Chapter 5
Software Process Management

© Pearson Education Limited 2005

Topics
- People management
- Risk management
- Quality management
- Change and configuration management

The theme

People management
- Reasoning is like that:
  - technology is changing all aspects of business,
  - the changing business environment impacts organizational structures resulting in network organizations of loosely connected cooperating units,
  - new organizational structures imply new management styles,
  - new management styles must be risk tolerant and innovative,
  - risk tolerance and innovation does not square well with a bureaucratic people management based on accountability and procedures,
  - modern work teams draw strength from opportunistic collaboration of people, where individuals do not automatically owe any loyalty to the organization but are motivated by the working environment and more abstract work-unrelated needs.

Acquiring and motivating people
- The essence of the team-building process
- Staffing management plan – part of the project plan
- Self-directed teams – favored by modern network organizations
- Leaders and managers
  - project manager must also be a leader (in particular on large projects)
  - team members must be inspired, as well as motivated

Team creation
- By reassignment of employees between projects
  - suitable, potentially available and interested employees
  - not necessarily the best employees
- By hiring new employees
  - better possibility of finding right people
  - but at a greater risk of misjudging the applicants
- The tests to assess applicants
  - aptitude tests to uncover person’s skills to perform certain tasks
  - psychometric tests uncover person’s attitude and suitability
- Software engineers tend to be cash-rich and time-poor
Motivational theories

- **Motivation** can be
  - extrinsic or
  - intrinsic

Theories (discussed next):
- Maslow’s Needs Hierarchy
- Hygiene Theory
- Expectancy Theory
- Achievement Theory

Maslow’s needs hierarchy

- Five groups of motivational needs:
  1. physiological needs for food, shelter, etc.,
  2. safety needs for physical welfare and the security of possessions,
  3. social needs for interaction with other people, sense of belonging, love, friendship, etc.,
  4. esteem needs for recognition, respect, accomplishment, etc., and
  5. self-realization needs for personal expression and development.

- Once a lower level need has been met and is continuously maintained, it does not serve as a motivator any more \(\rightarrow\) the next higher level becomes the new motivator.

Herzberg’s Hygiene Theory

- Known also as the Motivation-Hygiene Theory
- Identifies two motivational factors:
  1. hygiene factors
     - work conditions, personal relationships, pay, benefits, and similar work provisions
  2. motivators
     - aspects of the work that bring satisfaction from the job well done
     - opportunities for personal advancement, training programmes, challenging but feasible tasks, etc.

Expectancy Theory

- States that people act and behave according to expectations expressed by their environment, provided these expectations bring rewards
- The desire to deliver an outcome that satisfies expectations drives motivation
- To be true motivators, any rewards promised must always be fulfilled
- Managers should be
  - setting high expectations regarding performance
  - openly rewarding high performance

Achievement Theory

- Three motivational factors:
  1. achievement (desire to succeed)
  2. power (desire to influence others)
  3. affiliation (desire of belonging and companionship)
- The theory does not differentiate between these three motivators
- It states that the strength for each of these desires determines the blend responsible for the performance of the team

Project communications

- Communication is the activity of exchanging information and knowledge
- the process of sending messages from senders to recipients
- Software process management addresses project communications from at least four viewpoints (discussed next):
  - forms of communication
  - lines of communication
  - facilitating/inhibiting factors in communication
  - communication in conflict resolution
Forms of communication

- Messages are exchanged in some encoded form
  - verbal, in text, in pictures, in symbols, conveyed by facial expression or intonation, etc.
  - message can be confidential, public, formal, informal, internal, external, horizontal (between peers), vertical (between supervisors and subordinates), etc.

