
# Team

## SE423: Software Project Management

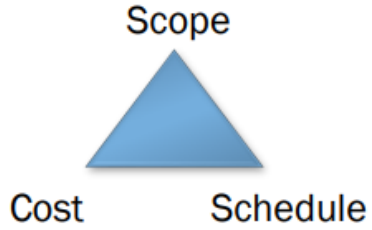
# Outline

- Team Development
- Building Trust
- Managing Conflict
- Embracing Accountability
- Team Models
- Managing People

# Team Development

# Team Development - Introduction

- What are your responsibilities as a team member?
- As a Project Manager?
- How does Team Development relate to the Project Triangle?




- What's more important? The strength of individual team members, or the strength of the team?
- What are some characteristics of a good team?

# Quote

"Software projects fail for one of two general reasons: the project team lacks the knowledge to conduct a software project successfully, or the project team lacks the resolve (*determination, commitment, mental toughness*) to conduct a project effectively."

- Steve C McConnell, Author of "code complete"

# Team Skills, Influence, and Style

- Skills

| Skill | Example Activity |
|---|---|
| Planning | Which resources are pre-assigned to the project? |
| Negotiation | Getting the best possible resources. Sharing resources. |
| Hiring | Outsourcing, virtual teams |
| Risk Management | What if resources become unavailable? |
| Judgment | Is it possible assumptions are being made regarding skill levels? |

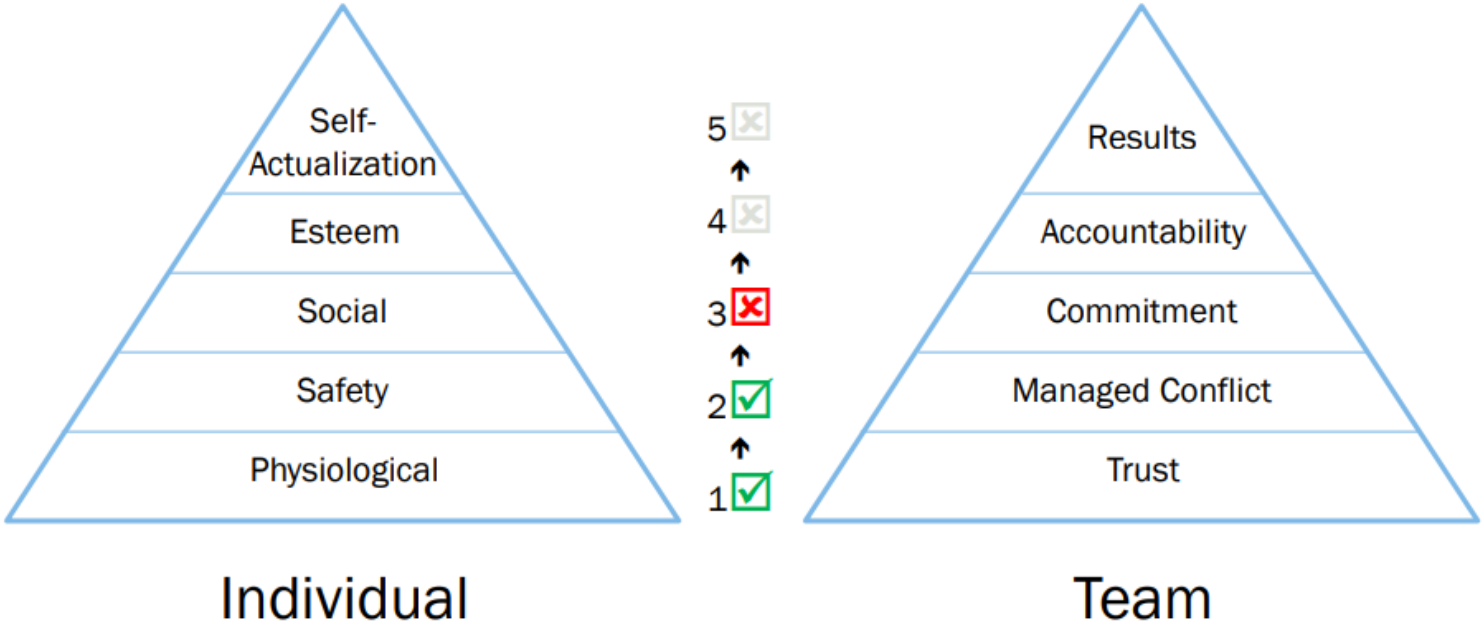
- Types of Influence

| Formal (Legitimate) | Reward | Penalty (Coercive) | Expert | Referent |
|---|---|---|---|---|
| Authority granted by organizational hierarchy or role. | Influence through the ability to offer incentives | Influence through the threat or use of punishment. | Influence based on specialized knowledge or skills. | Influence based on personal traits, charisma, or likability. |

