

# Object-Oriented Software Engineering

Using UML, Patterns, and Java

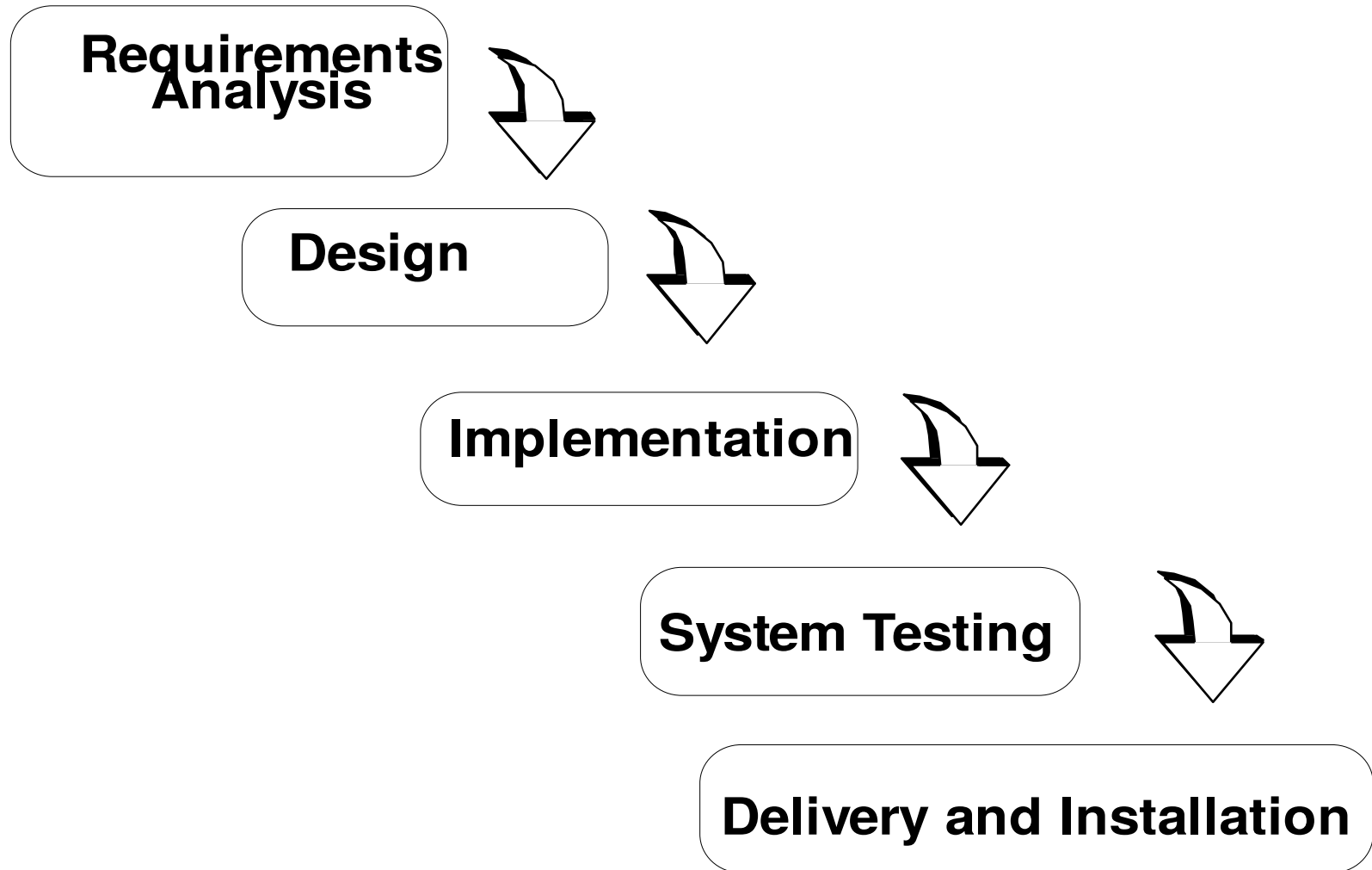
## 14.4 Project Organization



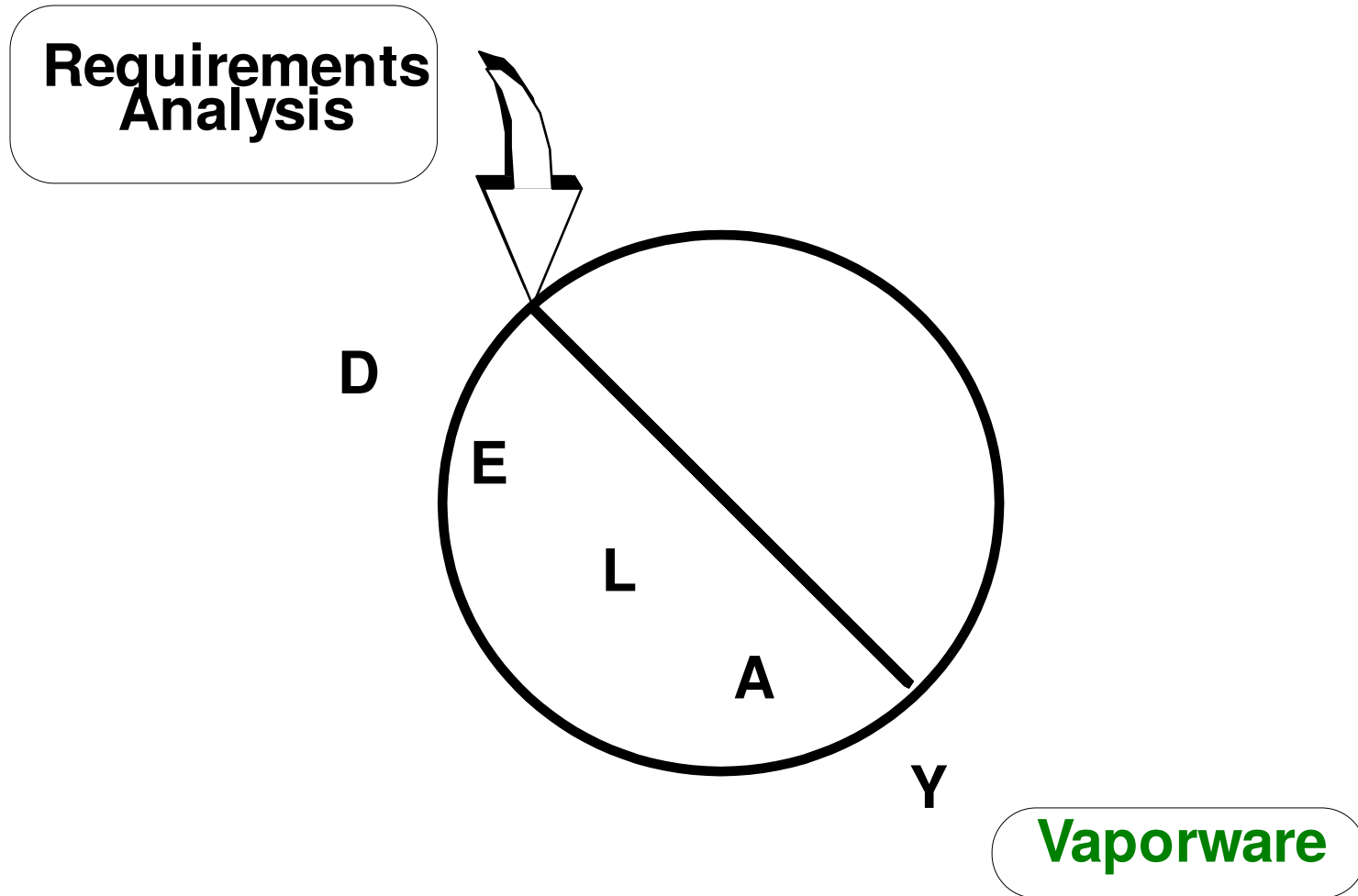
# *Laws of Project Management*

- ◆ Projects progress quickly until they are 90% complete. Then they remain at 90% complete forever.
- ◆ When things are going well, something will go wrong. When things just can't get worse, they will. When things appear to be going better, you have overlooked something.
- ◆ If project content is allowed to change freely, the rate of change will exceed the rate of progress.
- ◆ Project teams detest progress reporting because it manifests their lack of progress.