- Typical forms of communication are:
  - person-to-person (casual, in-room meetings, telephone conversations, Internet “conferencing”)
  - telephone voice mail
  - faxes
  - post mail
  - courier mail
  - electronic mail
  - web pages

Factors in communication

- It is never necessary for every single piece of information to be communicated to all people on the project
  - inform about things that affect or relate to the tasks performed
  - inform about global issues, project directions, selected political decisions, etc.
- Danger with hierarchical structures and vertical messages is that people at the same level do not communicate sufficiently
  - select team members to ensure a proper mix of extroverts and introverts
- Significant amount of communication is incidental

Organized forms of group communication  ➔ formal and informal meetings

Communication lines

- n people = n(n-1)/2 lines of communication
- 3 people = 3 communication lines, 7 possible teams
- 5 people = 10 communication lines, 31 possible teams

Communication in conflict resolution

- Lasting effect:
  - forcing  ➔ possible under supervisor-subordinate dependency
  - compromise  ➔ requires face-to-face communication
  - confrontation  ➔ most effective and most desirable solution, provided one right answer to the conflict exists  ➔ problem solving
- Temporary outcomes:
  - smoothing  ➔ the problem underpinning the conflict is shown as less significant than it is = “brain washing”
  - withdrawal  ➔ one party sees no point in discussing the conflict any further

Team development

- Issues in team development (team building):
  - acquiring and motivating people (as discussed)
  - project communications (as discussed)
  - team organization, collocation, training, appraisals, performance reports, external feedback, etc. (not discussed in detail here)
- Teams undergo four stages of development:
  - forming
    - setting up of the team
  - storming
    - resolving confrontation and conflicts
    - leaders are born
  - norming
    - pushing and pulling gives way to collaboration
    - friendships are born
  - performing
    - reached only by great teams

Risk management

- Project may be much more advanced in terms of schedule and budget than in terms of risk issues (e.g. project may be 80% complete but its risk of failing may be as high as 50%)
- Reactive risk strategies:
  - delay handling risks until they become critical factors in the project
- Proactive risk strategies:
  - encourage acting on risks as soon as they arise or even in anticipation of them happening
- Process of risk management (discussed next):
  - risk identification
  - risk assessment
  - risk handling
Risk identification

- Typical risk types:
  - project product
    - risk due to project size, uniqueness, complexity, etc.
  - project process
    - due to changing or not following the process
  - people
    - due to personality clashes, poor management, human resource allocations, etc.
  - business
    - due to changes in business conditions, policies, etc.
  - organizational
    - due to restructuring, financial difficulties, etc.
  - technology
    - due to technology incompatibility or technology changes
  - tools
    - due to incapability of software engineering tools, takeovers of tool vendors, etc.
  - facilities
    - due to communication difficulties, lack of interest in the project, etc.
  - requirements
    - due to changing, incomplete, ill-defined requirements, etc.
  - other projects
    - due to dependency on and impact of other projects
  - external factors
    - due to technology, regulations, legal implications, etc.

Risk assessment

- Called also risk analysis, risk estimation or risk projection
- It ranks the identified risks in two ways:
  - by the likelihood or probability of the risk happening
  - by the negative impact on the project
- Likelihood can use five-point scale, such as very likely, quite possible, possible, probable, unlikely
- Probability can be used on a nine-point scale, i.e. 10%, 20%, etc. up to 90%
- Risk with 100% probability is not a risk
- A certainty – a project risk cannot occur
- It is a certainty – a project will not be finished within the planned time
- It is a certainty – a project will be under budget
- Risk with 100% probability is not a risk

Risk management plan:
- document that defines risk management activities (part of project management plan)
- risk database

Risk handling strategies:
- risk avoidance
  - to eliminate the likelihood/probability of the risk occurring
- risk minimization
  - to diminish the impact of the risk when it occurs
- contingency planning
  - to know how to react when the risk happens
- Risk exposure and risk management cutoff
- Risk referent levels and project termination points
- Risk handling
Quality management

- Software quality management is an umbrella activity that intertwines with most other management undertakings and cuts across the entire development lifecycle and process.
- Quality management team should be separate from the development team and have its own reporting channels.
- Quality management plan should have its own budget and schedule.
- Quality standards for assuring quality in the products → quality assurance
  - CMM (discussed before)
  - ISO 9001:2000 – the latest ISO standard for managing key processes to continually improve them
- Controlling quality in the products → quality control

