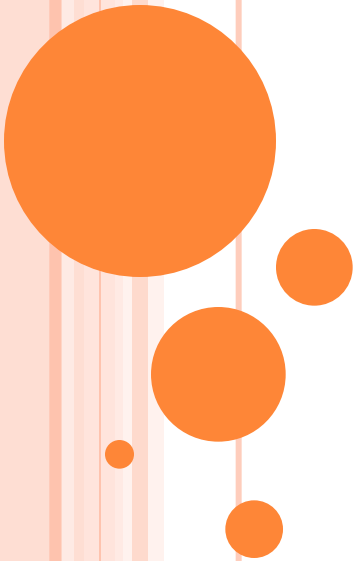
Software Engineering



Lecture-4

- Project Management Concepts

Topics Covered

- The Management Spectrum
- The People
- The Product
- The Process
- The Project

The Management Spectrum

- Effective software project management focuses on these items (in this order)
  - The people
    - Deals with the cultivation of motivated, highly skilled people
    - Consists of the stakeholders, the team leaders, and the software team
  - The product
    - Product objectives and scope should be established before a project can be planned
  - The process
    - The software process provides the framework from which a comprehensive plan for software development can be established
  - The project
    - Planning and controlling a software project is done for one primary reason…it is the only known way to manage complexity
    - In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns

People

Product

Process

Project

The People: The Stakeholders

- Five categories of stakeholders
  - **Senior managers** – define business issues that often have significant influence on the project
  - **Project (technical) managers** – plan, motivate, organize, and control the practitioners who do the work
  - **Practitioners** – deliver the technical skills that are necessary to engineer a product or application
  - **Customers** – specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome
  - **End users** – interact with the software once it is released for production use

The People: Team Leaders

- Competent practitioners often fail to make good team leaders; they just don't have the right people skills
- Qualities to look for in a team leader
  - **Motivation** – the ability to encourage technical people to produce to their best ability
  - **Organization** – the ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product
  - **Ideas or innovation** – the ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application
- Team leaders should use a problem-solving management style
  - Concentrate on understanding the problem to be solved
  - Manage the flow of ideas
  - Let everyone on the team know, by words and actions, that quality counts and that it will not be compromised

(More on next slide)

The People: Team Leaders (continued)

- Another set of useful leadership traits
  - **Problem solving** – diagnose, structure a solution, apply lessons learned, remain flexible
  - **Managerial identity** – take charge of the project, have confidence to assume control, have assurance to allow good people to do their jobs
  - **Achievement** – reward initiative, demonstrate that controlled risk taking will not be punished
  - **Influence and team building** – be able to "read" people, understand verbal and nonverbal signals, be able to react to signals, remain under control in high-stress situations

The People: The Software Team

- Seven project factors to consider when structuring a software development team
  - The difficulty of the problem to be solved
  - The size of the resultant program(s) in source lines of code
  - The time that the team will stay together
  - The degree to which the problem can be modularized
  - The required quality and reliability of the system to be built
  - The rigidity of the delivery date
  - The degree of sociability (communication) required for the project

(More on next slide)

The People: The Software Team (continued)

- Four organizational paradigms for software development teams
  - **Closed paradigm** – traditional hierarchy of authority; works well when producing software similar to past efforts; members are less likely to be innovative
  - **Random paradigm** – depends on individual initiative of team members; works well for projects requiring innovation or technological breakthrough; members may struggle when orderly performance is required
  - **Open paradigm** – hybrid of the closed and random paradigm; works well for solving complex problems; requires collaboration, communication, and consensus among members
  - **Synchronous paradigm** – organizes team members based on the natural pieces of the problem; members have little communication outside of their subgroups

(More on next slide)

The People: The Software Team (continued)

- Five factors that cause team toxity (i.e., a toxic team environment)
  - A frenzied work atmosphere
  - High frustration that causes friction among team members
  - A fragmented or poorly coordinated software process
  - An unclear definition of roles on the software team
  - Continuous and repeated exposure to failure
- How to avoid these problems
  - Give the team access to all information required to do the job
  - Do not modify major goals and objectives, once they are defined, unless absolutely necessary
  - Give the team as much responsibility for decision making as possible
  - Let the team recommend its own process model
  - Let the team establish its own mechanisms for accountability (i.e., reviews)
  - Establish team-based techniques for feedback and problem solving

