Session 5

Project Management

Definitions

- **Management**
  - The **activities** and tasks undertaken by **one** or **more** persons for the purpose of **planning and controlling** the activities of **other** in order to achieve **objectives** that **could not** be achieved by the others acting alone.

- **Project Management**
  - A **system** of management procedures, practices, technologies, skills, and experience necessary to successfully manage a project

- **Software Engineering Project Management**
  - Project management where the product is software

Project vs. Acquisition

- Two types of software projects
  - Development projects
  - Acquisition projects

- Software Development
  - The process of developing this system to deliver on time and within budget

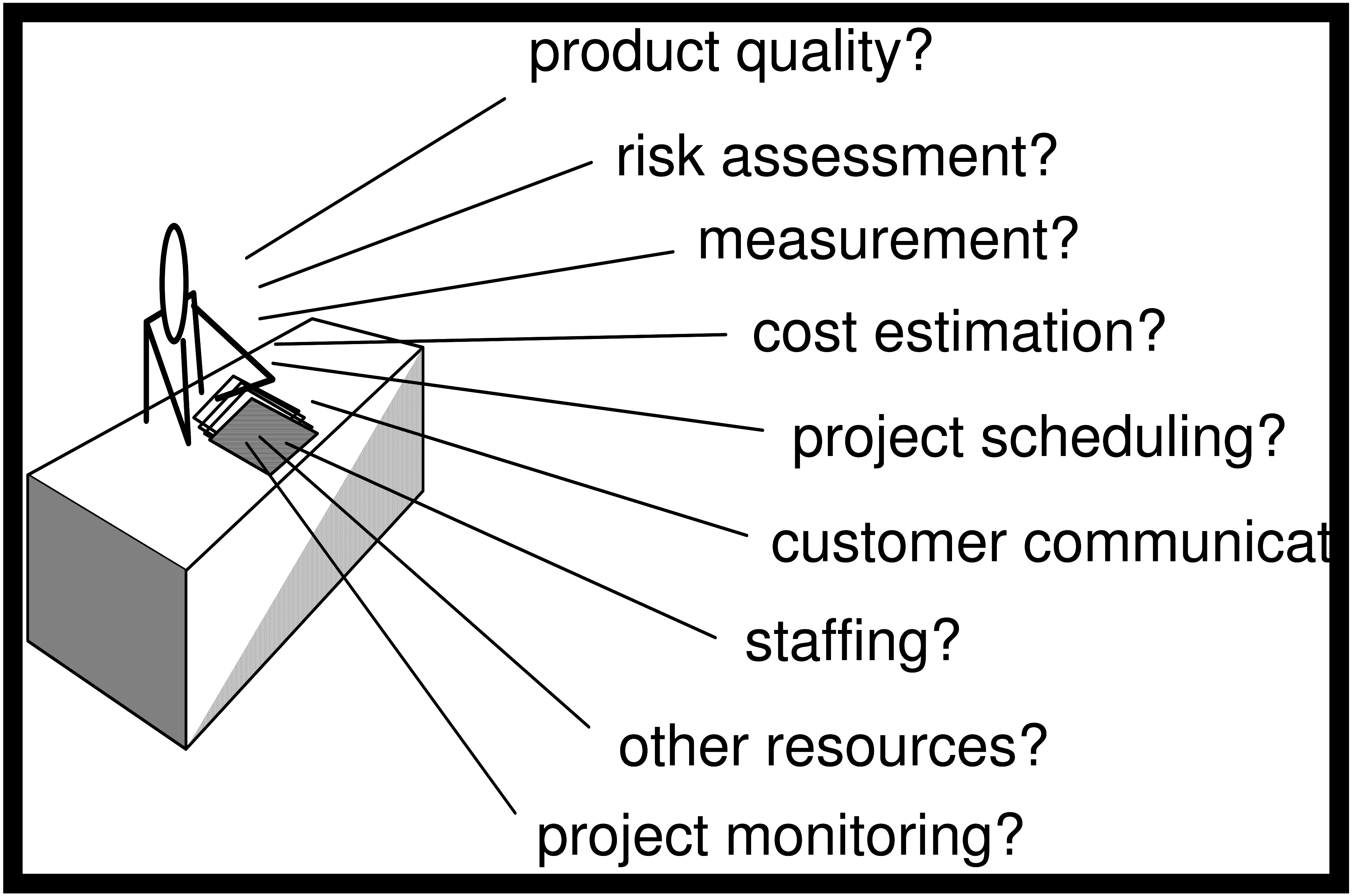
- Software Acquisition
  - The process of contractually obtaining a software-intensive system from a third party, for the use of **another groups** or organization

Project Management

- Software Projects – **factors** that **influence** results:

  - size
  - delivery deadline
  - user requirements
  - available resources
  - budgets and costs
  - application domain
  - technology to be implemented
  - system constraints

Project Management Concerns



product quality?

risk assessment?

measurement?

cost estimation?