- Leadership Styles

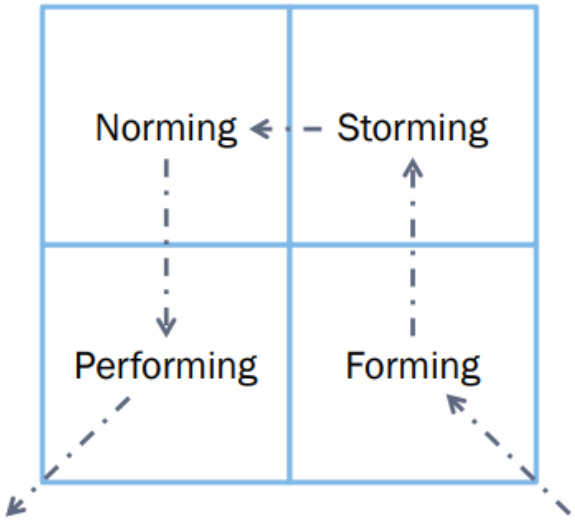
| Directing | Facilitating | Coaching | Supporting |
|---|---|---|---|
| Autocratic | Consultative | Consultative- Autocratic | Consensus |
| Delegating | Bureaucratic | Charismatic | Democratic / Participative |
| Laissez-Faire | Analytical | Driver | Influencing |

# Needs and the Team




- Lower level needs must be satisfied before higher level needs can be addressed

# Tuckman's Model for Team Development




1. Forming: The group comes together and gets to initially know one another and form as a group.
2. Storming: A chaotic competing for leadership and trials of group processes (compete for influence or control, often without clear structure or authority)
3. Norming: Eventually agreement is reached on how the group operates
4. Performing: The group practices its craft and becomes effective in meeting its objectives
5. Dissolving/Adjourning

# Team Structure

* She took her time explaining this.

- What's the team's objective? **three distinct types of team objectives** in project management, each shaping the team's structure, behavior, and required

| objective | Structure Implication |
|---|---|
| **Problem resolution**: Complex, poorly-defined problem. Focuses on 1-3 specific issues. Ex: fixing a showstopper defect (that stop progress entirely)<br>• Team traits: Sense of urgency and pressure | Flexible roles, rapid decision-making, and possibly a temporary task force |
| **Creativity**: New product development: Generating new ideas, products, or approaches.<br>• Team Traits: Encourages divergent thinking and experimentation, tolerates ambiguity and iteration. | Looser hierarchy, diverse skill sets, and a culture that supports risk-taking and idea sharing |
| **Tactical execution**: Carrying-out well-defined plan. Focused tasks and clear roles.<br>• Team Traits: Clear roles and responsibilities. Strong emphasis on timelines, coordination, and accountability. | Hierarchical or matrix structure with tight controls and performance tracking. |

# Building Trust

# Building Trust

- Trust is the foundation of teamwork (Without trust, teams struggle to collaborate, share ideas, or resolve conflict)
- Trust is all about vulnerability, which is difficult for most people: Vulnerability-based trust means: Admitting mistakes, Asking for help, Saying “I don’t know”
- Takes time (Trust isn’t built overnight. It requires: Consistent behavior, Follow-through on commitments, Shared experiences—especially under pressure)
- Needs to be maintained over time.
- Techniques – Behavioral profiling (like MyersBriggs). Helps to admit strengths & weaknesses: (Ex: Recognize their **introverted tendencies,** Adjust task assignments to match their strengths, Create a psychologically safe space for participation)

# Establishing a basis of trust

- A solid basis for trust is *earned,* not granted
  - Most human beings in most situations are inherently wary of others
- Maintain open communication within the project team– communication barriers create project risks
- Be fallible. Take responsibility for making mistakes, correct them, and make it clear how you plan to avoid these mistakes in the future
- Lead by example rather than by mandate. Show the team what you expect from yourself and from them
- Complete all your commitments on time. If you cannot do so, give adequate notice and an explanation, not an excuse
- Don’t confuse friendship with leadership

# Managing Conflict

# Managing Conflict

- Good (healthy) conflict among team members requires trust, which is all about engaging in unfiltered, passionate debate around issues.
- Even among the best teams, conflict will at times be uncomfortable.
- Conflict norms will vary in each team, and must be discussed and made clear:
  - Norms = rules of engagement: These define what's acceptable in debate—tone, language, timing, emotional expression.
- The fear of occasional personal conflict should not deter a team from having regular, productive debate:
  - Avoidance is costly: Teams that fear conflict often fall into **groupthink** (psychological phenomenon where a cohesive group prioritizes harmony and consensus over critical thinking and realistic decision-making), **suppress dissent** (actively prevent or silence disagreement, criticism, or opposition), or make poor decisions.