The People: Coordination and Communication Issues

- Key characteristics of modern software make projects fail
  - scale, uncertainty, interoperability
- To better ensure success
  - Establish effective methods for coordinating the people who do the work
  - Establish methods of formal and information communication among team members

Group Dynamics

- Based on studies published by B. Tuckman in 1965
- Updated later in 1977
- Describes a four-stage model
  - Forming
  - Storming
  - Norming
  - Performing

Group Dynamics Model

- Forming
  - Group members rely on safe, patterned behavior and look to the group leader for guidance and direction
  - Impressions are gathered and similarities and differences are noted
  - Serious topics and feelings are avoided
  - To grow, members must relinquish the comfort of non-threatening topics and risk the possibility of conflict

Group Dynamics Model

- Storming
  - As group members organize for the tasks, conflict inevitably results in their personal relations and cliques start to form
  - Individuals have to bend and mold their feelings to fit the group
  - Fear of exposure or fear of failure causes an increased desire for structural clarification and commitment
  - Conflicts arise over leadership, structure, power, and authority
  - Member behavior may have wide swings based on emerging issues of competition and hostilities
  - Some members remain silent while others attempt to dominate

Group Dynamics Model (continued)

- Norming
  - Members engage in active acknowledgement of all members' contributions, community building, and solving of group issues
  - Members are willing to change their preconceived ideas or opinions based on facts presented by the group
  - Leadership is shared, active listening occurs, and cliques dissolve
  - Members began to identify with one another, which leads to a level of trust in their personal relations and contributes to cohesion
  - Members begin to experience a sense of group belonging

Group Dynamics Model (continued)

- Performing
  - The capacity, range, and depth of personal relations in the group expand to true interdependence
  - Members can work independently, in subgroups, or altogether with equal ability and success
  - The group is most productive, members become self-assuring, and the need for group approval is past
  - Genuine problem solving can occur leading towards optimal solutions

People

Product

Process

Project

The Product

- The scope of the software development must be established and bounded
  - **Context** – How does the software to be built fit into a larger system, product, or business context, and what constraints are imposed as a result of the context?
  - **Information objectives** – What customer-visible data objects are produced as output from the software? What data objects are required for input?
  - **Function and performance** – What functions does the software perform to transform input data into output? Are there any special performance characteristics to be addressed?
- Software project scope must be unambiguous and understandable at both the managerial and technical levels

(More on next slide)

The Product (continued)

- Problem decomposition
  - Also referred to as partitioning or problem elaboration
  - Sits at the core of software requirements analysis
- Two major areas of problem decomposition
  - The functionality that must be delivered
  - The process that will be used to deliver it