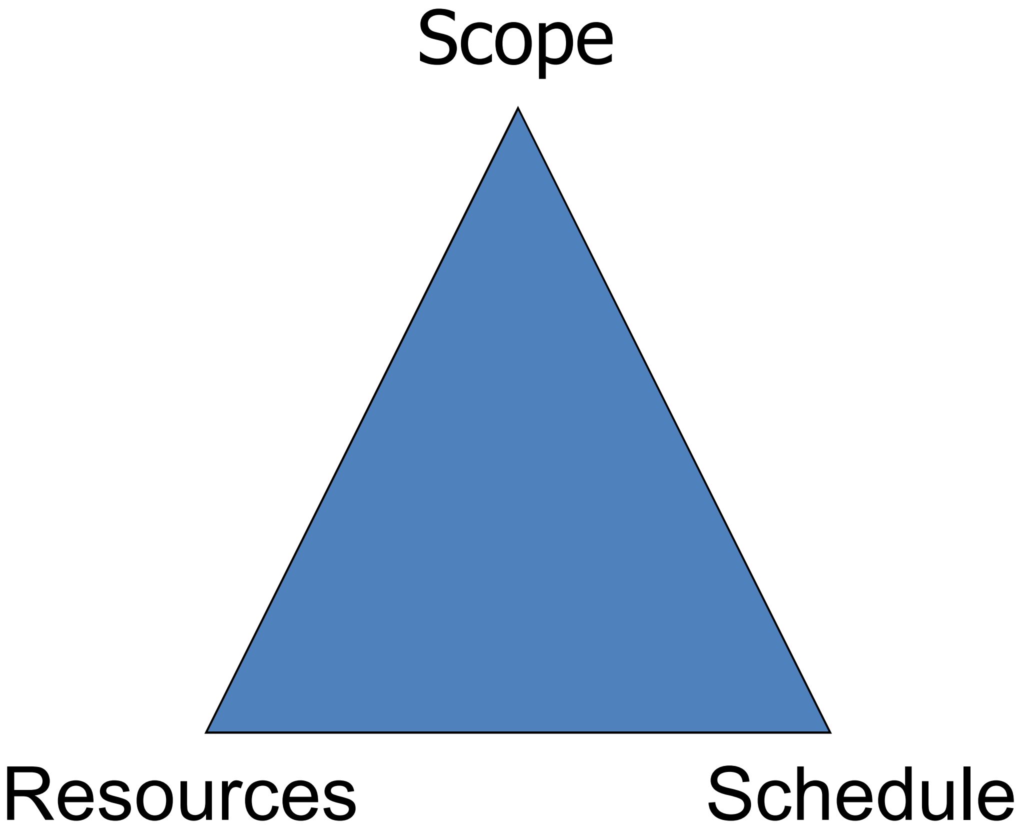
project scheduling?

customer communicat

staffing?

other resources?

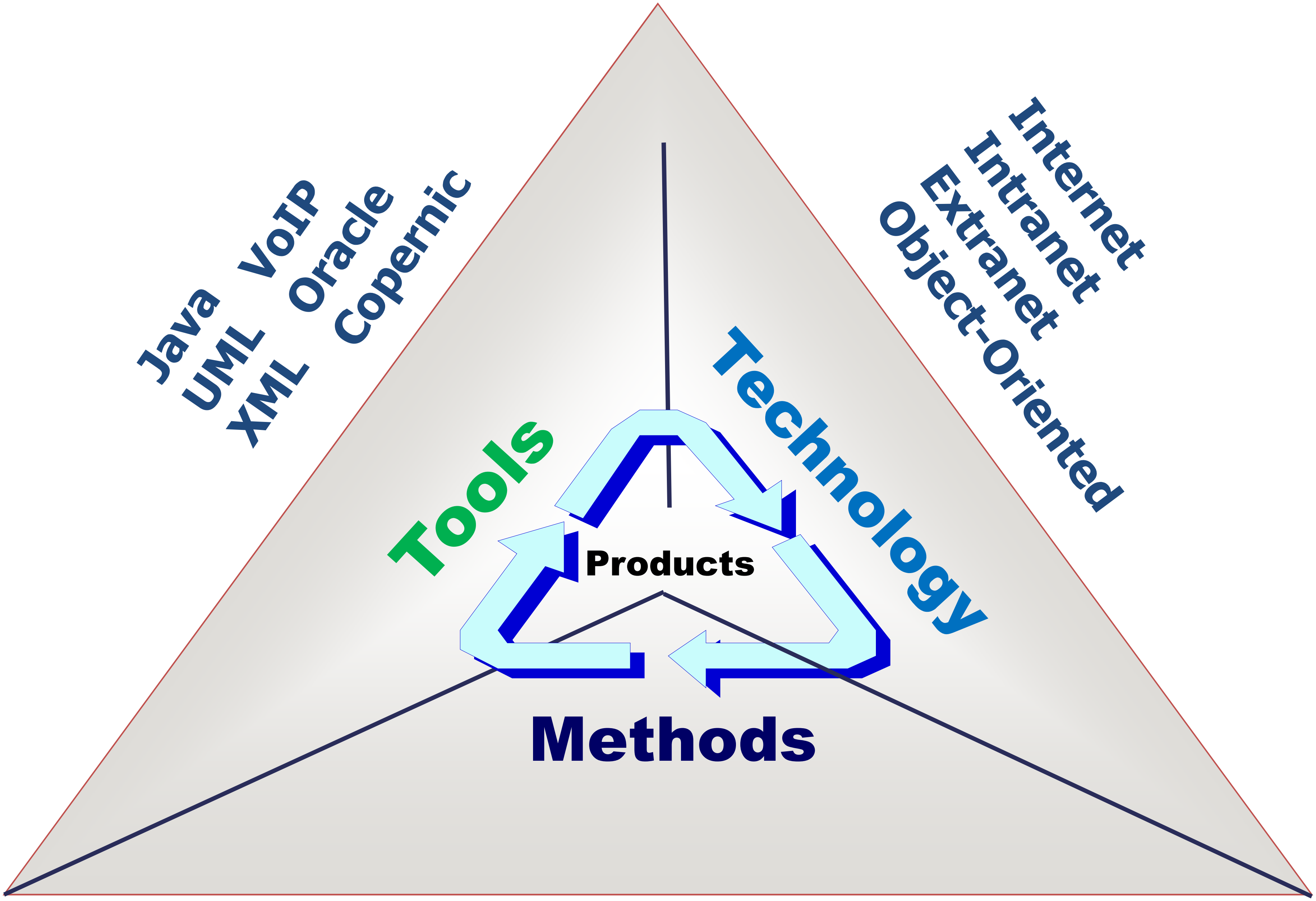
project monitoring?



