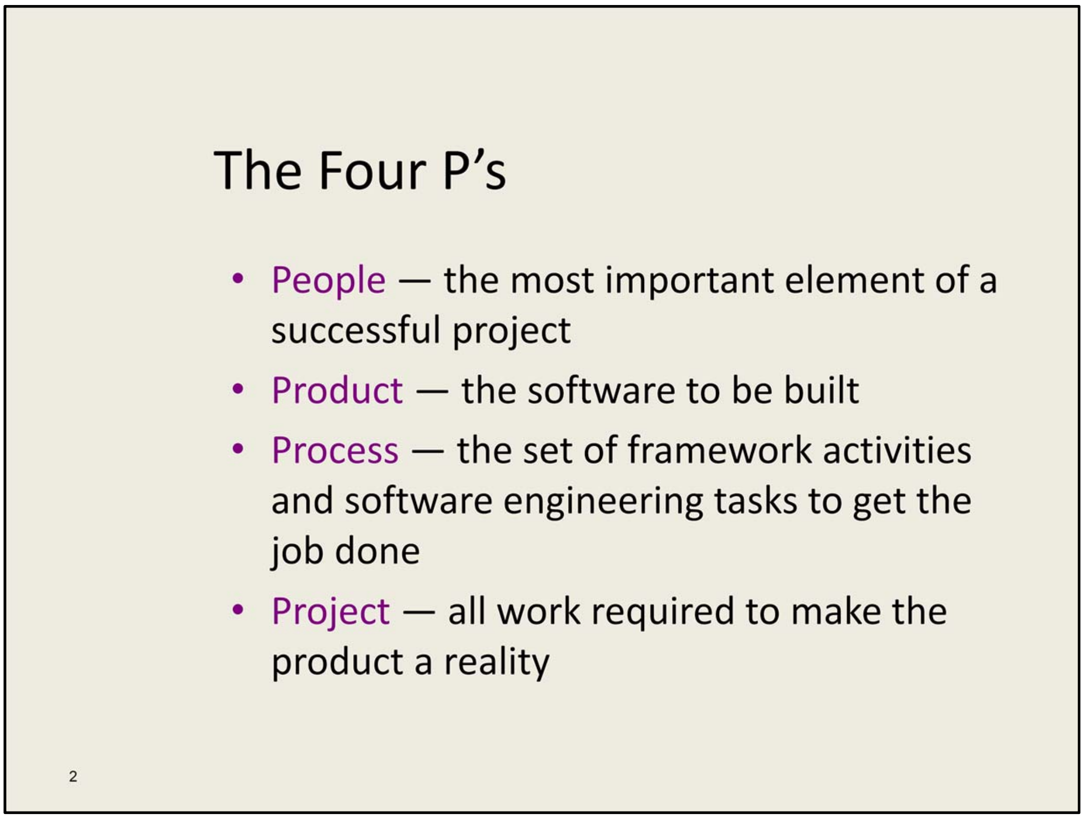
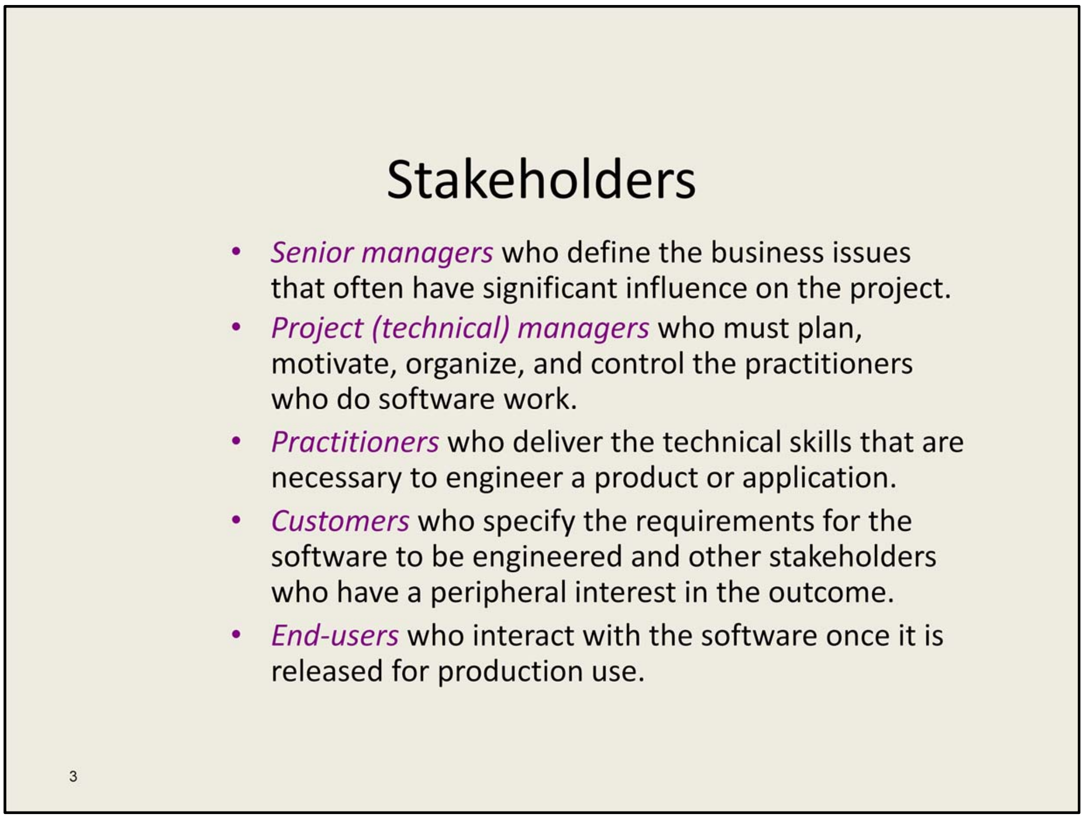
Software Project Management


The Four P's

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

2

Effective software project management focuses on the four P's: people, product, process, and project. **The order is not arbitrary**. The manager who forgets that software
engineering work is an intensely human endeavor will never have success in project management. A manager who fails to encourage comprehensive stakeholder
communication early in the evolution of a product risks building an elegant solution for the wrong problem. The manager who pays little attention to the process runs the risk of inserting competent technical methods and tools into a vacuum. The manager who embarks without a solid project plan jeopardizes the success of the project.



Stakeholders

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

3

Every software project is populated by people who fall within this taxonomy.2 To be

effective, the project team must be organized in a way that maximizes each person's

skills and abilities. And that's the job of the team leader.

Software Teams

How to lead?

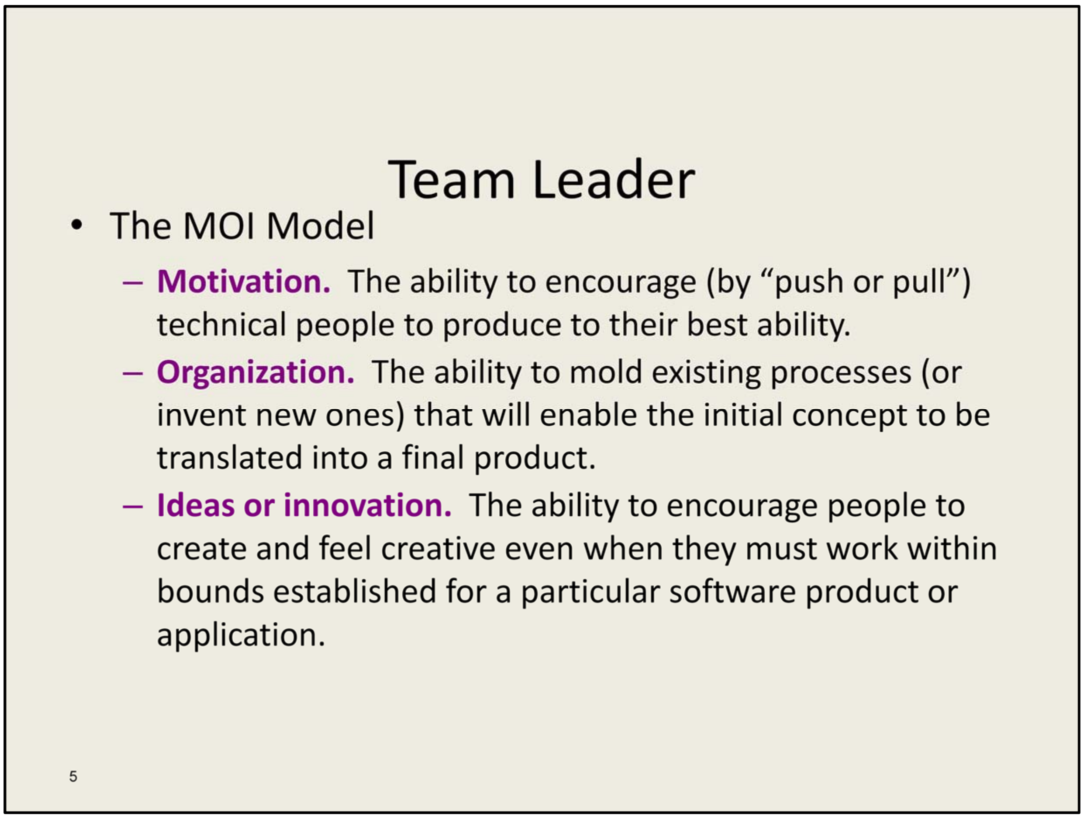
How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