People

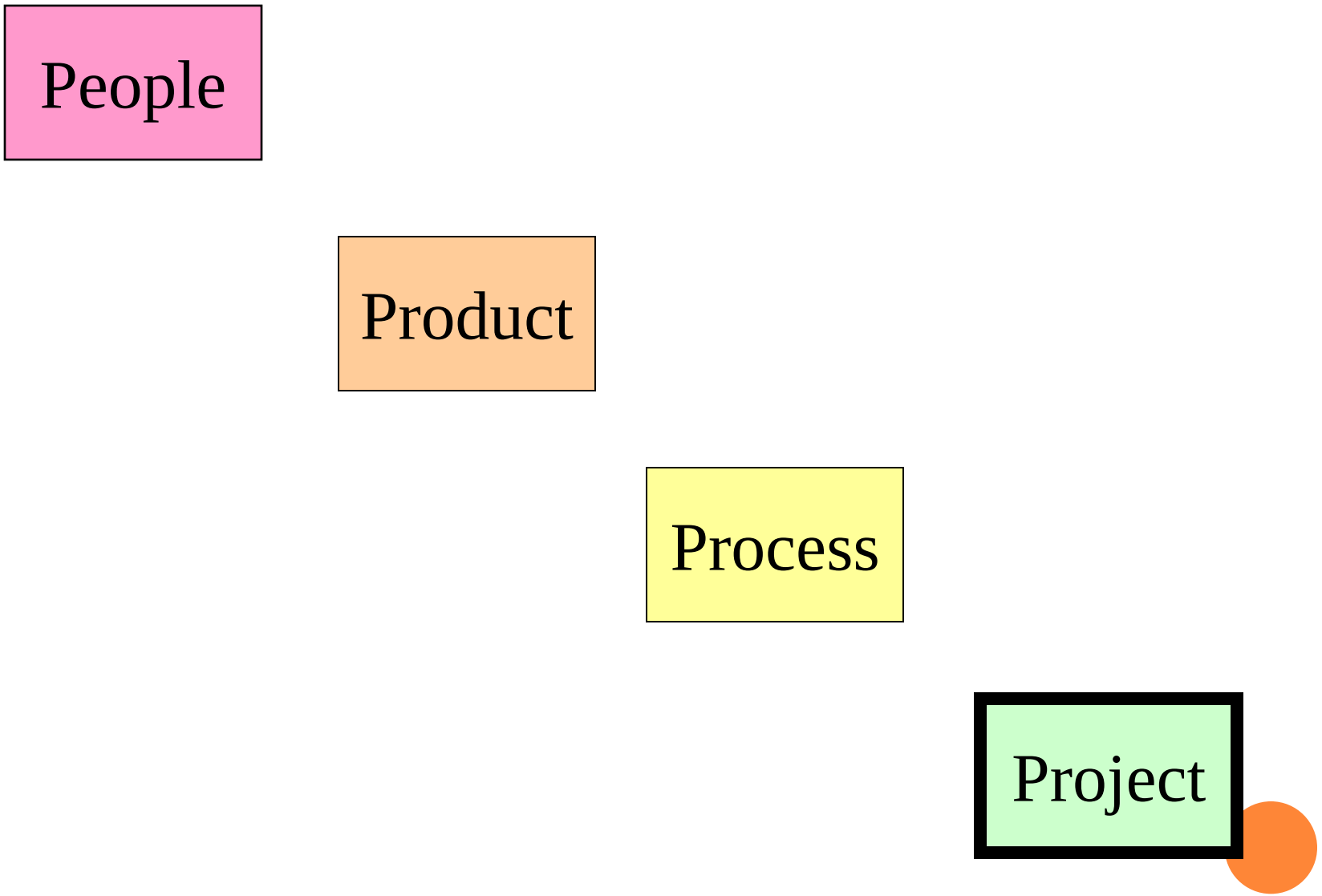
Product

Process

Project

The Process

- Getting Started
  - The project manager must decide which process model is most appropriate based on
    - The customers who have requested the product and the people who will do the work
    - The characteristics of the product itself
    - The project environment in which the software team works
  - Once a process model is selected, a preliminary project plan is established based on the process framework activities
  - Process decomposition then begins
  - The result is a complete plan reflecting the work tasks required to populate the framework activities
- Project planning begins as a melding of the product and the process based on the various framework activities


People
Product
Process
Project

The Project: A Common Sense Approach

- Start on the right foot
  - Understand the problem; set realistic objectives and expectations; form a good team
- Maintain momentum
  - Provide incentives to reduce turnover of people; emphasize quality in every task; have senior management stay out of the team's way
- Track progress
  - Track the completion of work products; collect software process and project measures; assess progress against expected averages
- Make smart decisions
  - Keep it simple; use COTS or existing software before writing new code; follow standard approaches; identify and avoid risks; always allocate more time than you think you need to do complex or risky tasks
- Conduct a post mortem analysis
  - Track lessons learned for each project; compare planned and actual schedules; collect and analyze software project metrics; get feedback from teams members and customers; record findings in written form

The Project: Signs that it is in Jeopardy

- Software people don't understand their customer's needs
- The product scope is poorly defined
- Changes are managed poorly
- The chosen technology changes
- Business needs change (or are poorly defined)
- Deadlines are unrealistic
- Users are resistant
- Sponsorship is lost (or was never properly obtained)
- The project team lacks people with appropriate skills
- Managers (and practitioners) avoid best practices and lessons learned

The Project: The W^{5}HH Principle

A series of questions that lead to a definition of key project characteristics and the resultant project plan

- **Why** is the system being developed?
  - Assesses the validity of business reasons and justifications
- **What** will be done?
  - Establishes the task set required for the project
- **When** will it be done?
  - Establishes a project schedule
- **Who** is responsible for a function?
  - Defines the role and responsibility of each team member
- **Where** are they organizationally located?
  - Notes the organizational location of team members, customers, and other stakeholders
- **How** will the job be done technically and managerially?
  - Establishes the management and technical strategy for the project
- **How** much of each resource is needed?
  - Establishes estimates based on the answers to the previous questions

Summary