# Managing Conflict – How?

- Conflict Resolution Techniques
  - Confronting (Problem Solving): Resolve the root cause through open, honest dialogue
  - Compromising: Reach a middle ground by mutual concessions
  - Withdrawal (Avoidance): Delay or sidestep the conflict
  - Smoothing (Accommodating): Preserve harmony by downplaying differences
  - Collaborating: Integrate multiple perspectives to find a creative solution
  - Forcing: Impose a solution through authority or pressure
- Problem Solving
  1. Define the real/root problem
  2. Analyze the problem
  3. Identify Solutions
  4. Pick a Solution
  5. Implement a solution
  6. Review the solution, and confirm that it solved the problem

# Managing and resolving conflict

- Every team encounters conflict: successful teams resolve conflicts, while unsuccessful teams force conflict to the background
- Disagreement is not the same as conflict: conflict involves a hardening of position and associated intractability (insolvability)
- Virtually all conflicts can be traced to a clash between two parties
- To preserve a team's effectiveness, those outside the conflict must assume a neutral stance, even if they themselves agree with one of the parties to the conflict
- One of those outside the conflict must take on the role of conciliator; he or she must mediate between the two parties to defuse the situation
- The goal is to reduce the conflict to a disagreement which can then be addressed through consensus
- Strive for reconciliation between the conflicting parties, if possible

# Achieving **Commitment**

- Commitment requires **clarity** & **buy-in**:
  - Clarity ensures everyone knows exactly what was decided, why it matters, and what's expected next.
  - Buy-in means team members support the decision—even if they didn't fully agree—because they trust the process and feel heard.
- Clarity requires that teams avoid assumptions and ambiguity, and end discussions with a clear understanding about what they've decided upon.
- Buy-in does not require consensus. Members of great teams learn to disagree with one another and still commit to a decision.

# Embracing Accountability

# Embracing Accountability

- Accountability on a strong team occurs directly among peers: In high-performing teams, members don't just rely on the manager to enforce standards—they hold each other accountable.
- For a culture of accountability to thrive, a leader must demonstrate a willingness to confront accountability: A project manager should model transparency, follow through on commitments, and address underperformance respectfully but firmly.
- Best opportunity occurs during meetings and regular review of accomplishments. Example: During a sprint review, the PM might say, "*We committed to five stories but completed three. Let's explore what happened and how we can improve next sprint.*"

# Embracing Accountability

- RACI – Responsible, Accountable, (Support), Consulted, Informed

| | Project Manager | Project Sponsor | Developer 1 | Tester 1 |
|---|---|---|---|---|
| Activity 1 | A | I | R | |
| Activity 2 | R | S | | |
| Activity 3 | C | | R | S |
| Activity 4 | RA | | | C |
| Activity 5 | A | R | S | |

Note: “Support” is not part of the original RACI acronym but is sometimes ad
in variants like RASCI to reflect collaborative roles more accurately.

# Responsibility Assignment Matrix

- A resource planning tool
- Who does What
- Can be for both planning and tracking
- Identify authority, accountability, responsibility
- Who: can be individual, team or department
- Can have totals/summary at end of row or column (ex: total Contributors on a task)

# Simple RAM

| ROLE / Project Deliverable (or Activity) | Executive Sponsor | Project Sponsor | Steering Committee | Advisory Committee | Role #5 | Project Manager | Tech Lead | Functional Lead | SME | Project Team Member | Developer | Administrative Support | Business Analyst | Role #4 | Role #5 | Consultant | PMO | Role #3 | Role #4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Project Leadership | | | | | Project Team Members | | | | | Project Sub-Teams | | | | | External Resources | | | |
| **Initiate Phase Activities** | | | | | | | | | | | | | | | | | | | |
| Request Review by PMO | A/C | R/A | | | | R/A | A/C | | C | | | | | | | | | | |
| Submit Project Request | | | | | | R | | | | | | | | | | | A | | |
| Research Solution | I | | | | | R/A | A/C | A/C | C | | | | C | | | C | | | |
| Develop Business Case | I | A/C | I | I | | R/A | C | C | C | | | | C | | | C | C | | |
| **Plan Phase Activities** | | | | | | | | | | | | | | | | | | | |
| Create Project Charter | C | C | | | | R/A | C | C | C | | | | C | | | C | | | |
| Create Schedule | I | I | I | I | | R/A | C | C | C | C | C | C | C | | | C | I | | |
| Create Additional Plans as Required | I | I | I | | | R/A | | | | I | I | I | I | | | C | I | | |
| **Execute Phase Activities** | | | | | | | | | | | | | | | | | | | |
| Build Deliverables | C/I | C/I | C/I | C/I | | | R/A | R/A | R/A | R/A | R/A | | | | | A/C | | | |
| Create Status Report | I | I | I | I | | R/A | R/A | R/A | R/A | | | | | | | C | I | | |
| **Control Phase Activities** | | | | | | | | | | | | | | | | | | | |
| Perform Change Management | | C | C | C | | R | A | A | A | | | | | | | C | I | | |
| **Close Phase Activities** | | | | | | | | | | | | | | | | | | | |
| Create Lessions Learned | C | C | C | C | | R/A | C | C | C | C | C | C | C | | | C | C | | |
| Create Project Closure Report | I | I | I | I | | R/A | I | I | I | I | I | I | I | | | | I | | |