4

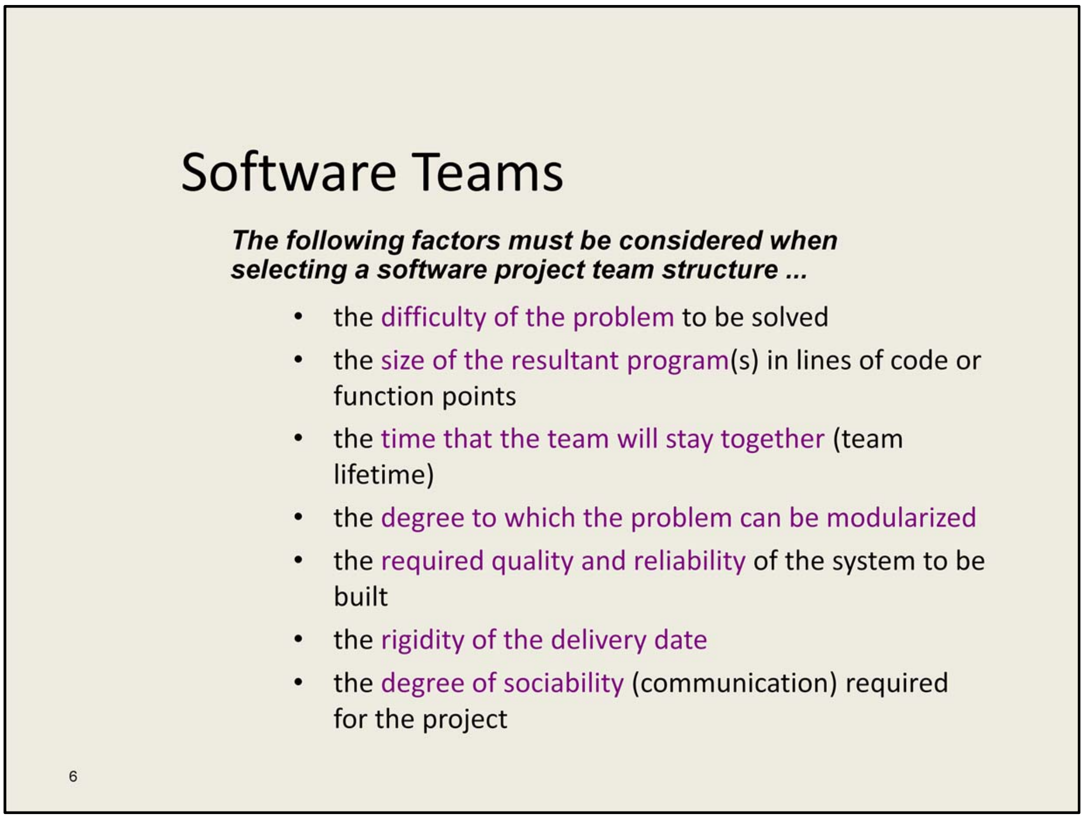


Team Leader

- The MOI Model
  - **Motivation.** The ability to encourage (by "push or pull") 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.

5

successful project leaders apply a problem-solving management
style. That is, a software project manager should concentrate on understanding
the problem to be solved, managing the flow of ideas, and at the same time,
letting everyone on the team know (by words and, far more important, by actions)
that quality counts and that it will not be compromised.



Software Teams

The following factors must be considered when selecting a software project team structure ...

- the difficulty of the problem to be solved
- the size of the resultant program(s) in lines of code or function points
- the time that the team will stay together (team lifetime)
- 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

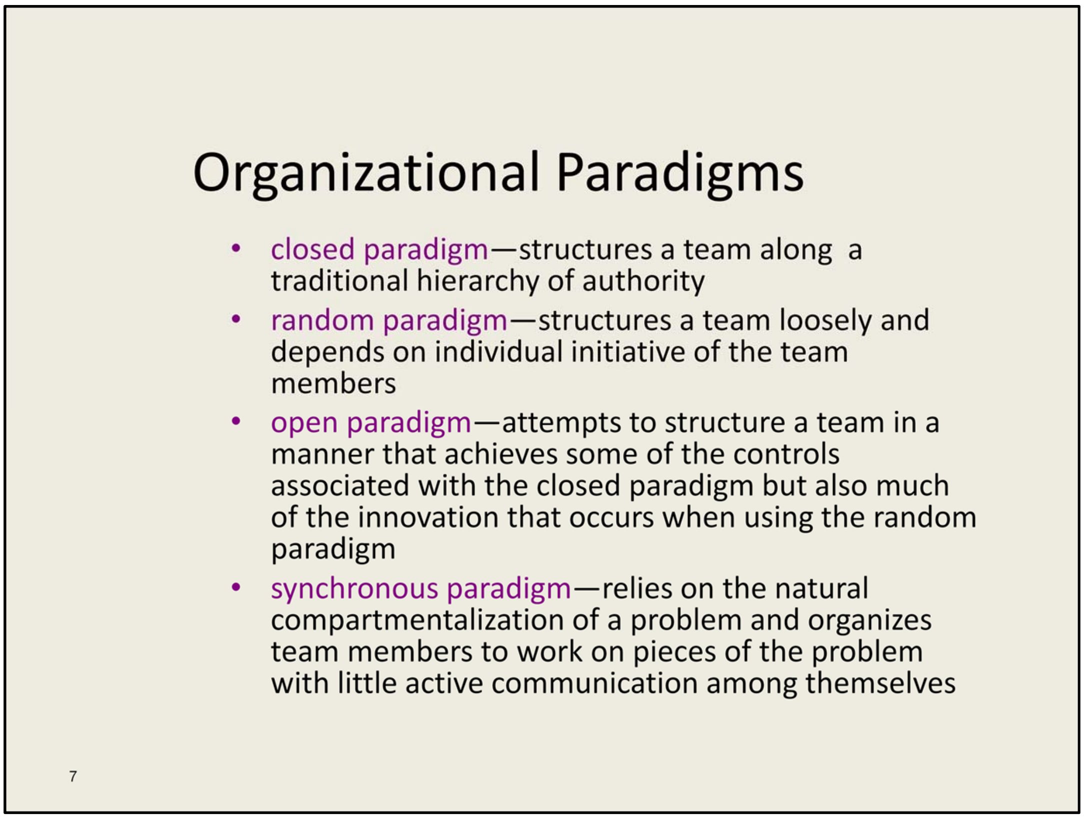
6

The “best” team structure depends on the management style of your organization,

the number of people who will populate the team and their skill levels, and the overall

problem difficulty. Mantei [Man81] describes seven project factors that should be

considered when planning the structure of software engineering teams:

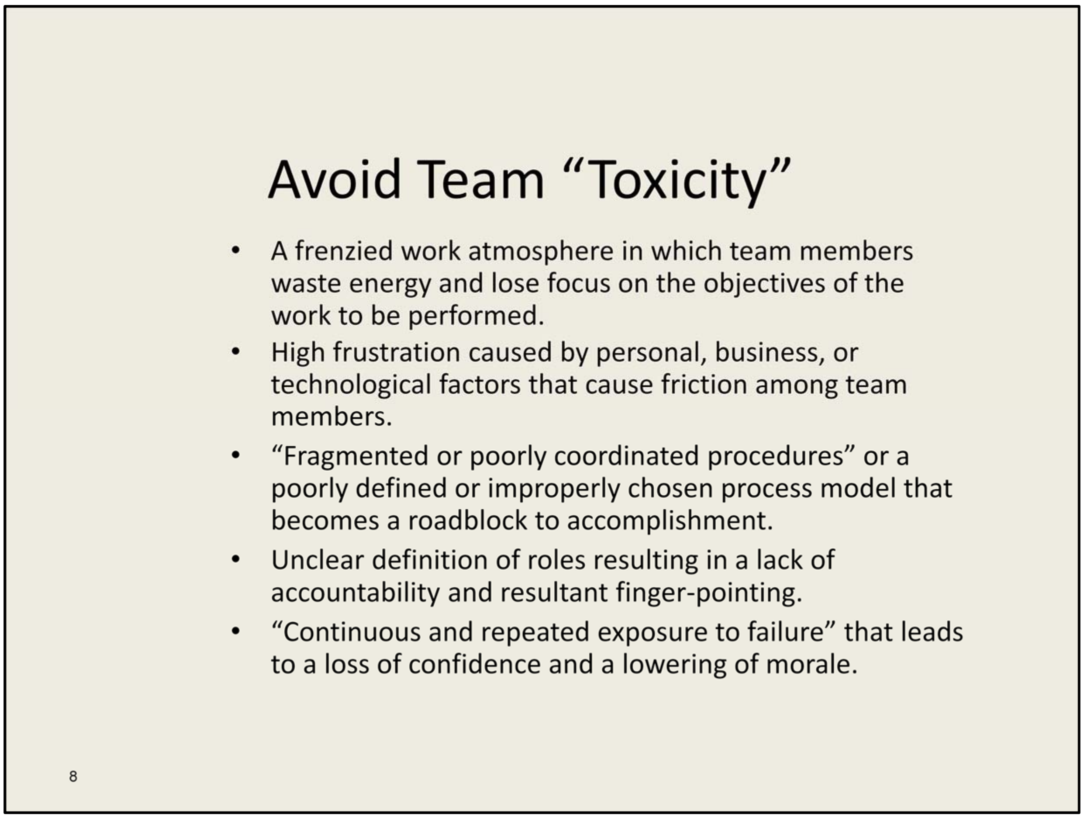


Organizational Paradigms

- closed paradigm—structures a team along a traditional hierarchy of authority
- random paradigm—structures a team loosely and depends on individual initiative of the team members
- open paradigm—attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
- synchronous paradigm—relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

7

As an historical footnote, one of the earliest software team organizations was a closed paradigm structure originally called the *chief programmer team.*



Avoid Team "Toxicity"

- A frenzied work atmosphere in which team members waste energy and lose focus on the objectives of the work to be performed.
- High frustration caused by personal, business, or technological factors that cause friction among team members.
- "Fragmented or poorly coordinated procedures" or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment.
- Unclear definition of roles resulting in a lack of accountability and resultant finger-pointing.
- "Continuous and repeated exposure to failure" that leads to a loss of confidence and a lowering of morale.

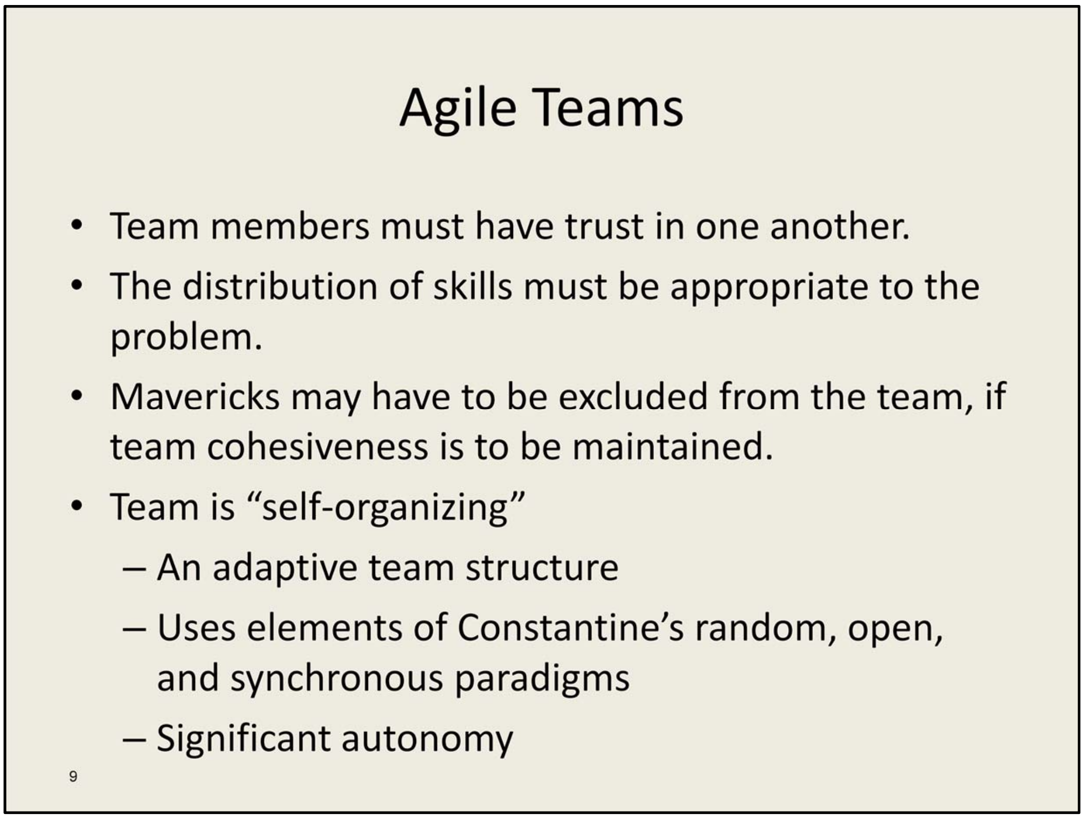
8

But not all teams jell. In fact, many teams suffer from what Jackman [Jac98] calls "team toxicity." She defines five factors that "foster a potentially toxic team environment":

(1) a frenzied work atmosphere, (2) high frustration that causes friction among team members, (3) a "fragmented or poorly coordinated" software process,

(4) an unclear definition of roles on the software team, and (5) "continuous and repeated exposure to failure."

To avoid a frenzied work environment, the project manager should be certain that the team has access to all information required to do the job and that major goals and objectives, once defined, should not be modified unless absolutely necessary. A software team can avoid frustration if it is given as much responsibility for decision making as possible.