Scope

Resources

Schedule



Java VoIP

UML Oracle

XML Copernic

Internet

Intranet

Extranet

Object-Oriented

Tools

Technology

Products

Methods

Software Engineering) (Project Management

Why Projects Fail?

- changing customer requirements
- ambiguous/incomplete requirements
- an unrealistic deadline is established
- an honest underestimate of effort
- predictable and/or unpredictable risks
- technical difficulties
- miscommunication among project staff
- failure in project management

The Software Team Organization

- Democratic decentralized **(DD)**
- Controlled decentralized **(CD)**
- Controlled centralized **(CC)**

The Software Team Organization

Democratic decentralized (DD)

- No permanent leader
- Task coordinators are appointed for short duration
- Decisions on problems and approach are made by group consensus
- Communication among team is horizontal

Controlled decentralized (CD)

- Defined leader who coordinates specific tasks
- Problem solving remains a group activity
- Communication among subgroups and individuals is horizontal. Vertical communication along the control hierarchy also occurs.

Controlled centralized (CC)

- Top level problem solving and internal team coordination are managed by the team leader
- Communication between the leader and team members is vertical.

The Management Spectrum (P4)

Effective project management has focus 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

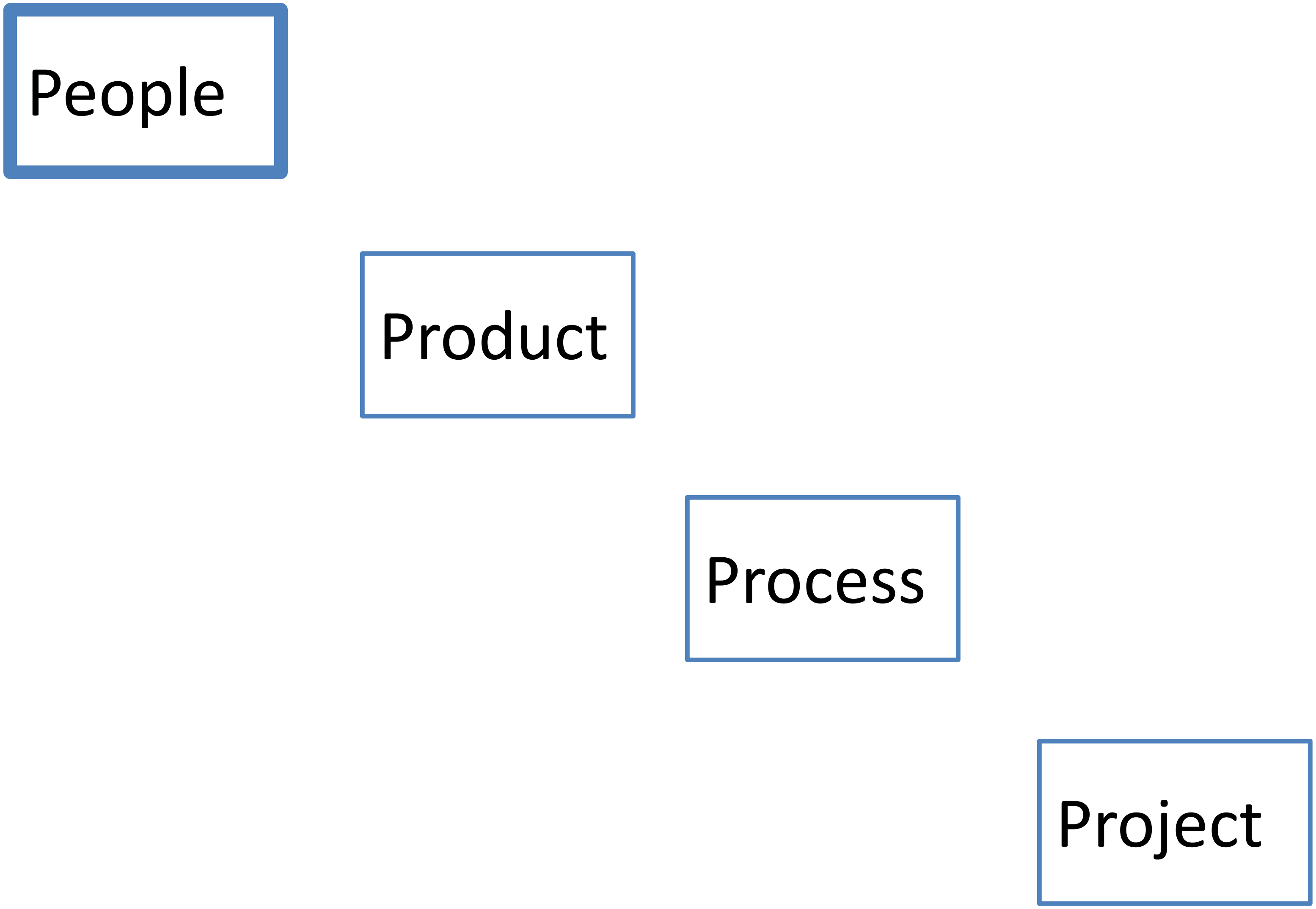
- Product objectives and scope should be established before a project can be planned

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 2014 survey, 39% 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 of a team leader
- **MOI:**
  - **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

The People: Team Leaders

- 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

  - **Quality control** - Let everyone on the team know, by words and actions, that quality counts and that it will not be compromised

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

The People: The Software Team

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**

The Software Team-Team Toxicity

- **Five factors that cause team toxicity:**
  - 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

