Chapter 22
Managing People
Managing people working as individuals and in group

Objectives
- To describe simple models of human cognition and their relevance for software managers
- To explain the key issues that determine the success or failure of team working
- To discuss the problems of selecting and retaining technical staff
- To introduce the people capability maturity model (P-CMM)

Topics covered
- Limits to thinking
- Group working
- Choosing and keeping people
- The people capability maturity model

Effective management
- Software project managers have to solve technical and non-technical problems by using the people in their team in the most effective way possible. They have to motivate the people, plan and organize their work and ensure that work is being done properly.
- Poor management of people is one of the most significant contributors to project failure.

People in the process
- People are an organization’s most important assets.
- The tasks of a manager are essentially people oriented. Unless there is a good understanding of people, management will be unsuccessful.
- Software engineering is primarily a cognitive activity. Cognitive limitations effectively limit the software process.

Activities in managing people
- Problem solving (using available people)
- Motivating (people who work on a project)
- Planning (what people are going to do)
- Estimating (how fast people will work)
- Controlling (people’s activities)
- Organizing (the way in which people work)
Limits to thinking

- People don’t all think the same way but everyone is subject to some basic constraints on their thinking due to:
  - Memory organization
  - Knowledge representation
  - Motivation influences
- If we understand these constraints, we can understand how they affect people participating in the software process.

Memory organization

- From senses
- Short-term memory
- Working memory
- Long-term memory

Short-term memory

- Fast access, limited capacity
- 5-7 locations
- Holds 'chunks' of information where the size of a chunk may vary depending on its familiarity
- Fast decay time

Working memory

- Larger capacity, longer access time
- Memory area used to integrate information from short-term memory and long-term memory
- Relatively fast decay time

Long-term memory

- Slow access, large capacity
- Unreliable retrieval mechanism
- Slow but finite decay time - information needs reinforced
- Relatively high threshold - work has to be done to get information into long-term memory

Information transfer

- Problem solving usually requires transfer between short-term memory and working memory
- Information may be lost or corrupted during this transfer
- Information processing occurs in the transfer from short-term to long-term memory
Cognitive chunking

Loop (process entire array)

Loop (process unsorted part of array)

- Compare adjacent elements
- Swap if necessary so that smaller comes first

Knowledge modelling

- **Semantic knowledge** knowledge of concepts such as the operation of assignment, concept of parameter passing etc.
- **Syntactic knowledge** knowledge of details of a representation e.g. an Ada while-loop.
- Semantic knowledge seems to be stored in a structured representation while syntactic knowledge is arbitrary and disorganized.

Knowledge acquisition

- Semantic knowledge through experience and active learning
- Syntactic knowledge acquired by memorization.
- New syntactic knowledge can interfere with existing syntactic knowledge.
  - Problems arise for experienced programmers in mixing up syntax of different programming languages

Semantic knowledge

- **Computing concepts** - notion of a writable store, iteration, concept of an object, etc.
- **Task concepts** - principally algorithmic - how to tackle a particular task
- Software development ability is the ability to integrate new knowledge with existing computer and task knowledge and hence derive creative problem solutions
- Thus, problem solving is language independent

Problem solving

- Requires the integration of different types of knowledge (computer, task, domain, organization)
- Development of a semantic model of the solution and testing of this model against the problem
- Representation of this model in an appropriate notation or programming language
Problem solving

Motivation

- An important role of a manager is to motivate the people working on a project.
- Motivation is a complex issue but it appears that there are different types of motivation based on
  - Basic needs (e.g. food, sleep, etc.)
  - Personal needs (e.g. respect, self-esteem)
  - Social needs (e.g. to be accepted as part of a group)

Motivating people

- Motivations depend on satisfying needs
- It can be assumed that physiological and safety needs are satisfied
- Social, esteem, and self-realization needs are most significant from a managerial viewpoint

Human needs hierarchy

Motivating people

Need satisfaction

- Social
  - Provide communal facilities
  - Allow informal communications
- Esteem
  - Recognition of achievements
  - Appropriate rewards
- Self-realization
  - Training - people want to learn more
  - Responsibility

Personality types

- The needs hierarchy is almost certainly an oversimplification
- Motivation should also take into account different personality types:
  - Task-oriented
  - Self-oriented
  - Interaction-oriented
Personality types

- **Task-oriented**
  - The motivation for doing the work is the work itself.

- **Self-oriented**
  - The work is a means to an end which is the achievement of individual goals - e.g. to get rich, to play tennis, to travel etc.

- **Interaction-oriented**
  - The principal motivation is the presence and actions of co-workers. People go to work because they like to go to work.

Motivation balance

- Individual motivations are made up of elements of each class.
- Balance can change depending on personal circumstances and external events.
- However, people are not just motivated by personal factors but also by being part of a group and culture.
- People go to work because they are motivated by the people that they work with.

Group working

- Most software engineering work is a group activity
  - The development schedule for most non-trivial software projects is such that they cannot be completed by one person working alone.
- Group interaction is a key determinant of group performance
- Flexibility in group composition is limited
  - Managers must do the best they can with available people.

Group composition

- Group composed of members who share the same motivation can be problematic
  - Task-oriented - everyone wants to do their own thing
  - Self-oriented - everyone wants to be the boss
  - Interaction-oriented - too much chatting, not enough work
- An effective group has a balance of all types
- Can be difficult to achieve because most engineers are task-oriented
- Need for all members to be involved in decisions which affect the group

Group leadership

- Leadership depends on respect not titular status.
- There may be both a technical and an administrative leader.
- Democratic leadership is more effective than autocratic leadership.
- A career path based on technical competence should be supported.