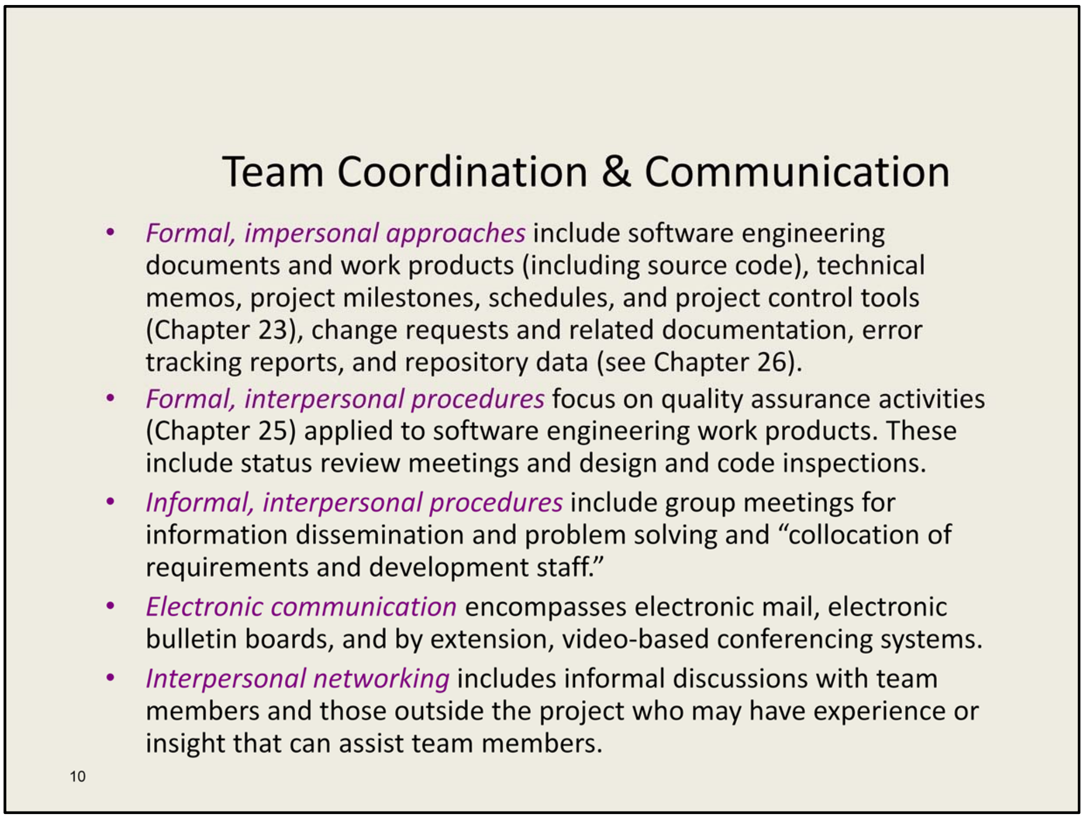


Agile Teams

- Team members must have trust in one another.
- The distribution of skills must be appropriate to the problem.
- Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.
- Team is “self-organizing”
  – An adaptive team structure
  – Uses elements of Constantine’s random, open, and synchronous paradigms
  – Significant autonomy

9

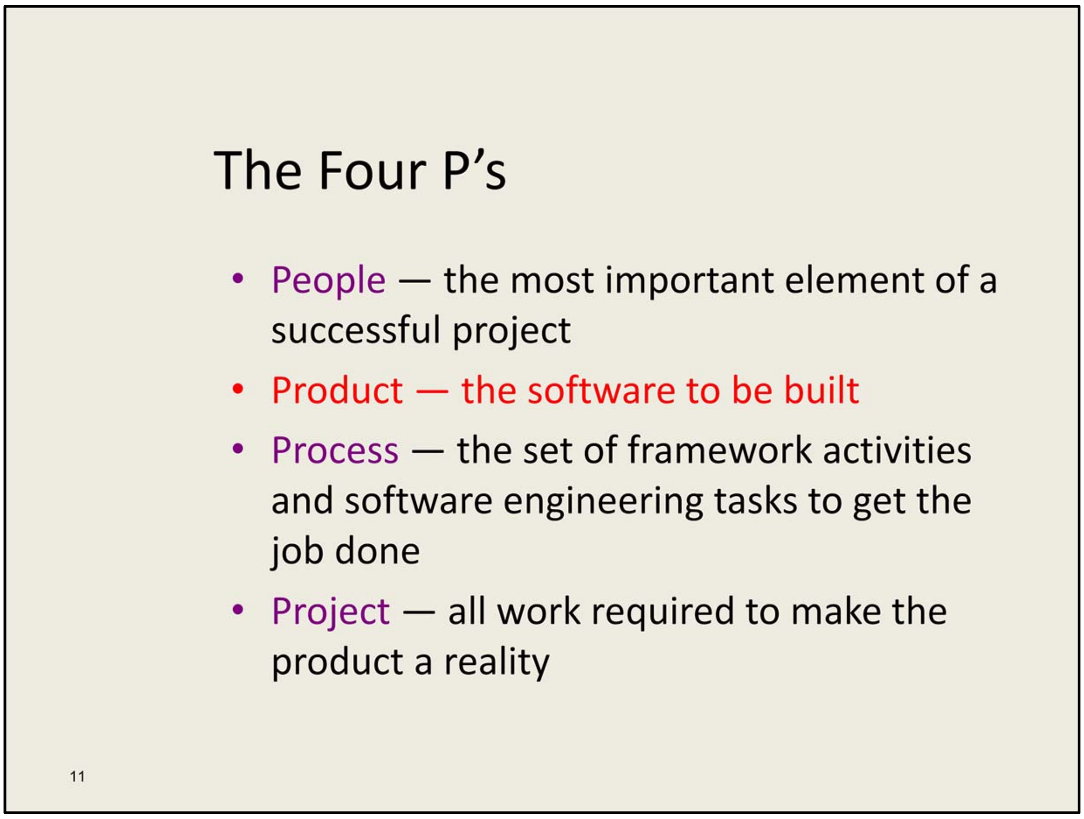
the agile philosophy encourages customer satisfaction and early incremental delivery of software, small highly motivated project teams, informal methods, minimal software engineering work products, and overall development simplicity. The small, highly motivated project team, also called an *agile team,* adopts many of the characteristics of successful software project teams discussed in the preceding section and avoids many of the toxins that create problems. However, the agile philosophy stresses individual (team member) competency coupled with group collaboration as critical success factors for the team.



Team Coordination & Communication

- *Formal, impersonal approaches* include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools (Chapter 23), change requests and related documentation, error tracking reports, and repository data (see Chapter 26).
- *Formal, interpersonal procedures* focus on quality assurance activities (Chapter 25) applied to software engineering work products. These include status review meetings and design and code inspections.
- *Informal, interpersonal procedures* include group meetings for information dissemination and problem solving and “collocation of requirements and development staff.”
- *Electronic communication* encompasses electronic mail, electronic bulletin boards, and by extension, video-based conferencing systems.
- *Interpersonal networking* includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

