasticity of project schedules

Putnam-Norden-Rayleight (PNR) curve

\[ E_a = m \left( \frac{t_d^4}{t_a^4} \right) \]

Where
- \( E_a \) = Effort in person months
- \( t_d \) = The nominal delivery time for the schedule
- \( t_a \) = Actual delivery time desired
Effort distribution

40-20-40 rule (recommended distribution):
• 40% of all effort is allocated to front-end analysis and design
• 20% is for coding
• 40% is applied to back-end testing
Defining a set of tasks

- Depends on:
  - type of the project (concept development, new application development, application enhancement, application maintenance, reengineering)
  - software development methodology
  - life-cycle model
  - size of the project
  - mission criticality
  - project staff and etc.
Defining a task network

Critical path: the chain of tasks that determines the duration of the project.
Scheduling

- Critical path method (CPM)
- Program evaluation and review technique (PERT): when not knowing precisely the details and durations of all the activities
- Resource constraints
Work breakdown structure (WBS)

• WBS is a deliverable oriented decomposition of a project into smaller components. It defines and groups a project's discrete work elements in a way that helps organize and define the total work scope of the project.
• Gantt charts have become a common technique for representing the phases and activities of a project WBS.
• A Gantt chart is a type of bar chart that illustrates a project schedule (illustrate the start and finish dates of the terminal elements and summary elements of a project).
Gantt chart example

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Predecessors</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start</td>
<td></td>
<td>0 days</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>1</td>
<td>4 days</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>1</td>
<td>5.33 days</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>2</td>
<td>5.17 days</td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>2</td>
<td>6.33 days</td>
</tr>
<tr>
<td>6</td>
<td>e</td>
<td>3, 4</td>
<td>5.17 days</td>
</tr>
<tr>
<td>7</td>
<td>f</td>
<td>5</td>
<td>4.5 days</td>
</tr>
<tr>
<td>8</td>
<td>g</td>
<td>6</td>
<td>5.17 days</td>
</tr>
<tr>
<td>9</td>
<td>Finish</td>
<td>7, 8</td>
<td>0 days</td>
</tr>
</tbody>
</table>
Tracking project progress

• Conducting periodic project status meetings in which each team member reports progress and problems.
• Evaluating the results of all reviews conducted.
• Determining whether formal milestones have been accomplished.
• Comparing actual start-date to planned start-date for each project task.
• Meeting informally with practitioners to obtain their subjective assessment of the progress to date and problems on the horizon.
• Using earned value analysis.
Earned value method. Definitions

- **Budgeted cost of work schedule (BCWS)**: estimated value for each task represented in schedule (in person-months).
- **Budget at completion (BAC)**: sum of BCWS for all tasks.
- **Budgeted cost of work performed (BCWP)**: sum of the BCWS values for **completed** tasks.
- **Actual Cost of Work Performed (ACWP)**: cost incurred to accomplish the task.
- **Planned value (PV)**: \( \text{BCWS} / \text{BAC} \times 100\% \).
- **Earned value (EV)**: \( \text{BCWP} / \text{BAC} \times 100\% \).
Earned value method. Derived metrics

- **Schedule Variance:**
  \[ SV = BCWP - BCWS \]

- **Cost variance:**
  \[ CV = BCWP - ACWP \]

- **Schedule Performance Index:**
  \[ SPI = \frac{BCWP}{BCWS} \]

- **Cost Performance Index:**
  \[ CPI = \frac{BCWP}{ACWP} \]

- **Cost Schedule Index:**
  \[ CSI = CPI \times SPI \]
RISK MANAGEMENT
Definition

Project risk is an **uncertain** event or condition that, if occurs, has a **positive** or negative **effect** on at least one project objective, such as time, cost, scope, or quality.

A risk may have one or more causes and, if it occurs, one or more impacts.