RACI – Responsible, Accountable, (Support), Consulted, Informed

# Skills Matrix

- Another resource planning tool
- Resources on one axis, skills on other
- Skills can be high level or very specific
- Cells can be X’s or numeric (ex: level, # yrs.)

# Focusing on Results

- The true measure of a great team is that it accomplishes the results it sets out to achieve.
- To avoid distractions, team members must prioritize the results of the team over their individual or organizational (dept) needs.
- To stay focused, team must publicly clarify their desired results and keep them visible.

# Team Models

# Team Models

- Two early philosophies
  - Decentralized/democratic
  - Centralized/autocratic
- Variation
  - Controlled Decentralized

# Team Models

- **Business Team**: commonly used in software projects and general business environments.
  - Most common model
  - Technical lead + team (rest of team at equal status)
  - Hierarchical with one principal contact
  - Can be strong (The lead has clear authority and control) or loose hierarchy (The lead facilitates rather than commands—more collaborative)
  - Adaptable and general
  - Variation: **Democratic Team**
    - All decisions made by whole team
    - The Business Team is often the default starting point—especially in structured environments. But as teams mature, gain trust, and develop shared norms, they may evolve toward a Democratic Team model.

# 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
- Sometimes known as self-organizing teams.

# Agile programming groups

- Agile programming groups are variants of democratic organization
- In Agile programming groups, some ‘management’ decisions are delegated to group members
- In XP, Programmers work in pairs and take a collective responsibility for code that is developed

# Team Models

**Chief-Programmer Team** a.k.a. ‘surgical team: From IBM in 70’s*:* Puts a superstar at the top: Just as a surgeon performs the critical operation while being supported by anesthetists, nurses, and technicians, the **chief programmer** writes the core code while being supported by a carefully selected team of specialists.

- Backup Programmer/co-pilot (Equally skilled—ready to step in if needed, often reviews or shadows the chief.)
- Program Clerk (Manages documentation, version control, and technical records.)
- Administrator (Handles administrative tasks, freeing technical staff to focus on development.)
- Tool-smith (Builds and maintains development tools and environments.)
- “Language lawyer” (Deep expert in the programming language(s) used—consulted for tricky syntax or semantics.)

- Issues: Difficult to achieve, Ego issues: superstar and/or team
- Can be appropriate for creative projects or tactical execution

# 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 so 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

# Team Models

- Skunk works Team
  - Put a bunch of talented, creative developers away from the mother ship
    - Off-site literally or figuratively
  - Pro: Creates high ownership & buy-in
  - Con: Little visibility into team progress
  - Applicable: exploratory projects needing creativity
    - Not on well-defined or narrow problem
- SWAT Team
  - Highly skilled team
  - Skills tightly match goal
  - Members often work together
  - Ex: security swat team, Oracle performance team

# Team Models

- Large teams
  - Communication increases multiplicatively
    - Square of the number of people
    - 50 programmers = 1200 possible paths
    - Communication must be formalized
  - Always use a hierarchy
  - Reduce units to optimal team sizes
    - Always less than 10 (seven plus or minus one)
- What is the **optimal** team size?
  - 4-6 developers
    - Tech lead + developers
  - Small projects inspire stronger identification
  - Increases cohesiveness
  - QA, ops, and design on top of this