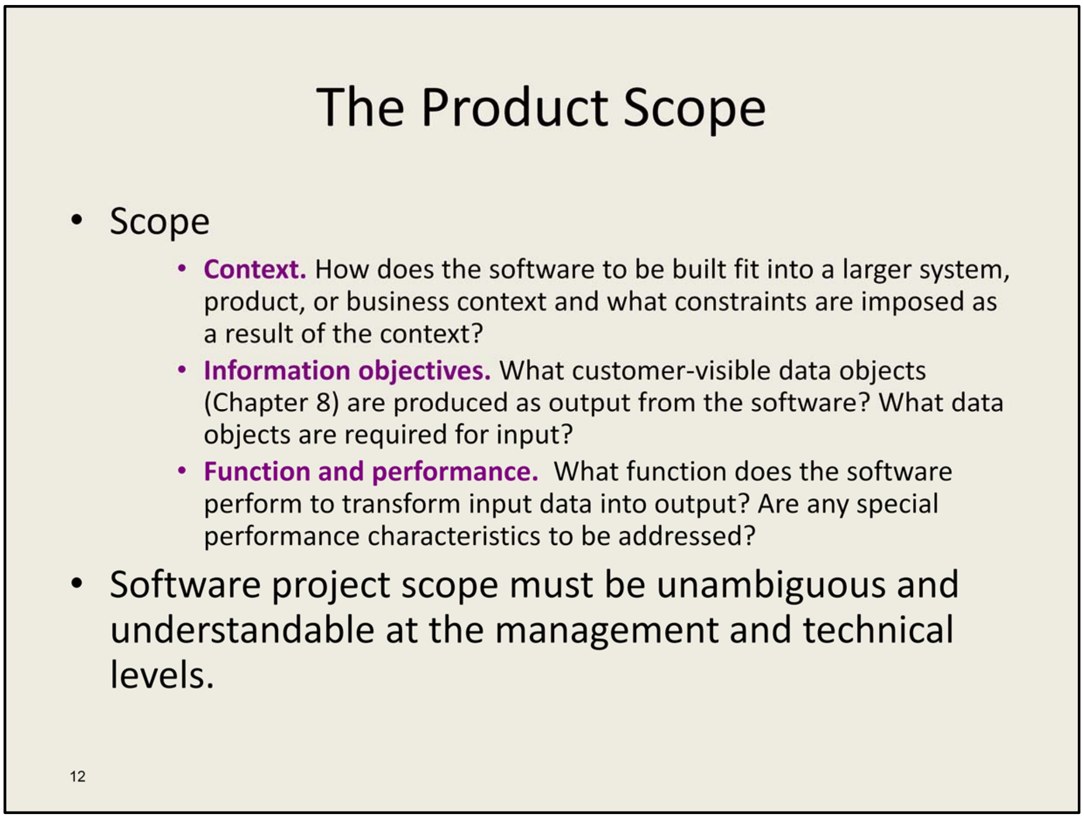
10



The Four P’s

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

11

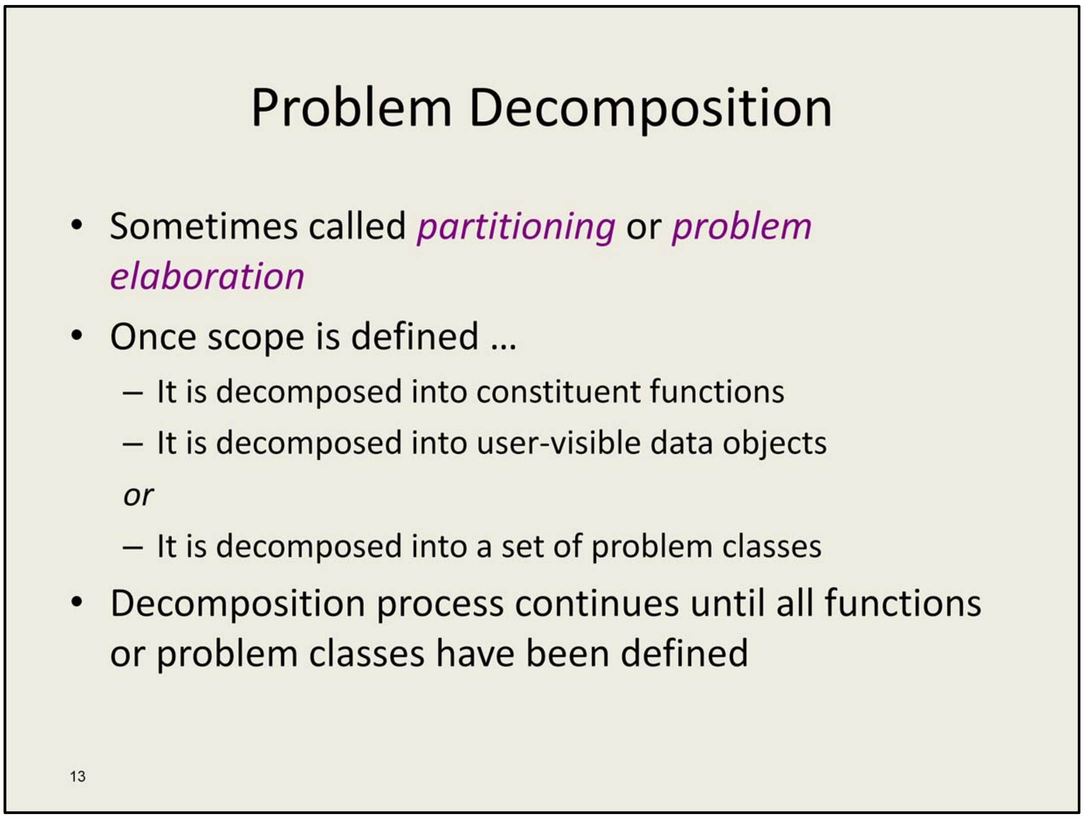


The Product Scope

- Scope
  - **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 (Chapter 8) are produced as output from the software? What data objects are required for input?
  - **Function and performance.** What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?
- Software project scope must be unambiguous and understandable at the management and technical levels.

12

Like it or not, you must examine the product and the problem it is intended to solve at the very beginning of the project. **At a minimum, the scope** of the product
must be established and bounded.



Problem Decomposition

- Sometimes called *partitioning* or *problem elaboration*
- Once scope is defined …
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects

  or
  - It is decomposed into a set of problem classes
- Decomposition process continues until all functions or problem classes have been defined

13

Problem decomposition, sometimes called *partitioning* or *problem elaboration,* is an

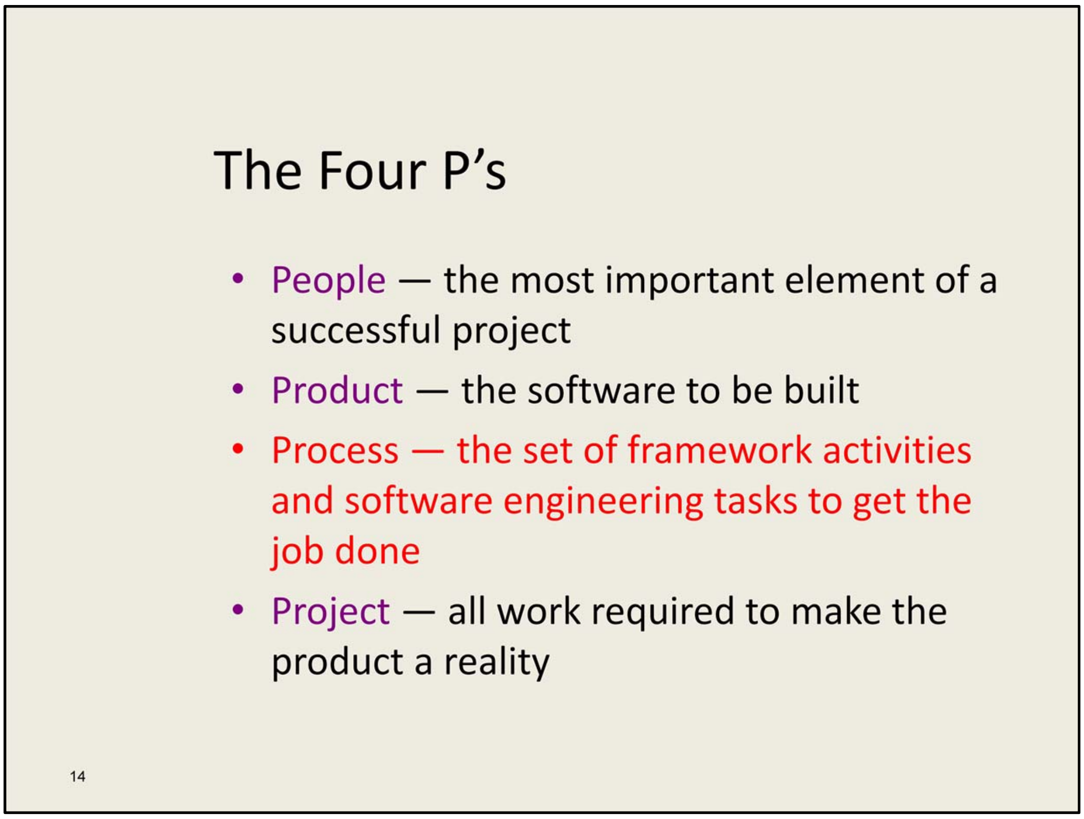
activity that sits at the core of software requirements analysis (Chapters 6 and 7).

During the scoping activity no attempt is made to fully decompose the problem.