Group cohesiveness

- In a cohesive group, members consider the group to be more important than any individual in it.
- Advantages of a cohesive group are:
  - Group quality standards can be developed
  - Group members work closely together so inhibitions caused by ignorance are reduced
  - Team members learn from each other and get to know each other’s work
  - Egoless programming where members strive to improve each other’s programs can be practiced
Developing cohesiveness

- Cohesiveness is influenced by factors such as the organizational culture and the personalities in the group
- Cohesiveness can be enhanced through
  - Social events
  - Development of a group identity and territory
  - Explicit team-building activities
- Openness with information is a simple way of ensuring all group members feel part of the group

Potential problems with a cohesive group

- Irrational resistance to a leadership change
- Groupthink: the name given to a situation where the critical abilities of members are eroded by group loyalties

Group communications

- Good communications are essential for effective group working.
- Information must be exchanged on the status of work, design decisions, and changes to previous decisions.
- Good communications also strengthens group cohesion as it promotes understanding.

Group organization

- Software engineering group sizes should be relatively small (< 8 members)
- Break big projects down into multiple smaller projects
- Small teams may be organized in an informal, democratic way
- Chief programmer teams try to make the most effective use of skills and experience

Democratic team organization

- The group acts as a whole and comes to a consensus on decisions affecting the system
- The group leader serves as the external interface of the group but does not allocate specific work items
- Rather, work is discussed by the group as a whole and tasks are allocated according to ability and experience
- This approach is successful for groups where all members are experienced and competent
**Extreme programming groups**

- Extreme programming groups are variants of democratic organization.
- In extreme programming groups, some ‘management’ decisions are devolved by the group members.
- Programmers work in pairs and take a collective responsibility for code that is developed.

**Chief programmer teams**

- Consist of a kernel of specialists helped by others added to the project as required.
- The motivation behind their development is the wide difference in ability in different programmers.
- Chief programmer teams provide a supporting environment for very able programmers to be responsible for most of the system development.

**Problems**

- This chief programmer approach, in different forms, has undoubtedly been successful.
- However, it suffers from a number of problems:
  - Talented designers and programmers are hard to find. Without exceptional people in these roles, the approach will fail.
  - Other group members may resent the chief programmer taking the credit for success, thus may deliberately undermine his/her role.
  - High project risk as the project will fail if both the chief and deputy programmer are unavailable.
  - Organizational structures and grades may be unable to accommodate this type of group.

**Choosing and keeping people**

- Choosing people to work on a project is a major managerial responsibility.
- Appointment decisions are usually based on:
  - information provided by the candidate (their resume or CV)
  - information gained at an interview
  - recommendations from other people who know the candidate
- Some companies use psychological or aptitude tests:
  - There is no agreement on whether or not these tests are actually useful.

---

[Staff selection factors table]

<table>
<thead>
<tr>
<th>Factor</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application domain experience</td>
<td>For a project to develop a successful system, the developers must understand the application domain.</td>
</tr>
<tr>
<td>Platform experience</td>
<td>May be significant if low-level programming is involved; otherwise, not usually a critical attribute.</td>
</tr>
<tr>
<td>Programming language experience</td>
<td>Generally only significant for short-duration projects where there is insufficient time to learn a new language.</td>
</tr>
<tr>
<td>Educational background</td>
<td>May provide an indicator of the basic fundamentals which the candidate should know and other ability to learn. This factor becomes increasingly irrelevant as engineers gain experience across a range of projects.</td>
</tr>
<tr>
<td>Communication ability</td>
<td>Very important because of the need to communicate orally and in writing with other engineers, managers, and customers.</td>
</tr>
<tr>
<td>Adaptability</td>
<td>Adaptability may be judged by looking at different types of experience which candidates have talked to as an important attribute as it indicates an ability to learn.</td>
</tr>
<tr>
<td>Attitude</td>
<td>Project staff should have a positive attitude to their work and should be willing to learn new skills. This is an important attribute but often very difficult to assess.</td>
</tr>
<tr>
<td>Personality</td>
<td>Another important attribute but difficult to assess. Candidates must reasonably compatible with other team members. No particular type of personality is more or less suited to software engineering.</td>
</tr>
</tbody>
</table>
Physical workplace provision has an important effect on individual productivity and satisfaction:
- Comfort
- Privacy
- Facilities

Health and safety considerations must be taken into account:
- Lighting
- Heating
- Furniture

Privacy - each engineer requires an area for uninterrupted work.
Outside awareness - people prefer to work in natural light.
Personalization - individuals adopt different working practices and like to organize their environment in different ways.

Workspace should provide private spaces where people can work without interruption. Providing individual offices for staff has been shown to increase productivity.
However, teams working together also require spaces where formal and informal meetings can be held.

Workspace organization

Office layout

The People Capability Maturity Model
- Intended as a framework for managing the development of people involved in software development.
- Five stage model:
  - Initial: Ad-hoc people management
  - Repeatable: Policies developed for capability improvement
  - Defined: Standardized people management across the organization
  - Managed: Quantitative goals for people management in place
  - Optimizing: Continuous focus on improving individual competence and workforce motivation.
P-CMM Objectives

- To improve organizational capability by improving workforce capability
- To ensure that software development capability is not reliant on a small number of individuals
- To align the motivation of individuals with that of the organization
- To help retain people with critical knowledge and skills

Key points

- Managers must have some understanding of human factors to avoid making unrealistic demands on people.
- Problem solving involves integrating information from long-term memory with new information from short-term memory.
- Staff selection factors include education, domain experience, adaptability, and personality.

Key points (Continued)

- Software development groups should be small and cohesive.
- Group communications are affected by status, group size, group organization and the sexual composition of the group.
- The working environment has a significant effect on productivity.
- The People Capability Maturity Model is a framework for improving the capabilities of staff in an organization.