g

# Managing People

# People in the process

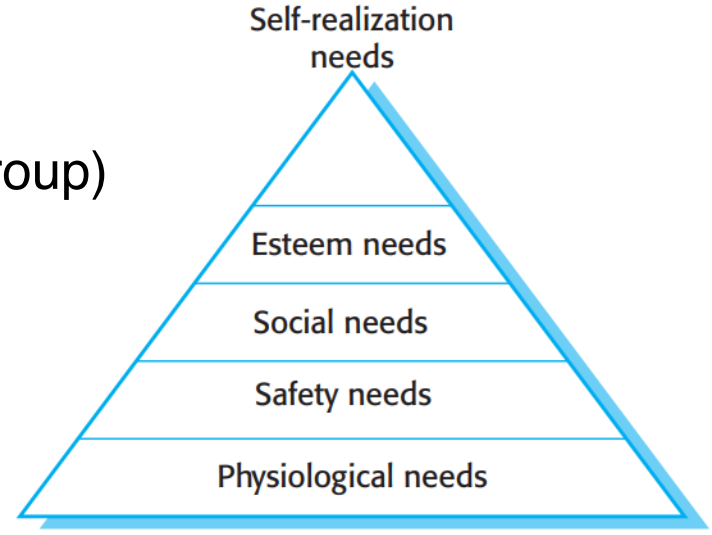
- People are an organization’s most important assets
- The tasks of a manager are essentially people oriented. Unless there is some understanding of people, management will be unsuccessful

# Management activities (related to 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)

# Motivation

- An important role of a manager is to motivate the people working on a project (motivation is needs-based)
- Motivation is a complex issue but it 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)
- Maslow's hierarchy of needs



# Motivating people

- People's Motivations depend on satisfying needs
- It can be assumed that physiological and safety needs are satisfied
- Social, esteem and self-realization (self-actualization) needs are most significant from a managerial viewpoint
- If a lower set of needs is continually unmet for an extended period of time, the individual will temporarily shift focus downwards those needs – dropping down to that level until those lower needs are reasonably satisfied again.

# Need satisfaction

- Social
  - Provide communal facilities (shared spaces, break rooms, collaborative zones, etc.)
  - Allow informal communications (support casual conversations, team chats, non-work-related exchanges such as during lunch, etc.)
- Esteem
  - Recognition of achievements
  - Appropriate rewards (bonuses, promotions, certificates, symbolic gestures, etc.)
- Self-realization
  - Training – people want to learn more
  - Responsibility (empower team members to make decisions)

# Personality types

- The needs hierarchy is almost certainly an over-simplification
- Motivation should also take into account different 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 buy a house, 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

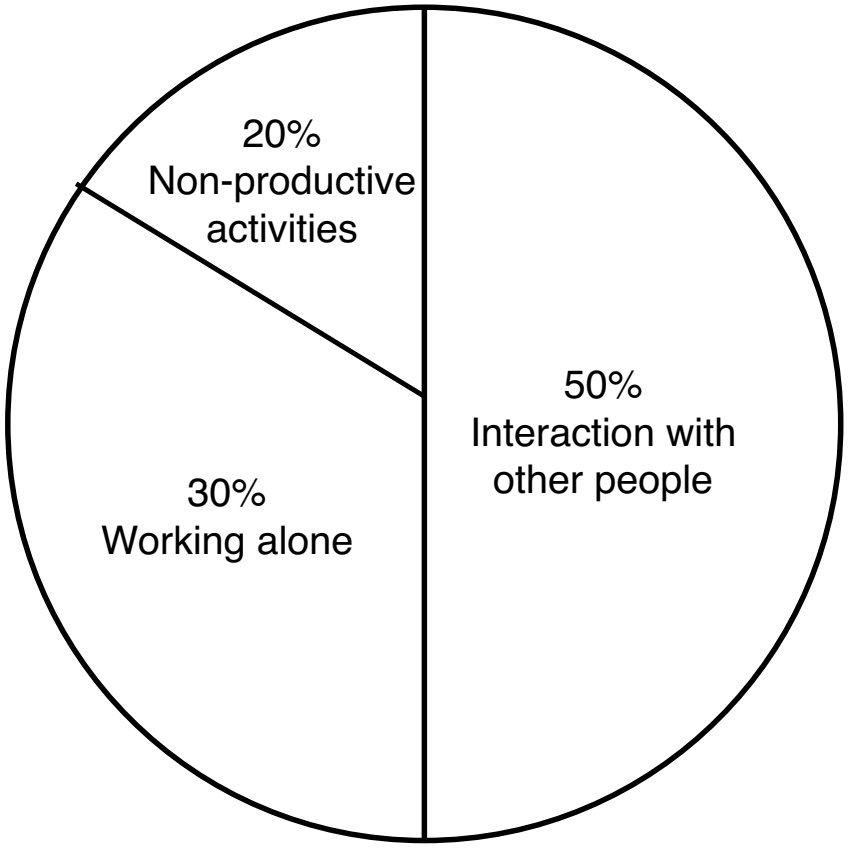
# Motivation balance

- Individual motivations are made up of elements of each class (from previous slide).
- 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 also because they are motivated by the people that they work with