Rather, decomposition is applied in two major areas: (1) the functionality and

content (information) that must be delivered and (2) the process that will be used to

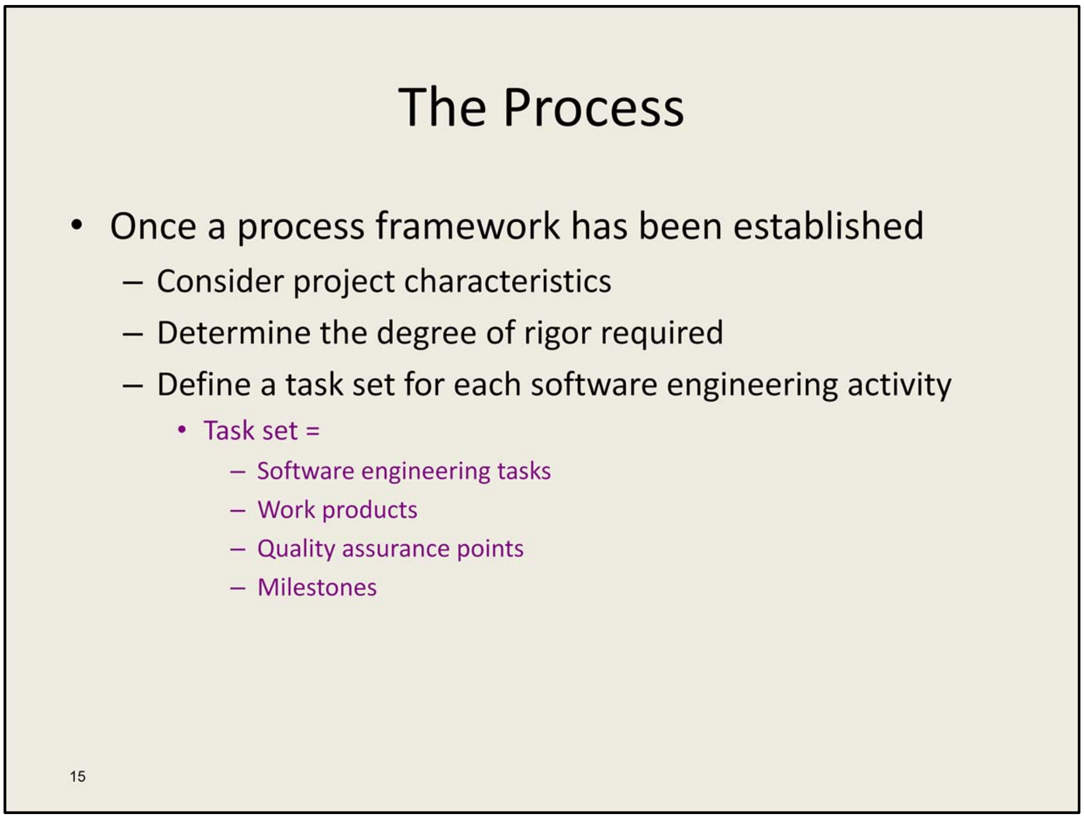
deliver it.



The Four P's

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

14



The Process

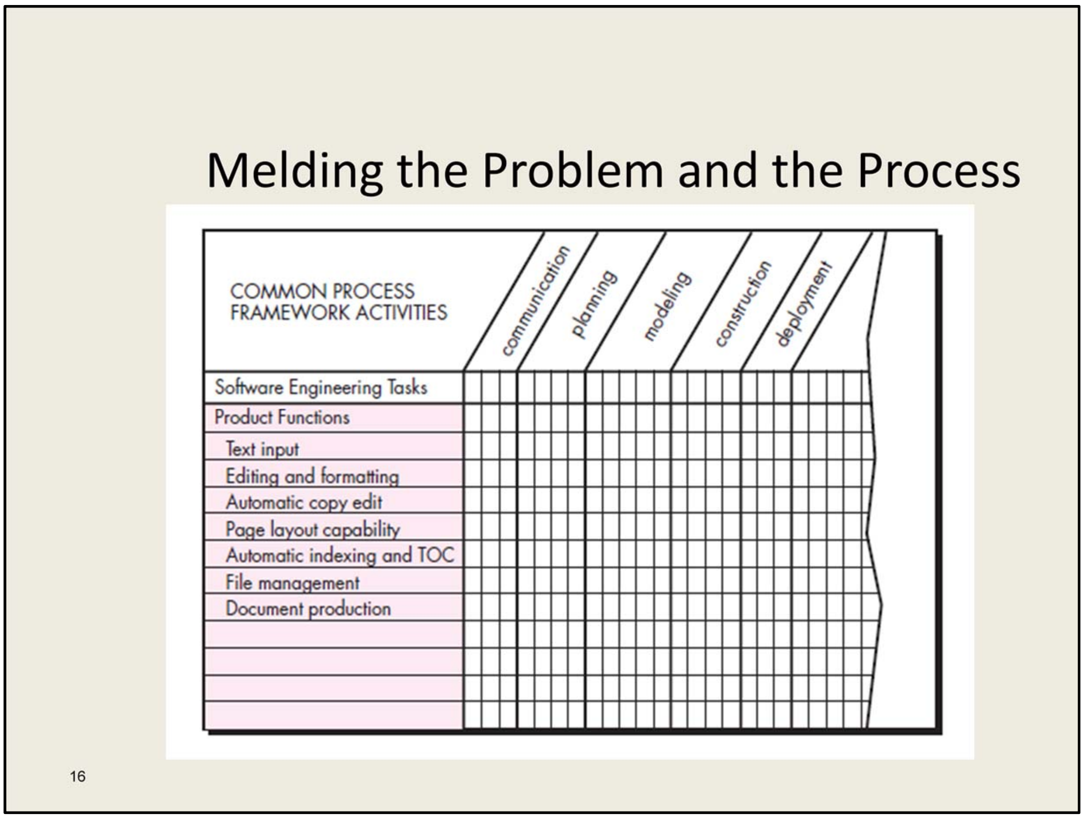
- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones

15

The framework activities (Chapter 2) that characterize the software process are

applicable to all software projects. The problem is to select the process model that is

appropriate for the software to be engineered by your project team.