- 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

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

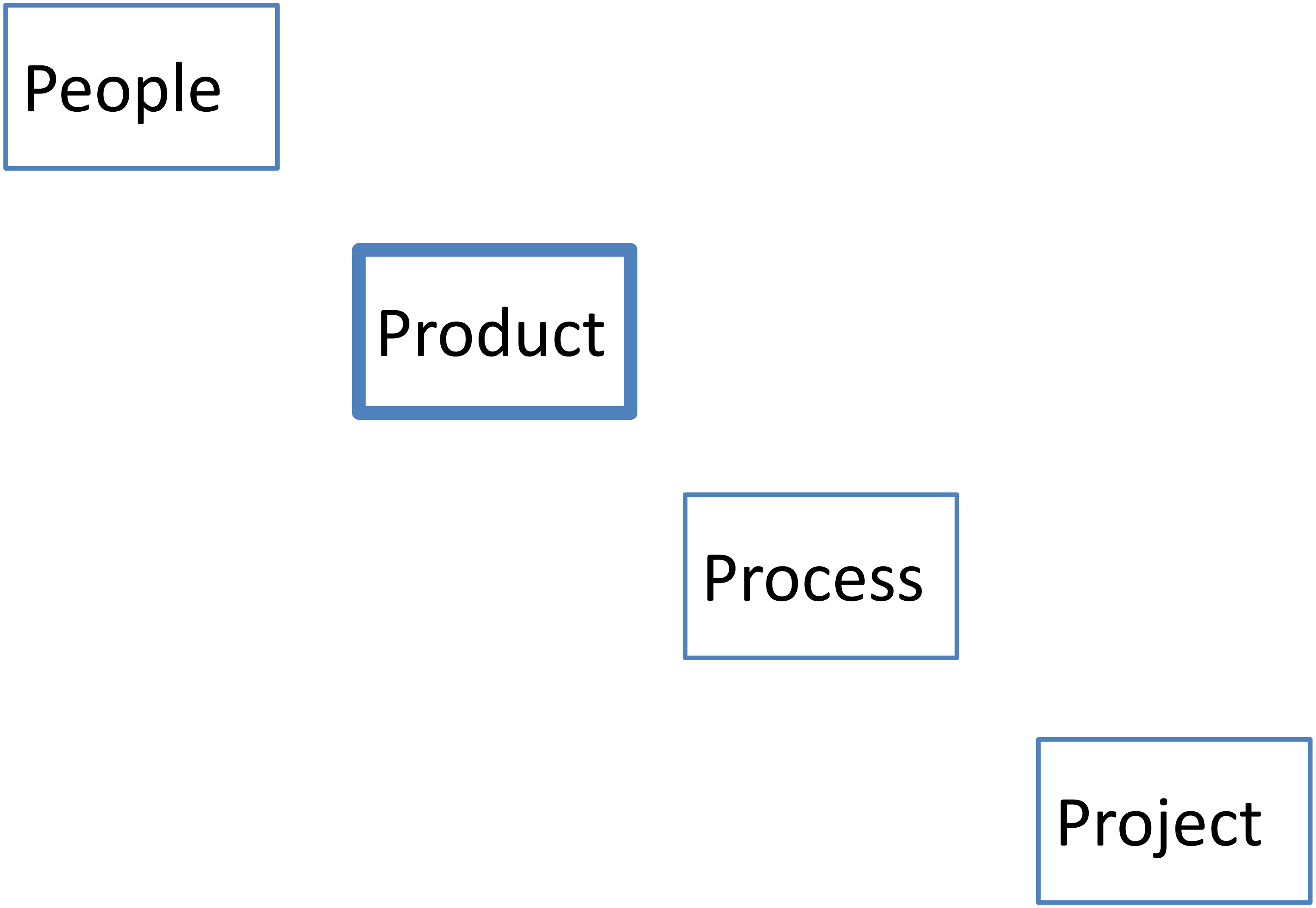
- **Storming**
  - As group members organize for the tasks, **conflict inevitably results** in their personal relations and cliques start to form
  - Individuals have to **bend** and **mold** their feelings to fit the group
  - **Fear** of exposure or fear of **failure, causes** an **increased desire 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** dissolves
  - 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 **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?**