# Group composition: group working

- Most software engineering 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

# Time distribution



# 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

# Group composition: group leadership

- Leadership depends on respect not titular status (titular = holding a title)
- 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 encouraged through
  - Social events, such as team outings
  - Developing 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

# Group loyalties

- Group members tend to be loyal to cohesive groups
- 'Groupthink' (العصبية للمجموعة) is preservation of group irrespective of technical or organizational considerations
- Management should act positively to avoid groupthink by forcing external involvement with each group

# 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

Notes:

- Status of group members
  - Higher status members tend to dominate conversations
- Personalities in groups
  - Too many people of the same personality type can be a problem
- Communication channels
  - Communications channelled though a central coordinator tend to be ineffective

# Group organization

- Software engineering group sizes should be relatively small (< 10 members) optimal is less than seven (see previous slides)
- 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

# 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
  - Interviews with potential co-workers
  - 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

| Factor | Explanation |
|---|---|
| Application domain experience | For a project to develop a successful system, the developers must understand the application domain |
| Platform experience | May be significant if low-level programming is involved. Otherwise, not usually a critical attribute. |
| Programming language experience | Normally only significant for short duration projects where there is insufficient time to learn a new language. |
| Educational background | May provide an indicator of the basic fundamentals which the candidate should know and of their ability to learn. This factor becomes increasingly irrelevant as engineers gain experience across a range of projects. |
| Communication ability | Very important because of need for project staff to communicate orally and in writing with other engineers, managers, and customers. |
| Adaptability | Adaptability may be judged by looking at the different types of experience that candidates have had. This is an important attribute as it indicates an ability to learn. |
| Attitude | Project staff should have a positive attitude towards their work and should be willing to learn new skills. This is an important attribute but often very difficult to assess. |
| Personality | Again, an important attribute but difficult to assess. Candidates must be reasonably compatible with other team members. No particular type of personality is more or less suited to software engineering. |

# Working environments

- Physical workplace provision has an important effect on individual productivity and satisfaction
  - Comfort
  - Privacy [see next slide]
  - Facilities
- Health and safety considerations must be taken into account
  - Lighting
  - Heating
  - Furniture

# Environmental factors

- Privacy – each engineer requires an area for uninterrupted work (see next slide)
- 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 organization

- Workspaces 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

# Our perspective about the team members

- The phrase ‘human resources’ has its origins as an economic and corporate asset term
- These origins lead to two semantic **problems**:
  - People are (to some extent) considered in the same light as inanimate resources: “maximizing the ROI in human capital is the goal of HR management”
  - People are considered (to some extent) as being uniform, interchangeable, and expendable
- To improve the chances of project success, the project manager should ensure that he views and acts with his team members as “collaborators”, “stakeholders”, “teammates” by recognizing and nurturing their individual talent and fostering psychological safety and team cohesion.

# The project manager's skills

- The project manager must develop people skills in a number of related areas:
  - Interpersonal skills
  - Shaping project culture
  - Managing people
  - Making people better
  - Leadership

# Interpersonal skills

- The project manager needs a well-developed set of interpersonal skills in order to effectively manage a project
- *Effective communication*.
  - Follows a defined protocol: structured channels and formats (e.g., status reports, stand-ups, escalationتصعيد paths). This ensures clarity and consistency.
  - Reaches all recipients in a timely manner
  - Provides all the information necessary, no more or no less
- *Influencing the organization*. The ability to get a project completed depends on the abilities of and respect for the PM and PM team (influence, negotiate, and build credibility)
- Leadership: is one of the most misunderstood concepts in project management–leadership has little to do with authority: it's about inspiring, guiding, and aligning people toward a shared goal

# Interpersonal skills (2)

- *Motivation*. Providing people with incentives to act in the interest of the project and overcome inevitable challenges
  - Motivation comes in a number of forms; studies show financial incentives are among the least effective motivators
- *Negotiation and conflict management*. Working with others to come to agreement and overcome differences, real or perceived
  - Reaching consensus and defusing conflict are two elements of these skills
- *Problem solving*. Being able to recognize and define problems; formulate, evaluate, and make decisions regarding possible solutions
  - Problem solving is a skill that can be learned but requires practice: progressive elaboration helps manage the complexity of most project problems