Software qualities

- correctness
- reliability
- robustness
- performance
- usability
- understandability
- maintainability (repair ability)
- scalability (evolvability)
- reusability
- portability
- interoperability
- productivity
- timeliness
- visibility

Correctness, reliability, robustness, performance

- Correctness
  - defined by the correspondence between the software product and its functional specifications
  - it is not possible to prove a program correct
  - possible to have a correct software for incorrect requirements
- Reliability (called also dependability)
  - software that behaves well and in the way users expect
  - in software engineering a reliable software is allowed to contain “known bugs”
- Robustness
  - software that is unlikely to fail or break in an unrecoverable way
- Performance
  - software that operates satisfactorily, according to predefined targets
  - usual target is response time of the software system
  - black-and-white quality, either the system meets the performance indicator or not

Usability, understandability, maintainability, scalability

- Usability
  - user friendliness of the software
  - predominantly a feature of the design of the user interface for the system
- Understandability
  - ease with which the meaning of the internal structure and behavior of the software can be analyzed and comprehended by a software maintainer
- Maintainability (repairability)
  - ability to correct software errors and deficiencies (a narrow meaning of the concept)
- Scalability (evolvability)
  - ease with which the software can be scaled up or evolved in response to the growth in demand for its functionality
- Understandability, maintainability and scalability are known under a combined name of supportability

Reusability, portability, interoperability, productivity, timeliness, visibility

- Reusability
  - level to which the software components can be used for construction of other products
  - reused as a component or as a generic framework
- Portability
  - software that can run on various hardware/software platforms without any modifications or after undergoing minor customization or parameterization
- Interoperability
  - ability of software to coexist or work together with other software, possibly even with the future software that does not exist yet → open systems
- Productivity
  - rate at which the process allows to produce software given certain amount of resources
- Timeliness
  - ability of the process to produce software on time
- Visibility
  - process with clearly defined and documented stages and activities

Quality control

- Quality control (variation control) → testing quality of a product
  - reactive concept
  - operational and tactical effort
- Quality assurance → building quality into the product
  - proactive undertaking
  - strong strategic aspect
- Sometimes, quality control is considered a part of quality assurance
- Quality control is about finding variations in a work or end product compared to a given specification → software testing
**Relative cost of correcting an error**

<table>
<thead>
<tr>
<th>Software lifecycle phase in which an error has been uncovered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis</td>
</tr>
<tr>
<td>Relative cost of correcting an error</td>
</tr>
</tbody>
</table>

Relative costs:
- Requirements Analysis: 1
- System Design: 10
- Implementation: 100
- Integration and Deployment: 1000
- Operation and Maintenance: 10000

**Testing techniques**

- **Partitioning**
  - Equivalence partitioning
  - Boundary values

- **Visibility**
  - Black Box
  - White Box

- **Coverage**
  - Operation coverage
  - Path coverage

- **Automation**
  - Manual
  - Automatic
  - Regression
  - Exercising

- **Scripting**

**Equivalence partitioning and boundary value**

- **Equivalence partitioning**
  - groups data inputs (and, implicitly, data outputs) into partitions constituting homogeneous test targets
  - the assumption is that testing with any one member of the partition is as good as testing with other members

- **Boundary value**
  - additional data analysis technique to assist in equivalence partitioning and, consequently, to assists in black box testing
  - boundary values are extreme cases within equivalence partitions

**Operation and path coverage**

- **Coverage**
  - techniques determine how much code is going to be exercised by a white box test

- **Operation coverage (method coverage)**
  - ensures that each operation in the code is exercised at least once by the white box test
  - modern object-oriented substitute for statement and branch coverage, which apply in procedural programming languages

- **Path coverage**
  - aims at numbering possible execution paths in the program and exercising them one by one
  - the number of such paths is indefinite in large programs
**Manual and automated testing**