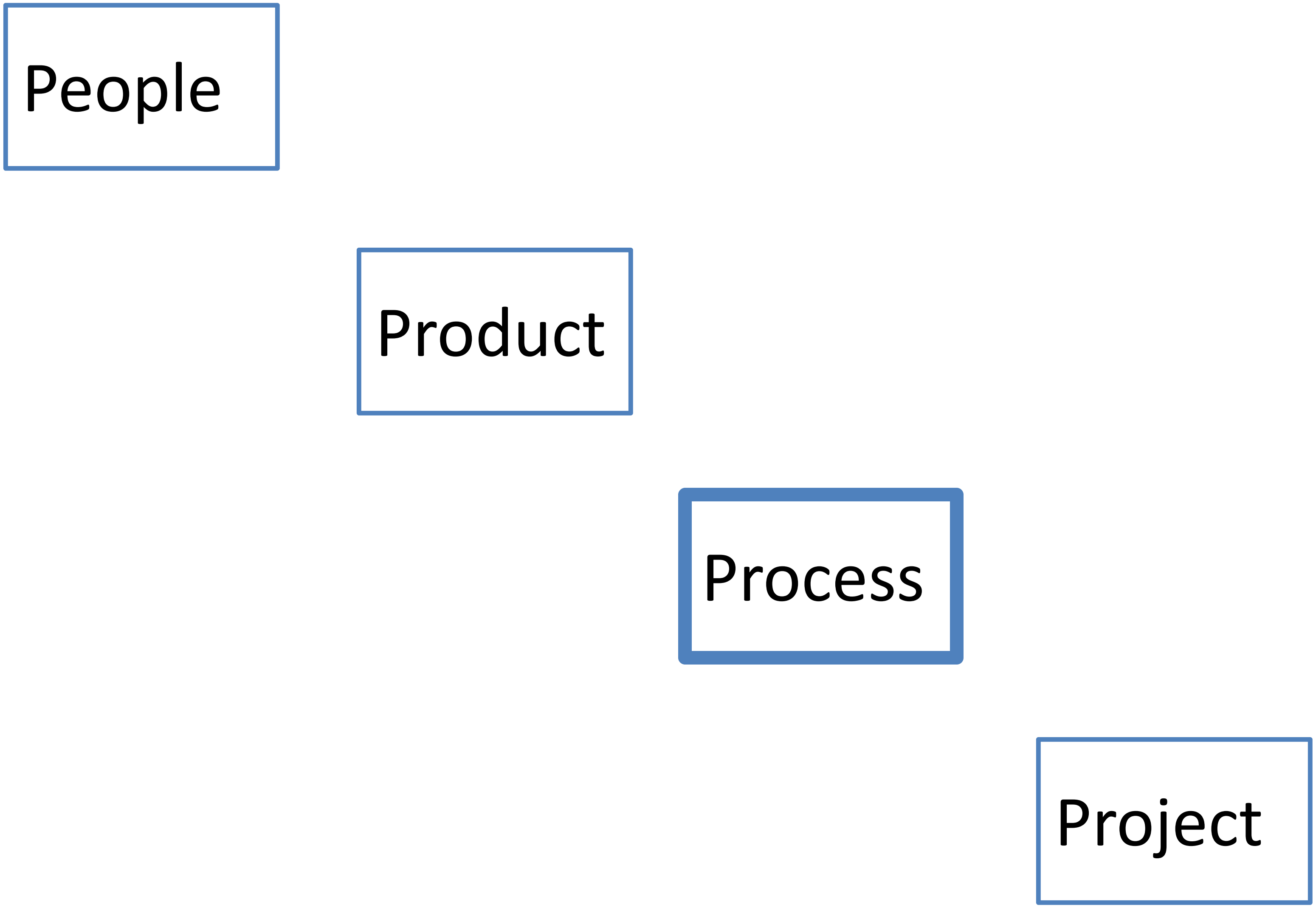
Information objectives:

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 **un**ambiguous and **un**derstandable at both the managerial and technical levels