# Shaping project culture

- The project manager must shape the project culture within the broader context of the framework of the organizational culture
- *Understand organizational culture*
  - Understanding the attitude of the organization toward the project and toward project management processes is essential (Every organization has its own values, norms, and attitudes toward projects—some embrace structured methodologies, others prefer informal or ad hoc approaches)
  - Organization attitudes toward *real*–as opposed to fanciful–project management practices varies widely (practices that exist **in theory but not in reality**)
- *Understand each team member's engineering and personal background*
  - Education, experience, generation, and personality all determine what roles the person may take and how they will fit in with the rest of the team
- Assess each team member's cultural role or roles and match this to their project role

# Shaping project culture

- Match cultural and project roles to people
- Representative cultural roles include:
  - Leader (Takes initiative, inspires others)
  - Listener/talker (Facilitates communication, bridges gaps)
  - Complainer/naysayer (who habitually expresses negative or skeptical views: Raises concerns, challenges assumptions)
  - Expert (Offers deep knowledge and credibility)
  - Charger/plodder (Pushes forward with energy)
- Significant project roles include:
  - *Project manager*. Guides the team planning and tracks progress against the plan
  - *Team leader*. Builds and maintains an effective team
  - *Development leader*. Produces a superior product
- Remain non-judgmental: every cultural role has potential value for the team

# Project environments (context: project culture)

- Cultural and social factors
  - Projects are planned and implemented in a social and cultural context
  - Some factors that will influence a project might be: economic, educational, ethical, ethnic, and religious
  - These factors can affect the authority of the project manager
- International and political
  - As globalization expands, familiarity with local laws and customs is essential for effective project management
  - Shifting international, national, and local political factors can have dramatic influences on a project
  - Some mundane factors that will influence a project might be: time zone differences, holidays, travel requirements, teleconferencing capabilities

# Shaping project culture

- Monitor and manage team culture as you would manage technical issues
  - Make each person's project role clear
  - Understand each person's personality and make known your understanding of each person's social role in a positive way
  - State and maintain your view of the team
    - Example: "we are all equals in this team, but if a critical situation arises, I will make the decision."

•Recognize potential cultural role problems before they have a negative impact on the team (A dominant voice silencing others, A skeptic undermining morale)

•Solve project role problems before they have a negative impact on the project

# Project Culture

- Monitor team members' satisfaction with their role.
  - You will need to monitor team role satisfaction throughout the project.
  - It is especially important during project launch when roles can be changed easily.
- Monitor the engineering task-to-people fit to make sure each person can perform the engineering task.
  - Role mismatches lead to project problems and schedule slip.
  - If you don't review task-to-people fit, you will be unaware of each person's ability, or lack thereof, to perform tasks.
- Monitor the cultural climate of the team as a whole.
  - Monitor project culture by watching and listening to each team member.
  - Guide the cultural views of the project by your own example.
  - You need to be able to monitor the team culture throughout the project in order to build confident people and a confident team.

# Managing People

- The main problems of our work are not so much technological as sociological in nature
- (optional reading) One of the most enlightened sources for information about managing people is:

  *Peopleware: Productive Projects and Teams*, Third Edition,
- Most technical managers come from technical backgrounds and manage that way as if they are solving an engineering problem
- Most managers do not consider individual idiosyncrasies (personal strange habit), yet these can make or break a project

# Managing People

Guidelines for Managing People:

- Gain visibility without micromanagement.
- Review process and products, not people.
  - Say: “This testing regime allows too many bugs to slip through to the customer; what can we do to correct this?”
  - Not: “You all don’t seem to know how to write unit tests properly.”
- Coordinate, don’t manipulate.
- Use your knowledge, not your position of power.
- Channel people, don’t put dams in front of them.
  - Redirect misplaced energy or effort, don’t cut it off
- Focus on project and people needs, not your authority as manager.
  - This will maintain your leadership role without explicit effort

# How NOT to manage 'thought workers': production line efficiency

- The application of **production line** efficiency principles to thought workers can be a sign of disaster for a project
- Common errors include:
  - Squeeze out error – make the 'machine' run as smoothly as possible
  - Take a hard line about people goofing off on the job (Assuming that any non-visible output equals laziness: creative people need time to reflect)
  - Treat workers as interchangeable pieces of the machine
  - Optimize the steady state – don't worry about start up and shut down
  - Standardize procedure – run by the book (rigid processes)
  - Eliminate experimentation (Discouraging trial-and-error or creative risk-taking)
- Consider how each of these errors might impact your own work

# Making People Better

- Make professional development a project goal
- Recognize long- and short-term development goals
  - Short-term goals focus on skills needed for the project
  - Long-term goals prepare team members for future projects
- Let each team member specify personal improvement goals
  - Starting point: Competency framework goals such as: Technical, leadership, domain, general/personal, communication, and management
- For a fixed interval, have team members track their individual time expenditure in detail
  - This concept goes well beyond the weekly timesheet concept
  - This is an awareness-building activity, *not* a monitoring activity
  - The *Personal Software Process (PSP)* is a formal application of this idea

# Making People Better

g

- *Have team members track their individual time*
  - Goes beyond the weekly timesheet concept
- One formal approach to this is the *Personal Software Process* (PSP) which shows developers how to:
  - Manage the quality of their projects
  - Make commitments they can meet
  - Improve estimating and planning
  - Reduce defects in their products

[See note below for more info.]