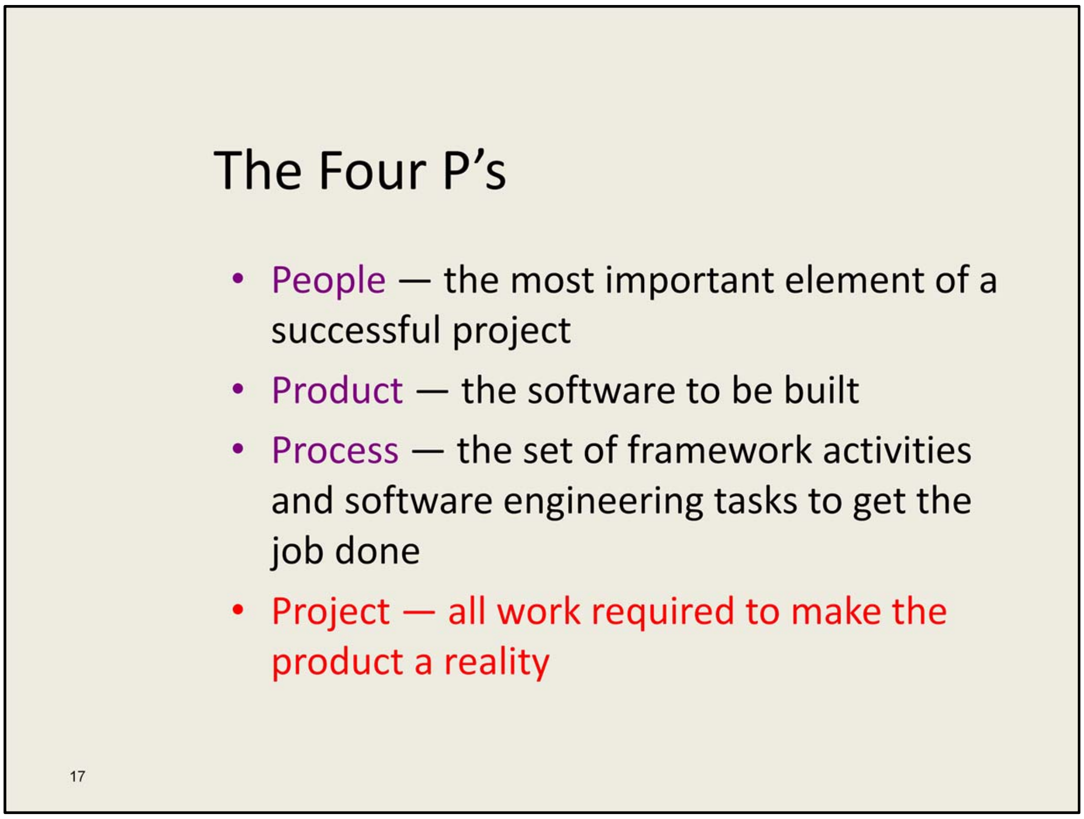
Melding the Problem and the Process

COMMON PROCESS FRAMEWORK ACTIVITIES	communication	planning	modeling	construction	deployment
Software Engineering Tasks					
Product Functions					
Text input					
Editing and formatting					
Automatic copy edit					
Page layout capability					
Automatic indexing and TOC					
File management					
Document production					

16

Project planning begins with the melding of the product and the process. Each function
to be engineered by your team must pass through the set of framework activities
that have been defined for your software organization.

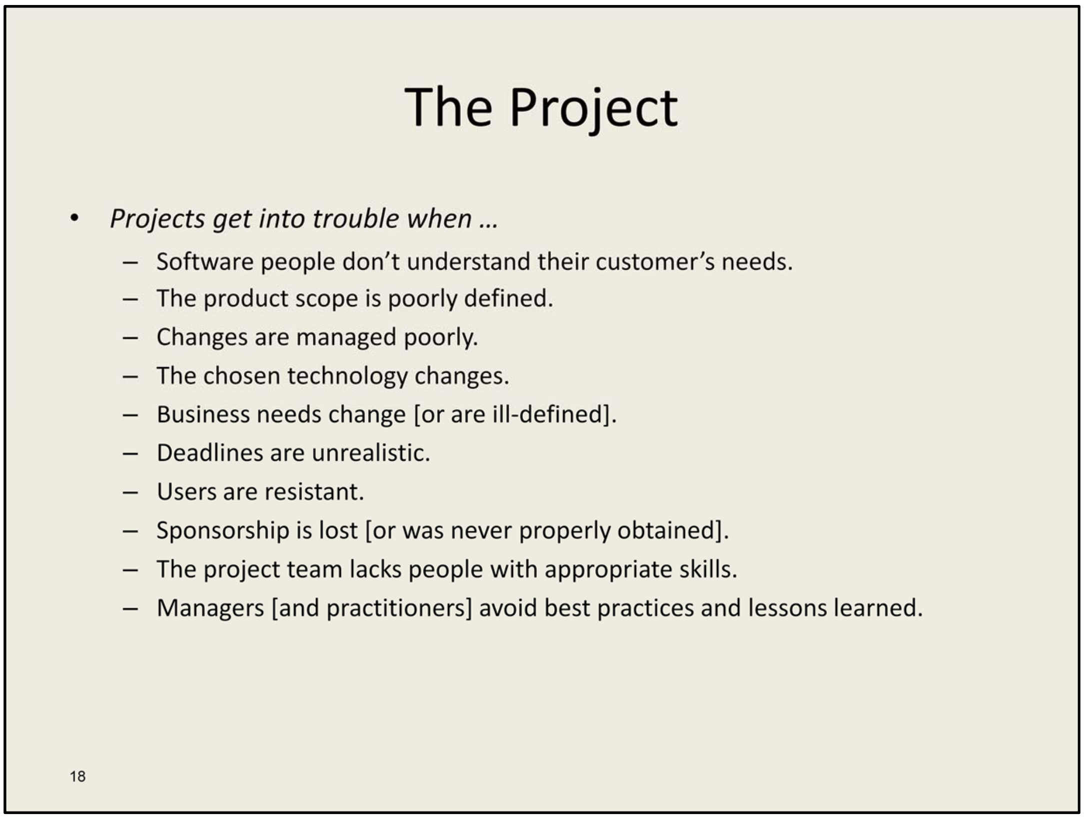
Each major product function (the figure notes functions
for the word-processing software discussed earlier) is listed in the left-hand
column. Framework activities are listed in the top row. Software engineering work
tasks (for each framework activity) would be entered in the following row.5 The job
of the project manager (and other team members) is to estimate resource requirements
for each matrix cell, start and end dates for the tasks associated with each cell,
and work products to be produced as a consequence of each task.



The Four P's

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

17


The Project

- *Projects get into trouble when …*
  - 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 ill-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.

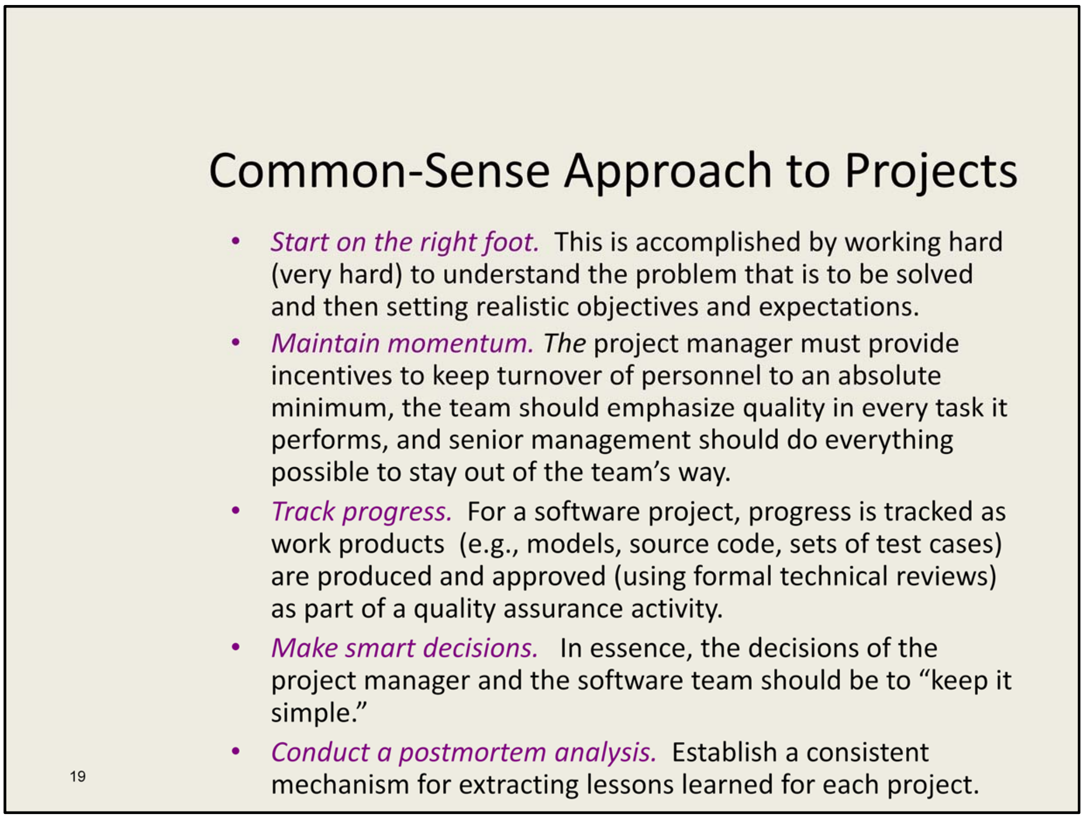
18

In order to manage a successful software project, you have to understand what can

go wrong so that problems can be avoided. In an excellent paper on software projects,