People

Product

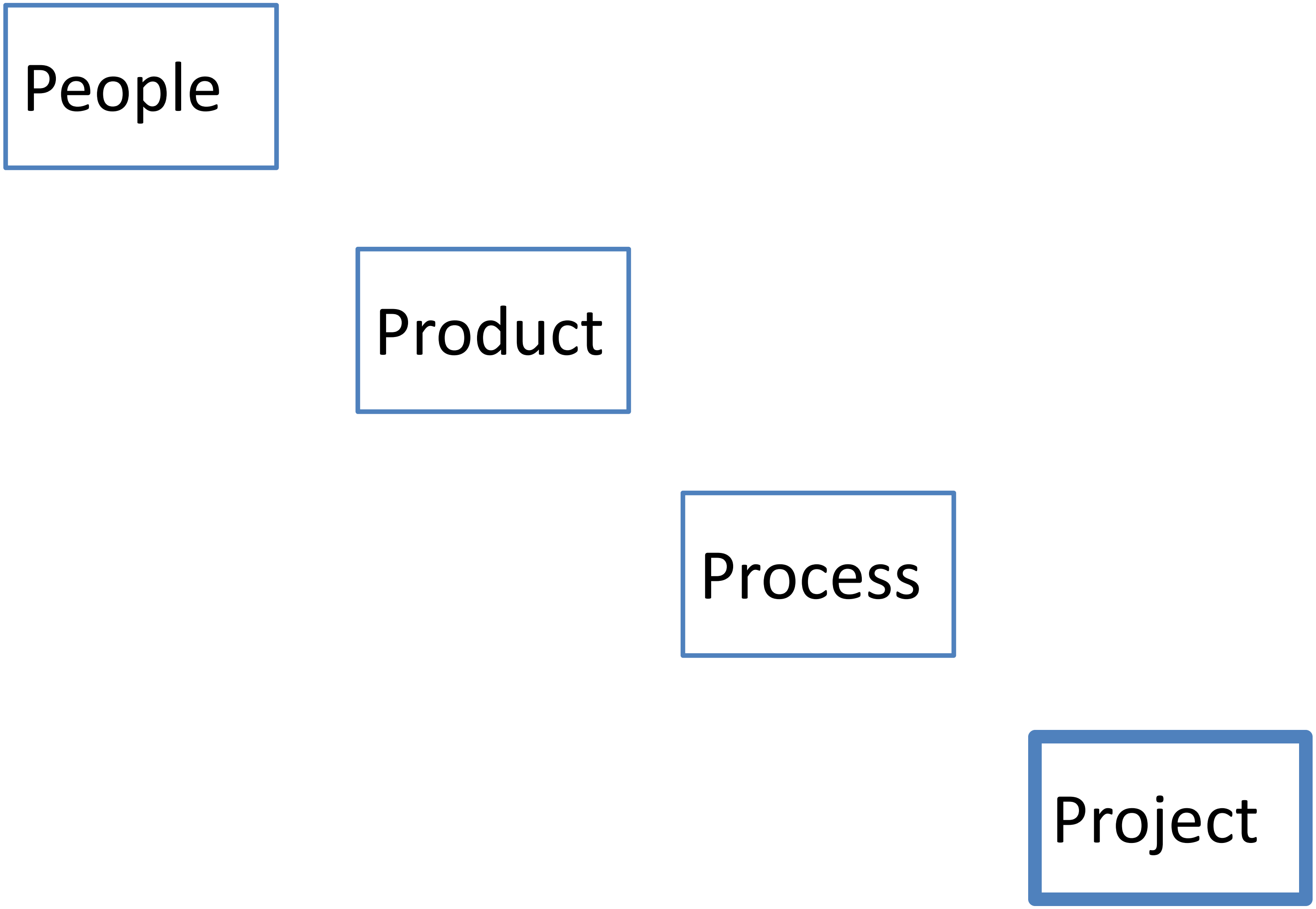
Process

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**



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

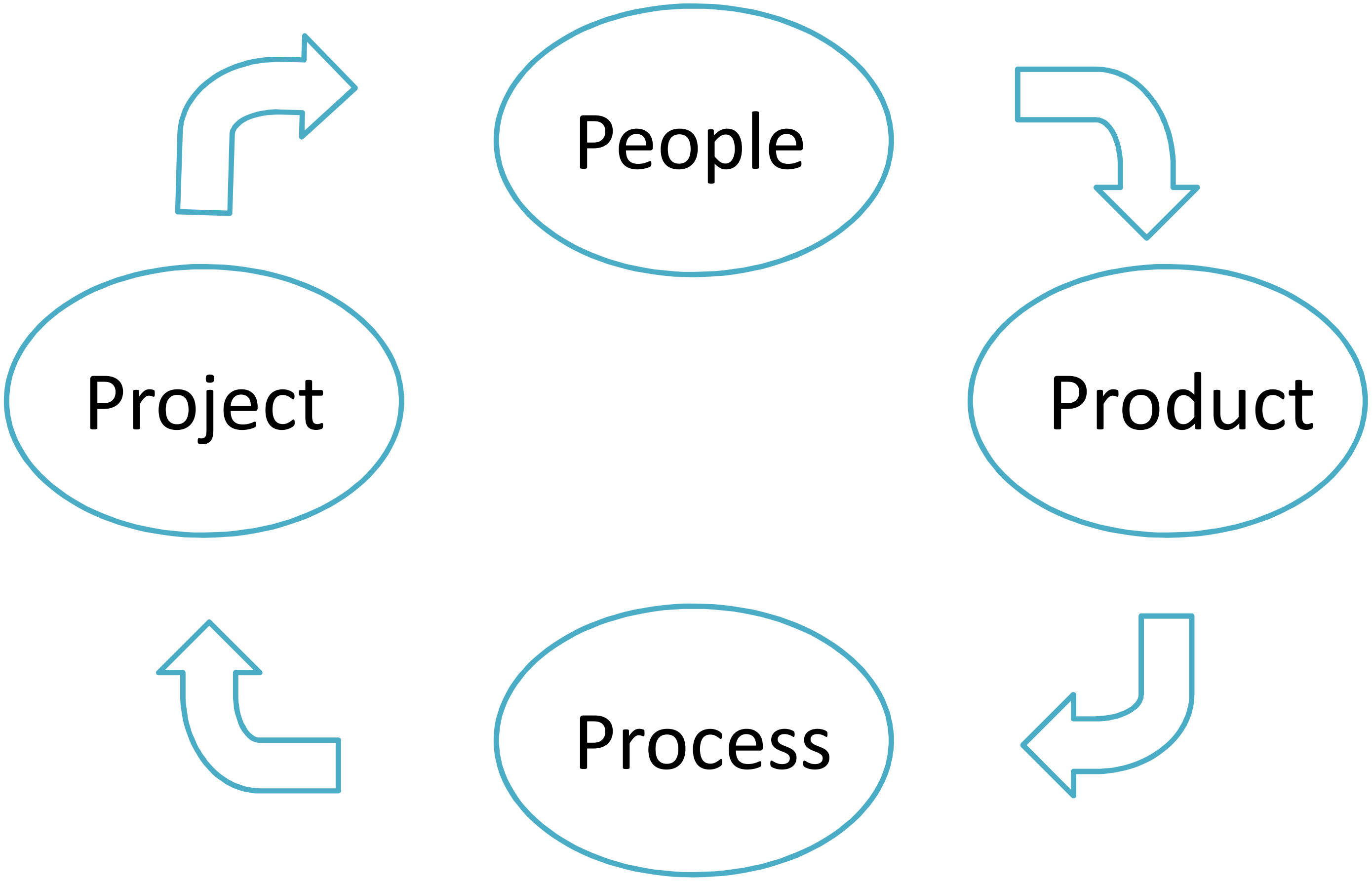
- Software people don't understand their customer's needs
- The product scope is poorly defined
- Changes are managed poorly
- The chosen technology changes
- Business needs change (or are poorly defined)
- Deadlines are unrealistic
- Users are resistant
- Sponsorship is lost (or was never properly obtained)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