- One element of PSP is to track daily time spent on *all* project-related tasks:
  - Reading
  - Writing
  - Meeting
  - Training
  - Requirements
  - Design
  - Coding
  - Testing
  - Deployment
  - E-mail
  - Phone

# Leadership

- *Leadership* is the capacity to mobilize a group to solve the most challenging problems (wrong to think that leadership is about authority)
- Two types of leadership:
  - *Technical leadership* (a.k.a. *authority*) motivates people when the problem can be solved by *known* means: the leader directs, instructs and applies best practices.
  - *Adaptive leadership* motivates people to solve problems that *cannot* be solved by known means (needed when the problem is **complex or novel**)
    - Requires learning on the part of the leader as well as the group (facilitate learning and research)
    - Often involves leading with good questions (provoke thinking, guide inquiry)
    - Requires a deep clarity of purpose (to keep the team anchored toward its goals) : Example: during brainstorming meetings, the leader is the one that needs to stop the conversation from diverging to unimportant arguments, problems and bring the discussion back to the essential questions.

# Technical *vs. adaptive leadership*

| | Technical leadership | Adaptive Leadership |
|---|---|---|
| Approach to problem-solving | Technically-oriented: solves problems by known means | Adaptive: approaches problems not solved by known means |
| Motivation driver | Vision: personal view of outcome. The leader's personal vision drives the team —"Here's what I see for us." | Sense of purpose: big-picture view of goals: The leader anchors the team in shared meaning —"Why does this matter?" |
| Tool for motivation | Personality: Charisma, technical expertise, confidence, and decisiveness inspire trust and action. | Progress: Motivation comes from learning, iteration, and visible movement toward clarity. |

# Technical vs. adaptive leadership

- Need to be aware of which kind of leadership is needed and choose between technical leadership and adaptive leadership for a project based on whether the problem can be solved by known means or not.
- Most IT projects have a mix of known and unknown problems.
- An adaptive leader can address known problems by *delegation*–allow one who can lead the team in solving the problem by known means do so when appropriate

# Working toward consensus

- *Consensus* simply means ‘general agreement: But in the context of project teams, **consensus** doesn’t mean everyone agrees—it means everyone feels heard, and the group can move forward with a shared commitment.
- If possible, reach consensus on all decisions affecting the team
- Simple majority rule can be detrimental on a small team in which everyone’s commitment is essential
- To reach that point, leaders must **curate the quality of discourse**: Valid argument need to be highlighted, invalid arguments need to constructively disqualified to protect decision quality:
  - **Example**: In a product design debate, a team member cites user data to support a feature. The leader highlights this as a valuable contribution, reinforcing data-driven thinking.

# Working toward consensus

- Steps to reaching consensus:
  - Pose the problem to be solved or decision to be made
  - Solicit input on options for the problem or decision
  - Discuss and weigh pros and cons of each option
  - Work toward the options that have the most benefit for the project
  - When it is clear which option is most suitable, work with its opponents to help them buy in to it–persuade, don't dictate
  - Consensus takes more time than mandate, but has huge pay-offs
- Beware passive/aggressive agreement–this is simply subsumed conflict

# Leading People

This is a reprise of our view of *establishing a basis of trust*

- Be confident in yourself and the team
- Maintain open communication within the project team– communication barriers create project risks
- Be fallible. Take responsibility for making mistakes, correct them, and make it clear how you plan to avoid these mistakes in the future
- Lead by example rather than by mandate. Show the team what you expect from yourself and from them
- Utilize all the talents of your team. You can't make the project succeed all by yourself
- Complete all your commitments on time. If you cannot do so, give adequate notice and an explanation, not an excuse
- Don't confuse friendship with leadership: Warmth and connection are vital—but leadership requires **clarity, boundaries, and accountability**

# Key points

- Managers must have some understanding of human factors to avoid making unrealistic demands on people
- Staff selection factors include education, domain experience, adaptability and personality
- Software development groups should be small and cohesive
- Group communications are affected by status, group size and the group composition and organization
- The working environment has a significant effect on productivity
- Develop methods to monitor the project culture and technical status early so you can continually monitor these issues as the project progresses.