[PMBOK]
Risk management strategies

• Reactive
• Proactive
Categories of risks

• Project risks
• Technical risks
• Business risks
• Known risks
• Predictable risks
• Unpredictable risks
• Generic risks
• Product specific risks
Risk identification checklist

• Product size
• Business impact
• Customer characteristics
• Process definition
• Development environment
• Technology to be built
• Staff size and experience
Risk components

Depending on what software/project characteristic it affects:

• Performance risk
• Cost risk
• Support risk
• Schedule risk
Risk impact categories

Qualitative:
• Negligible
• Marginal
• Critical
• Catastrophic

Quantitative approach is estimate the impact as the cost to the project should the risk occur.
Risk exposure = Probability * Cost
Assessing risk example

- Risk identification. Only 70% of software components scheduled for reuse will, in fact, be used. Other will have to be developed.
- Risk probability. 80%
- Risk impact. 60 reusable components are planned. 18 components would have to be developed from scratch. Average component is 100 LOC. Cost for LOC is $14. The overall cost (impact) would be $25,200 = 18 \times 100 \times 14$
- Risk exposure. $RE = 0.80 \times 25,200 = $20,200$
Risk table

Columns:
• Risk
• Category
• Probability
• Impact
• RMMM (Risk Mitigation, Monitoring, and Management)

Sorted by risk exposure.
Strategy for dealing with risk

Must consider three issues:

• Risk avoidance (mitigation).
• Risk monitoring.
• Risk management and contingency planning.
High staff turnover risk mitigation

- Meet with current staff to determine causes.
- Mitigate those causes that are under control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.
- Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner.
- Conduct peer reviews of all work products.
- Assign a backup staff member for every critical technologist.
RMMM plan

Risk information sheet includes for each risk:

• Risk ID, Date, Probability & Impact
• Description
• Refinement/context
• Mitigation/monitoring
• Management/contingency plan/trigger
• Current status
• Responsible
SOFTWARE QUALITY ASSURANCE
Definitions

- Quality is a characteristic or attribute of something. [American Heritage Dictionary]
- A product’s quality is a function of how much it changes the world for the better. [DeMarco]
- Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software. [Pressman]
IEEE definitions of Software quality

• The degree to which a system, component, or process meets specified requirements.
• The degree to which a system, component, or process meets customer or user needs or expectations.
ISO definition of Software quality

- The totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs. [ISO 9126]
Definitions of Quality assurance

• A planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements. [IEEE Std 730-1998]

• All the planned and systematic activities implemented within the quality system, and demonstrated as needed, to provide adequate confidence that an entity will fulfill requirements for quality. [IEEE/EIA 12207.0-1996]
More definitions

- software bug
- software error
- software defect
- software fault
- software failure
Causes of errors

- faulty requirements definition
- client-developer communication failures
- deliberate deviations from software requirements
- logical design errors
- coding errors
- non-compliance with documentation and coding instructions
- shortcomings of the testing process
- procedure errors
- documentation errors
Cost of errors

"Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion annually, or about 0.6 percent of the gross domestic product. ...

Although all errors cannot be removed, more than a third of these costs, or an estimated $22.2 billion, could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects. These are the savings associated with finding an increased percentage (but not 100 percent) of errors closer to the development stages in which they are introduced. Currently, over half of all errors are not found until "downstream" in the development process or during post-sale software use."

US Department of Commerce, June 2002
Cost of quality

• **Prevention costs** (quality planning, reviews, test equipment, and training)

• **Appraisal costs** (activities to gain insight into product condition the “first time through” each process)

• **Failure costs:**
  – Internal failure costs
  – External failure costs
SQA activities

Independent SQA group:

- 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 manager.
Software reviews

- (Author) review
- Peer review
- Formal technical review/inspection
Statistical SQA

• Six Sigma for Software Engineering.
  Core steps:
  – *Define* customer requirements, deliverables, and project goals.
  – *Measure* the existing process.
  – *Analyze* defect metrics and determine the vital few causes.
  Additional steps:
  – *Improve* the process be eliminating root causes of the defects.
  – *Control* the process.
Software reliability

• *Software reliability* is the probability of failure-free operation of a computer program in a specified environment for a specified time.