- **Manual testing**
  - human tester conducts the tests
  - test script defines step-by-step testing actions and expected outcomes
  - use cases and other requirements specification documents are used to write test scripts
  - frequently performed on “live data”
    - expected output is not always defined precisely in the test scripts
    - sometimes, the output is not even presented to the screen, but it is manifested in database changes

- **Automated testing**
  - employs software testing tools to execute large volumes of tests without human participation
  - the tools produce necessary post-test reports to facilitate management of test outcomes

**Regression and exercising testing**

- Automated testing can be divided into regression testing and exercising testing
  - **Regression testing**
    - repetitive execution of the same test scripts on the same baseline data to verify if previously accepted functionality of the system has not been broken by successive changes to the code
    - performed by automatic execution of pre-recorded test scripts
  - **Exercising**
    - an automated coverage testing
    - a tool generates automatically and randomly various possible actions
      - the best script
    - the defect script
      - script generated for actions that led to a problem
      - it can be played back at any time to reproduce the error and try to determine the reason behind it.

**Testing concepts**

- Test Suite
  - Test Plan
  - Test Case
  - Test Input
  - Test Output
  - Verification Points
  - Configurations (where)
  - Iterations (when)

- Testing Process
  - Test Plan
  - Test Case
  - Test Input
  - Test Output
  - Verification Points

- Electronic Test Case
  - Virtual Test

- Local Computer
  - Agent Computer
  - Human Tester

**Quality assurance**

- A management-level function
  - **Software Quality Assurance (SQA)**
    - independent from developers and reporting to functional management
    - monitors not just development product and processes, but also project plans – schedule, budget, allocation and utilization of resources
  - Verifies the compliance of software products and processes with the adopted standards

- Main assurance undertakings:
  - checklists
  - reviews
  - audits

**Checklist**

- A list of issues and questions against which a software product or process is verified for quality properties

- It can be:
  - a simple list of relatively independent checkpoints (like a questionnaire)
  - a logical sequence of steps such that each next step feeds off the previous step

- Can be used as part of another quality assurance task, such as a review or audit
  - checklist questions need to be addressed to obtain a CMM certification or ISO registration
  - An important element of shaping the quality culture within an organization

**Review**

- A formal meeting of developers, and possibly managers, to review a work product or process
  - walkthrough
  - inspection (done under management supervision)

- Document-driven

- Restricted to a small number of people

- Acknowledged problems recorded in a review issues list

- Corrections and follow-up meetings
Audit

An IS/IT quality assurance process similar in scope, required resources, and level of assumed expertise to traditional accounting audits.

Starts with extensive preparations and studies of the audited system.

Continues with interviews and inspections.

Conclusions about the status of the process or product are documented in a formal report, which also includes a risk assessment for the continuation of the project.

The scope of an audit is normally much wider than other quality assurance undertakings:

- Involves the whole range of organizational plans and finances, not just the project plan.
- Considers legal implications and project management issues.

Change and configuration management

Requirements changes

Changes to requirements must be approved in a formal change request process:

- Change Request Form (CRF).
- Change Control Authority (CCA) or Change Control Board (CCB).
- Engineering Change Order (ECO).

Any changes to requirements necessitate corresponding changes in test requirements.

Artifact versions

Baselines and releases

A configuration in the public workspace can be registered as a current (latest) baseline:

- The baseline configuration is frozen for changes in the public workspace, but a new configuration can be promoted to a baseline.

From developers’ perspective, a baseline distributed to customers is called a system release (or deliverable):

- From the management perspective, a release is more than baseline software; it involves a significant number of products and processes:
  - Executable client code.
  - Database server code (schema, trigger, stored procedures).
  - Database load data (if any) and any other data files.
  - Configuration files and installation instructions.
  - System documentation and on-line help features.
  - Distribution options and associated informational and accounting documentation.
Defects