John Reel [Ree99] defines 10 signs that indicate that an information systems

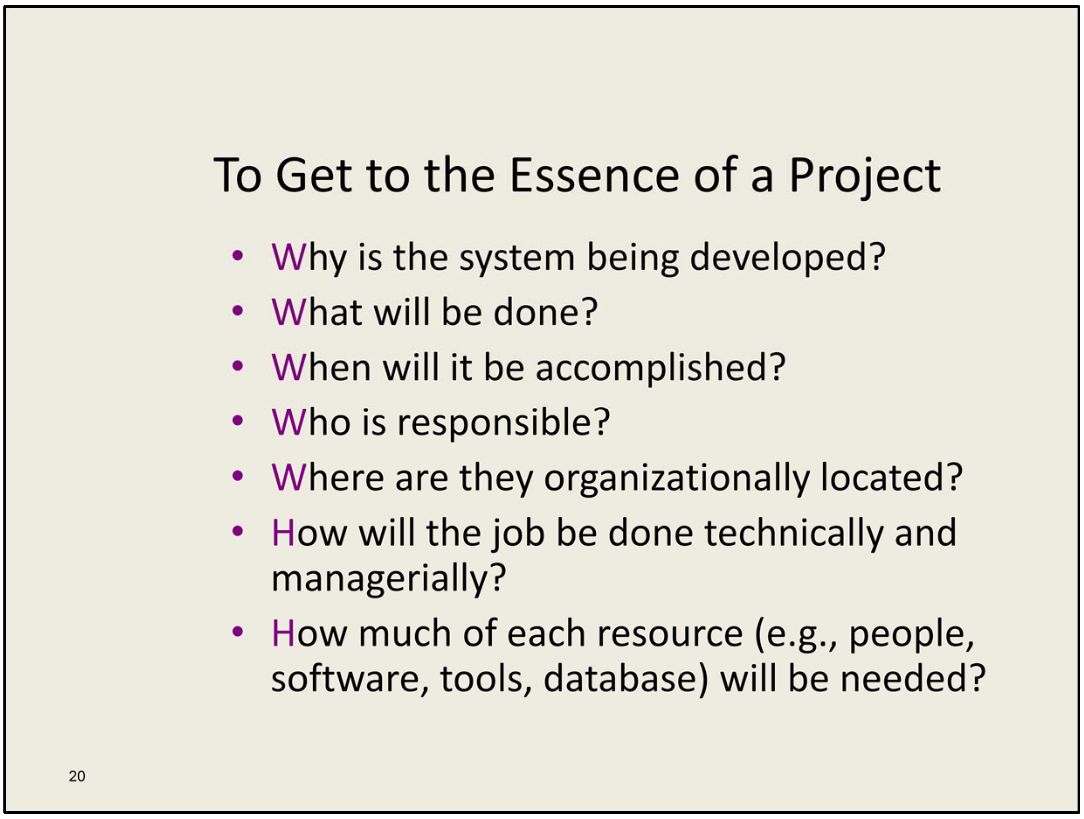
project is in jeopardy:



Common-Sense Approach to Projects

- *Start on the right foot.* This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.
- *Maintain momentum. The* project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.
- *Track progress.* For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.
- *Make smart decisions.* In essence, the decisions of the project manager and the software team should be to "keep it simple."
- *Conduct a postmortem analysis.* Establish a consistent mechanism for extracting lessons learned for each project.

19



To Get to the Essence of a Project

- Why is the system being developed?
- What will be done?
- When will it be accomplished?
- Who is responsible?
- Where are they organizationally located?
- How will the job be done technically and managerially?
- How much of each resource (e.g., people, software, tools, database) will be needed?

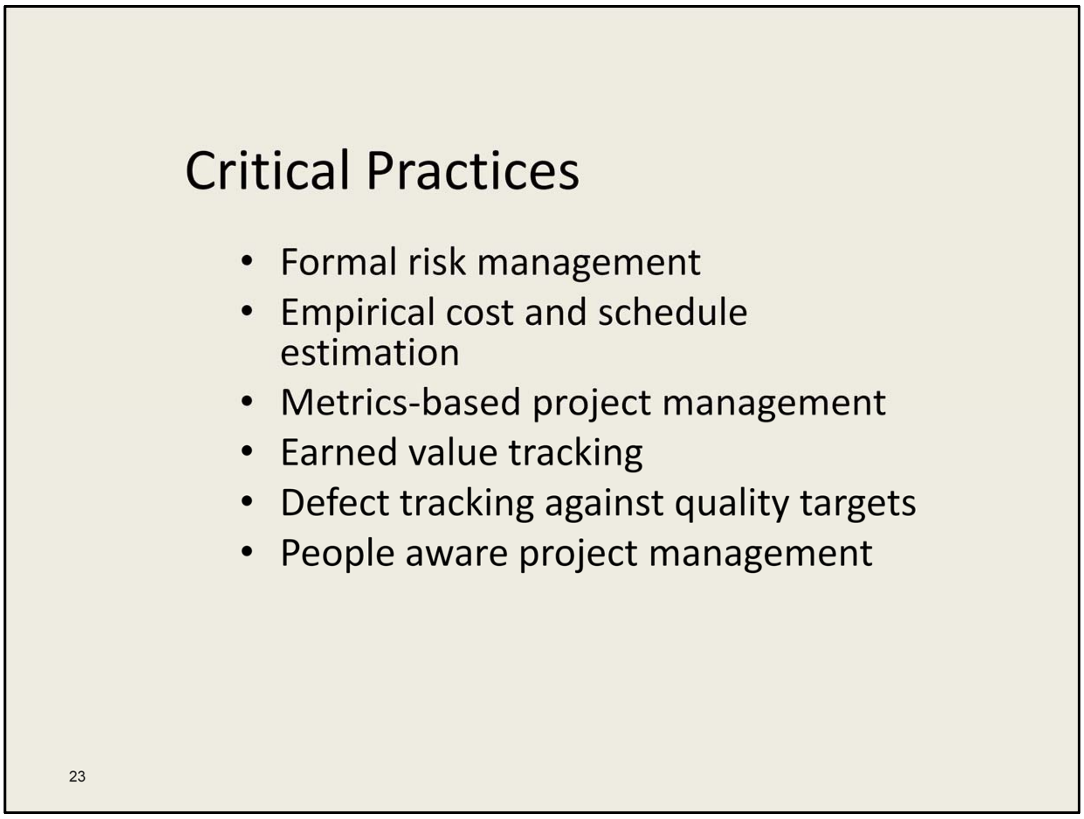
20

W^5HH Principle

- Boehm [BOE96] suggests a question-based approach that addresses project objectives, milestones and schedules, responsibilities, management and technical approaches, and required resources.
  - Why is the system being developed?
    - The answer to this question enables all parties to assess the validity of business reasons for the software work. Stated in another way, does the business purpose justify the expenditure of people, time, and money?
  - What will be done?
  - When will it be accomplished?
    - The answers to these two questions help the team to establish a project schedule by identifying key project tasks and the milestones that are required by the customer.

W^{5}HH Principle

- Who is responsible for a function?
  - The responsibilities of each member of the software team must be defined.
- Where are they organizationally located?
  - Not all roles and responsibilities reside within the software team itself. The customer, users, and other stakeholders also have responsibilities.
- How will the job be done technically and managerially?
  - Once product scope is established, a management and technical strategy for the project must be defined.
- How much of each resource (e.g., people, software, tools, database) will be needed?
  - The answer to this question is derived by developing estimates based on answers to earlier questions.



Critical Practices

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

23

The Airlie Council8 has developed a list of “critical software practices for performance-based management.” These practices are “consistently used by, and considered critical by, highly successful software projects and organizations whose
’bottom line’ performance is consistently much better than industry averages” [Air99].

Summary

- Software project management is about people, product, process, and project. It includes,
  - Team building,
  - Requirement solicitation and engineering,
  - Process model adapted to the people and problem,
  - Project organized in a manner that enables the software team to succeed.
- Remember, project management is a people business,
  - Emphasize on the communication: formal reviews and informal person-to-person communication have the most value for practitioners.
  - The end note to an engineering manager: if you take care of your people, they will take care of the work.