The Project: The W^5HH 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


People

Product

Process

Project

Example: Bar charts and activity networks

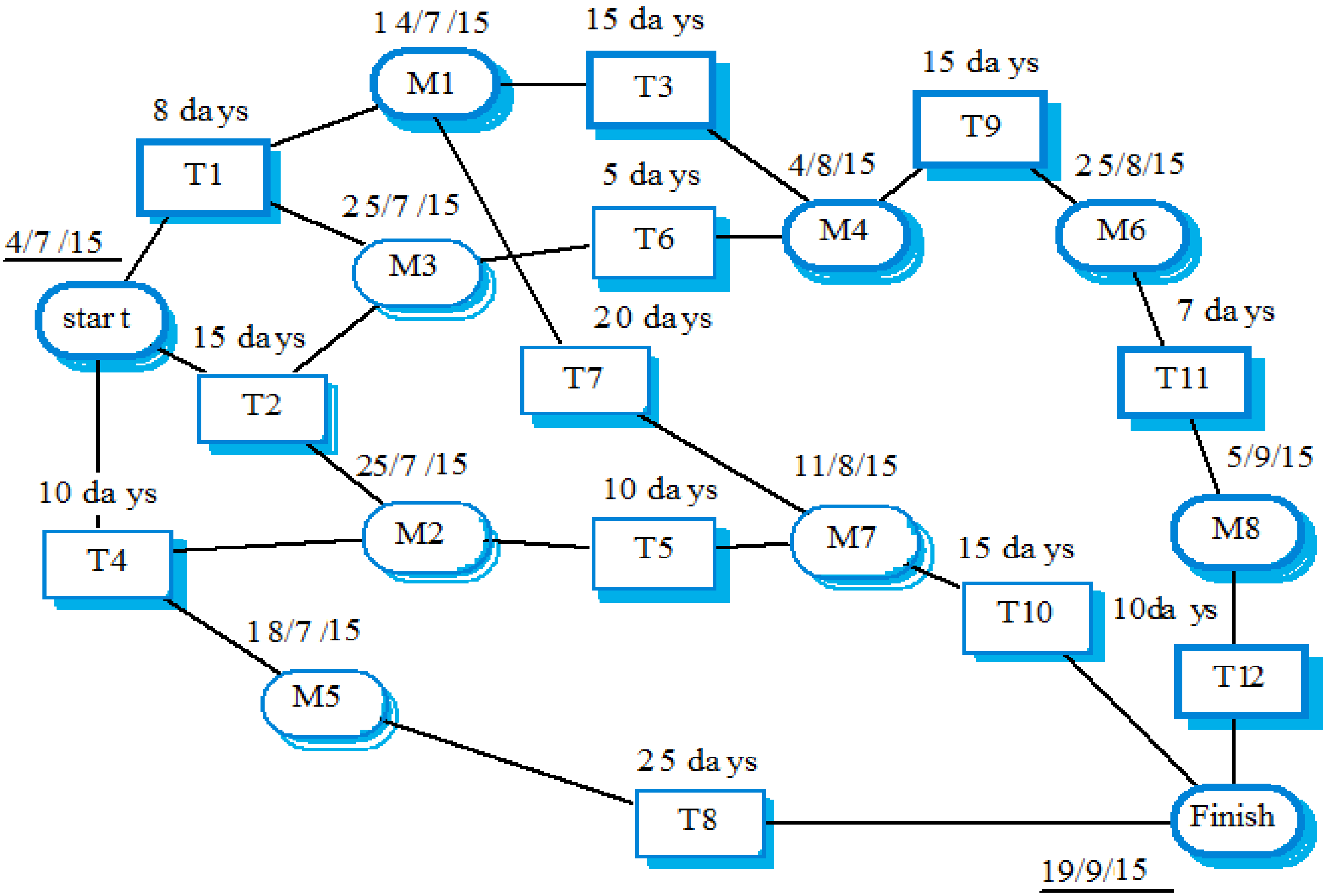
- Graphical notations used to illustrate the project schedule.
- Show project breakdown into tasks. Tasks should not be too small.

  They should take about a week or two.
- Activity charts show task dependencies and the critical path.
- Bar charts show schedule against calendar time.