Enhancements

Metrics

- Collection of quantitative numbers to assess such subjective issues as maintainability, scalability, complexity or just the size of code or the size of effort
- Approximate, ambiguous and ballpark numbers are better than no numbers
- Have a side effect of evaluating people
- CK metrics (Chidamber and Kemerer):
  - Weighted methods per class (WMC)
  - Depth of inheritance tree (DIT)
  - Number of children (NOC)
  - Coupling between object classes (CBO)
  - Response for a Class (RFC)
  - Lack of cohesion in methods (LCOM)

Weighted methods per class

- Weighted methods per class (WMC)
  - the sum of complexities of all methods in a class
  - complexities are normalized values in the range from 0 to 1
- Viewpoints on the WMC metric:
  - WMC is a predictor metric (as opposed to a control metric). The metric can serve as a predictor of time and effort needed to develop and maintain a class.
  - A large value of WMC creates an increased potential inheritance dependency if the class is reused as a superclass.
  - A large value of WMC is a likely indicator that the class is application specific, thus limiting the possibility for its reuse across many applications.

Depth of inheritance tree

- Depth of inheritance tree (DIT)
  - the maximum length (number of classes) from the node to the root of the tree
  - applies per class, not per inheritance tree or per system
  - applies trivially to single inheritance, but it applies also to multiple inheritance
- Viewpoints on DIT:
  - A class with larger DIT inherits larger number of methods and it is, therefore, more difficult to understand its behavior.
  - A significant number of classes with large DIT imply greater design complexity in terms of overall number of classes and methods in the system.
  - A class with larger DIT is likely to be taking better advantage of reuse of inherited methods.

Number of children

- Like DIT, number of children (NOC) is inheritance-related metric
  - it counts the number of immediate subclasses of a class
  - a large value of NOC is rather negative property
- Viewpoints on NOC:
  - Larger NOC means more reuse via inheritance (which may be good or bad depending on the perspective taken).
  - Larger NOC implies usually an improper or diluted abstraction of the superclass and the misuse of subclassing.
  - A class with large NOC is likely to be doing too much (taking too much responsibility in the system). This creates an unbalanced design and makes testing more difficult.
Coupling between object classes

- Two classes are coupled if they depend on each other (but dependency, and therefore coupling, can be in one direction only)
- The dependency can be structural (via instance variables) or behavioral (due to method dependencies)
- Coupling between object classes (CBO) is a count of the number of classes coupled to a given class
  - necessary for classes to collaborate
  - excessive coupling has significant undesirable effects
- Viewpoints on CBO are:
  - Excessive CBO is detrimental to architectural design quality.
  - Large CBO for a class prevents its reuse because the class depends on other classes. (This is true for inheritance reuse, but it can be argued that coupling itself is a reuse.)
  - Large CBO hinders class encapsulation and makes maintenance of the class more troublesome.
  - Design with excessive CBO creates extra challenges for testing.

Response for a class

- **Response for a class** (RFC)
  - the number of methods which services are potentially needed in order to satisfy a message request received by an object of the class
  - a number of method invocations to perform a service
  - he methods can be internal to the class and/or external
- Viewpoints on RFC:
  - The testing and debugging of the class is more complex with larger value of RFC.
  - Large RFC implies a complex class.
  - Being the worst-case value, RFC can be a good indicator of time needed to perform testing.

Lack of cohesion in methods

- **Lack of cohesion in methods** (LCOM)
  - a high value of cohesion in methods says that the methods of the class are not related to each other (are not similar) and, therefore, they possibly do not belong to a single abstraction (to a single class)
  - the similarity may be measured by a number of methods accessing the same instance variables
- Viewpoints on LCOM:
  - Existence of cohesion in methods promotes encapsulation and is, therefore, desirable.
  - Lack of cohesion means that the methods are independent and the class should be split into two or more smaller classes.
  - Low LCOM increases complexity of the design and the potential for errors.

Summary

- The Capability Maturity Model (CMM) model defines five levels of process maturity
- Network organizations necessitate new approaches to people management
- Risk management involves three stages: (1) risk identification, (2) risk assessment, and (3) risk handling
- Quality management consists of quality control and quality assurance
- Change and configuration management is the process of managing product and process artifacts and managing the teamwork activities in an evolving software system