• *Software availability* is the probability that a program is operating according to requirements at a given point in time.

• *Software safety* is a SQA activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.
Measures

• Reliability measure:
  Mean-time-between-failure (MTBF)
  \[ MTBF = MTTF + MTTR \]
  where MTTF is mean-time-to-failure and MTTR is mean-time-to-repair.

• Availability:
  \[ \text{Availability} = \left[ \frac{MTTF}{MTTF + MTTR} \right] \times 100\% \]
ISO standards

Product quality:

• ISO 9126. Software engineering - Product quality.

Quality management (process quality):

• ISO 9000 (family). Quality management
SOFTWARE CONFIGURATION MANAGEMENT
Software configuration management

• Change management
• Version control
• Release management
Baseline

• A specification or product that has been formally reviewed and agreed upon, that thereafter serves as basis for further development, and that can be changed only through formal change control procedure.

[IEEE Std. No. 610.12-1990]
SCM features

• Versioning
• Dependency tracking and change management
• Requirements tracing
• Configuration management
• Audit trails
Layers of SCM process

- Software Configuration Items
- Identification
- Change control
- Version control
- Configuration auditing
- Reporting
PROJECT MANAGEMENT TOOLS
Software tools examples

Software tools examples (2)

• Kovair Global Lifecycle.

Software tools examples (3)

• AtTask Inc. (@task).

Rich collection of tools at Ganththead.com (http://www.gantthead.com/tools/)
What we have learned?

- Concepts of project and project management
- Software metrics
- Estimation
- Scheduling
- Risk management
- Quality assurance
- Configuration management
- Tools
QUESTIONS?
Software Engineering practice

- Understand the problem (communication and analysis)
- Plan a solution (modeling and software design)
- Carry out the plan (code generation)
- Examine the result for accuracy (testing and quality assurance)
Core principles

1. The Reason It All Exists
2. KISS (Keep It Simple, Stupid!)
3. Maintain the Vision
4. What You Produce, Others Will Consume
5. Be Open to the Future
6. Plan Ahead for Reuse
7. Think!
Communication principles

• Listen.
• Prepare before you communicate.
• Someone should facilitate the activity.
• Face-to-face communication is best.
• Take notes and document decisions.
• Strive for collaboration.
• Stay focused, modularize your discussion.
Communication principles (2)

• If something is unclear, draw a picture.
• (a) Once you agree to something, move on;
  (b) If you can’t agree to something, move on;
  (c) If a feature or function is unclear and cannot be clarified at the moment, move on.
• Negotiation is not a contest or a game. It works best when both parties win.
Planning principles

• Understand the scope of the project.
• Involve the customer in the planning activity.
• Recognize that planning is iterative.
• Estimate based on what you know.
• Consider risk as you define the plan.
Planning principles (2)

- Be realistic.
- Adjust granularity as you define the plan.
- Define how you intend to ensure quality.
- Describe how you intend to accommodate change.
- Track the plan frequently and make adjustments as required.
Analysis modeling principles

• The information domain of a problem must be represented and understood.
• The functions that the software performs must be defined.
• The behavior of the software (as a consequence of external events) must be represented.
• The models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion.
• The analysis task should move from essential information toward implementation detail.
Design modeling principles

• Design should be traceable to the analysis model.
• Always consider the architecture of the system to be built.
• Design of data is as important as design of processing functions.
• Interfaces (both internal and external) must be design with care.
• User interface design should be tuned to the needs of the end-user.
Design modeling principles (2)

- Component-level design should be functionally independent.
- Components should be loosely coupled to another and to the external environment.
- Design representations (models) should be easily understandable.
- The design should be developed iteratively. With each iteration, the designer should strive for greater simplicity.
Coding. Preparation principles

Before you write one line of code, be sure you:

• Understand the problem you’re trying to solve.
• Understand basic design principles and concepts.
• Pick a programming language that meets the needs of the software to be built and the environment in which it operate.
• Select a programming environment that provides tools that will make your work easier.
• Create a set of unit tests that will be applied once the component you code is completed.
Coding. Coding principles

As you begin writing code, be sure you:

- Constraint your algorithms by followed programming practice.
- Select data structures that will meet the needs of the design.
- Understand the software architecture and create interfaces that are consistent with it.
- Keep conditional logic as simple as possible.
- Create nested loops in a way that makes them easily testable.
- Select meaningful variable names and follow other local coding standards.
- Write code that is self-documenting.
- Create a visual layout (e.g., indentation and blank lines) that aids understanding.
After you’ve completed your first coding pass, be sure you:

• Conduct a code walkthrough when appropriate.
• Perform unit tests and correct errors you’ve uncovered.
• Refactor the code.
Testing principles

• All tests should be traceable to customer requirements.
• Tests should be planned long before testing begins.
• The Pareto principle applies to software testing (80-20).
• Testing should begin “in the small” and progress toward testing “in the large”.
• Exhaustive testing is not possible.
Deployment principles

• Customer expectations for the software must be managed.
• A complete delivery package should be assembled and tested.
• A support regime must be established before the software is delivered.
• Appropriate instructional materials must be provided to end-users.
• Buggy software should be fixed first, delivered later.
PMBOK classifies processes by

- Process Groups
- Knowledge Areas

Further slides provide both these classifications.
Initiating processes

- Develop Project Charter
- Develop Preliminary Project Scope Statement
Planning processes

- Develop Project Management Plan
- Scope Planning
- Scope Definition
- Create WBS
- Activity Definition
- Activity Sequencing
- Activity Resource Estimating
- Activity Duration Estimation
- Schedule Development
- Cost Estimating
Planning processes (2)

- Cost Budgeting
- Quality Planning
- Human Resource Planning
- Communication Planning
- Risk Management Planning
- Risk Identification
- Qualitative Risk Analysis
- Quantitative Risk Analysis
- Risk Response Planning
- Plan Purchases and Acquisitions
- Plan Contracting
Executing processes

- Direct and Manage Project Execution
- Perform Quality Assurance
- Acquire Project Team
- Develop Project Team
- Information Distribution
- Request Seller Responses
- Select Sellers
Monitoring and controlling processes

• Monitor and Control Project Work
• Integrated Change Control
• Scope Verification
• Scope Control
• Schedule Control
• Cost Control
• Perform Quality Control
• Manage Project Team
• Perform Reporting
• Manage Stakeholders
• Risk Monitoring and Control
• Contract Administration
Closing processes

• Close Project
• Contract Closure
Integration KA processes

- Develop Project Charter
- Develop Preliminary Project Scope Statement
- Develop Project Management Plan
- Direct and Manage Project Execution
- Monitor and Control Project Work
- Integrated Change Control
- Close Project
Scope KA processes

- Scope Planning
- Scope Definition
- Create WBS
- Scope Verification
- Scope Control
Time KA processes

- Activity Definition
- Activity Sequencing
- Activity Resource Estimating
- Activity Duration Estimation
- Schedule Development
- Schedule Control
Cost KA processes

- Cost Estimating
- Cost Budgeting
- Cost Control
Quality KA processes

• Quality Planning
• Perform Quality Assurance
• Perform Quality Control
Human Resource KA processes

- Human Resource Planning
- Acquire Project Team
- Develop Project Team
- Manage Project Team
Communications KA processes

• Communications Planning
• Information Distribution
• Performance Reporting
• Manage Stakeholders
Risk KA processes

- Risk Management Planning
- Risk Identification
- Qualitative Risk Analysis
- Quantitative Risk Analysis
- Risk Response Planning
- Plan Purchases and Acquisitions
- Risk Monitoring and Control
Procurement KA processes

- Plan Purchases and Acquisitions
- Plan Contracting
- Request Seller Responses
- Select Sellers
- Contract Administration
- Contract Closure