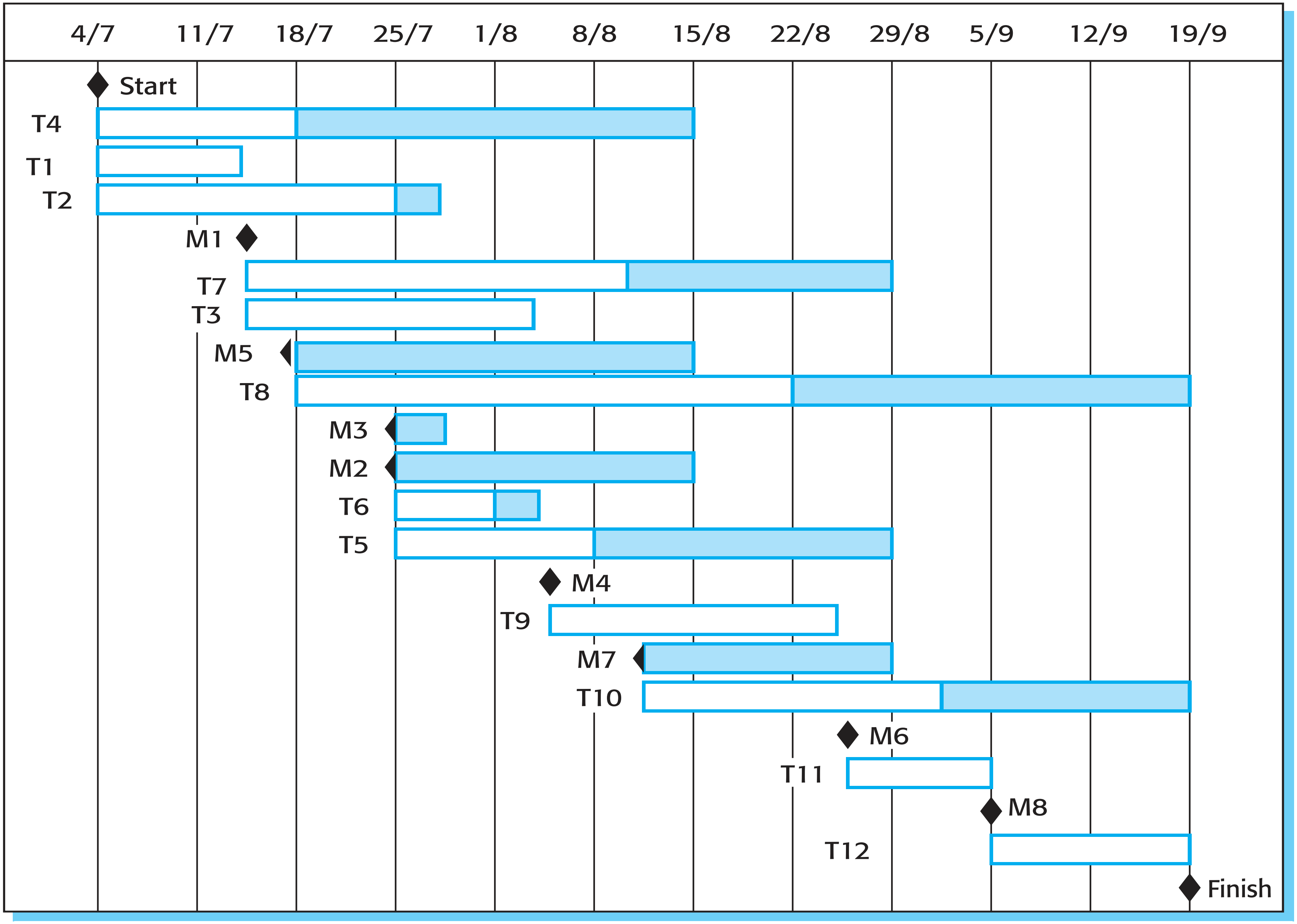
Task durations and dependencies

Activity	Duration (days)	Dependencies
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)
T6	5	T1, T2 (M3)
T7	20	T1 (M1)
T8	25	T4 (M5)
T9	15	T3, T6 (M4)
T10	15	T5, T7 (M7)
T11	7	T9 (M6)
T12	10	T11 (M8)

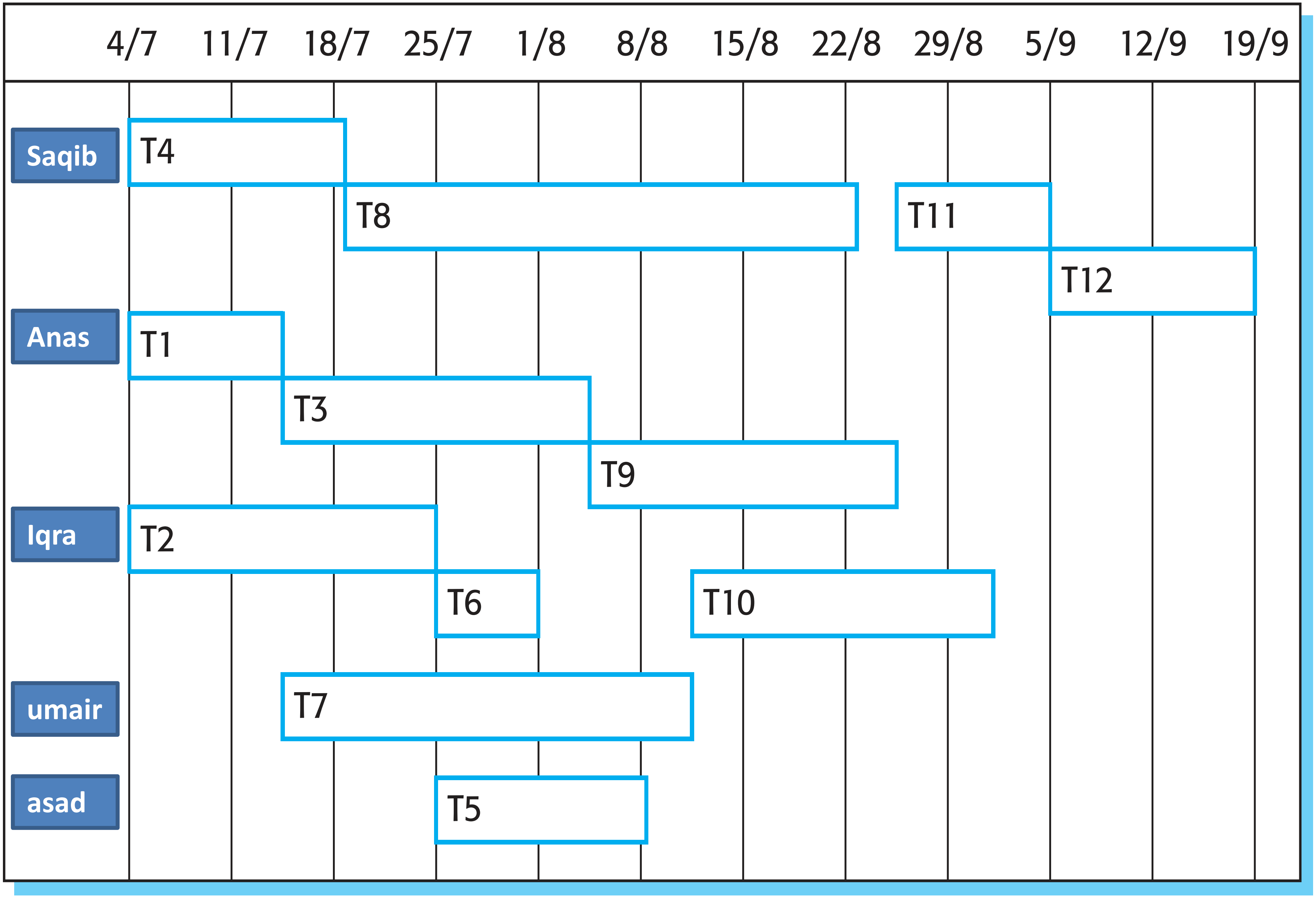
Activity network



Activity timeline



Staff allocation



Thank you for your time

Next session

- System requirement specification.
- Group activity (Group debate)