# *How it should go*



# *How it often goes*



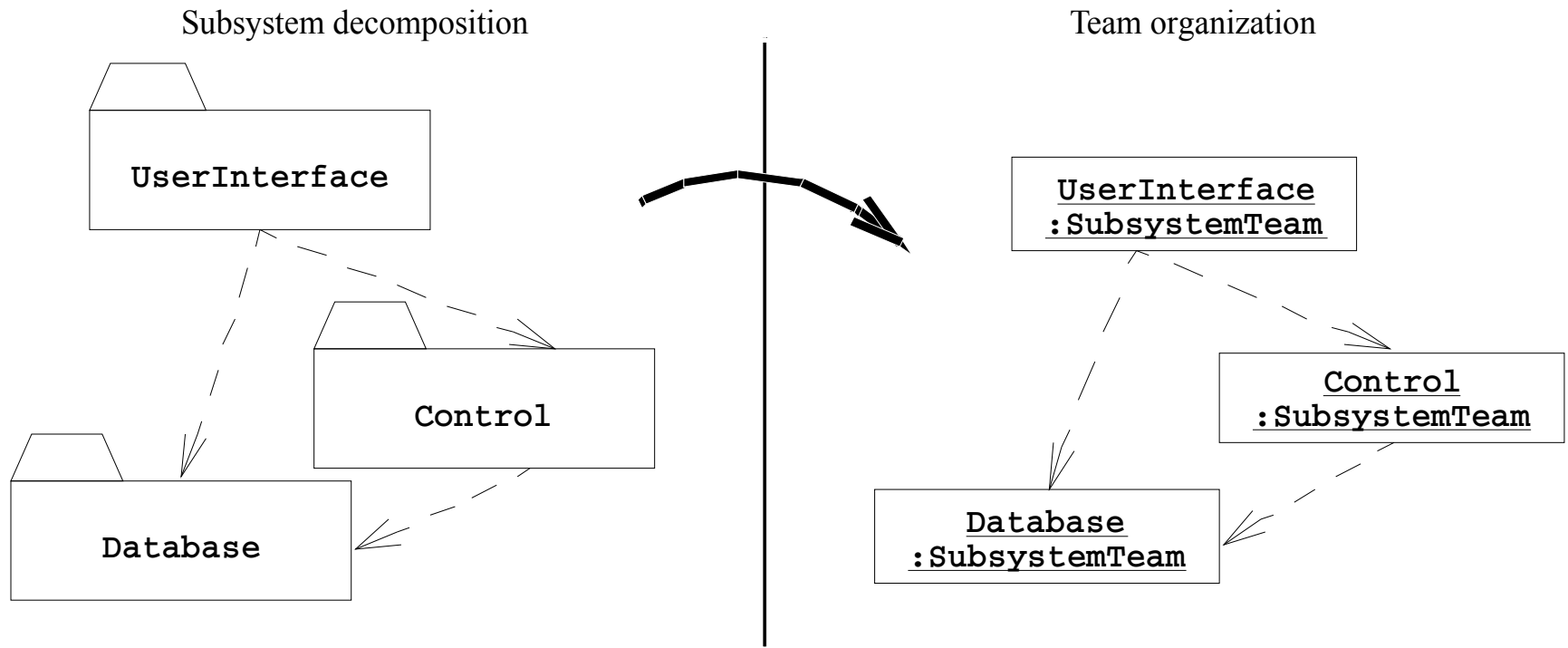
# *Outline*

- ◆ Organizational Structures
  - ◆ **Functional, Project and Matrix Organizations**
- ◆ Key project roles in organizational structures
  - ◆ **Project Manager, Team members, upper management, ...**
- ◆ Relationships between roles
- ◆ Information flows between roles
  - ◆ **Decision making, status reporting, communication**
- ◆ Identifying people
  - ◆ **Audience List, Drivers, Supporters, Observers**
  - ◆ **Involvement of audience members during the lifetime of a project**
- ◆ Properties of roles:
  - ◆ **Responsibilities, Authority and Delegation**
- ◆ And if time permits:
  - ◆ **Micromanagement (and how to avoid it)**

# *Organizational Structures*

- ◆ Types of Organization
  - ◆ **Functional organization**
  - ◆ **Project-based organization**
  - ◆ **Matrix organization**
- ◆ Parameters for each organization type
  - ◆ **Organizational Unit**
  - ◆ **Key players**
  - ◆ **Roles and Responsibilities**
  - ◆ **Structure: Information flow between roles**
  - ◆ **Benefits and Challenges (“pros and cons”)**
- ◆ Heuristics
- ◆ Let’s start with an example and a few definitions....

# Toy Project with 3 Teams



# *Groups, Teams and Committees*

- ◆ **Group:** A set of people who are assigned to a common task and who work individually to accomplish their assignment.
- ◆ **Team:** A small group of people working on the same problem or subproblem in a project. The team members depend on one another to do their tasks.
  - ◆ **Project Team:** Based on the premise that every member can and must make a valuable contribution to the project.
- ◆ **Committee:** Comprised of people who come together to review and critique issues, propose recommendations for action.



# *Organization*

- ◆ **Definition Organization:** A set of *organizational* units and their different *relationships* with each other.
- ◆ Organizational units can be organized according to many different categories, for example by function or by project type. Typical examples of organizational units:
  - ◆ **Functional organization:** Research, Development, Marketing, Sales
  - ◆ **Project organization:** Project 1, Project 2, ....
- ◆ An organization usually has 3 different types of relationships between organizational units.
  - ◆ **Reporting structure:** To report status information
  - ◆ **Decision structure:** To propagate decisions
  - ◆ **Communication structure:** To exchange information

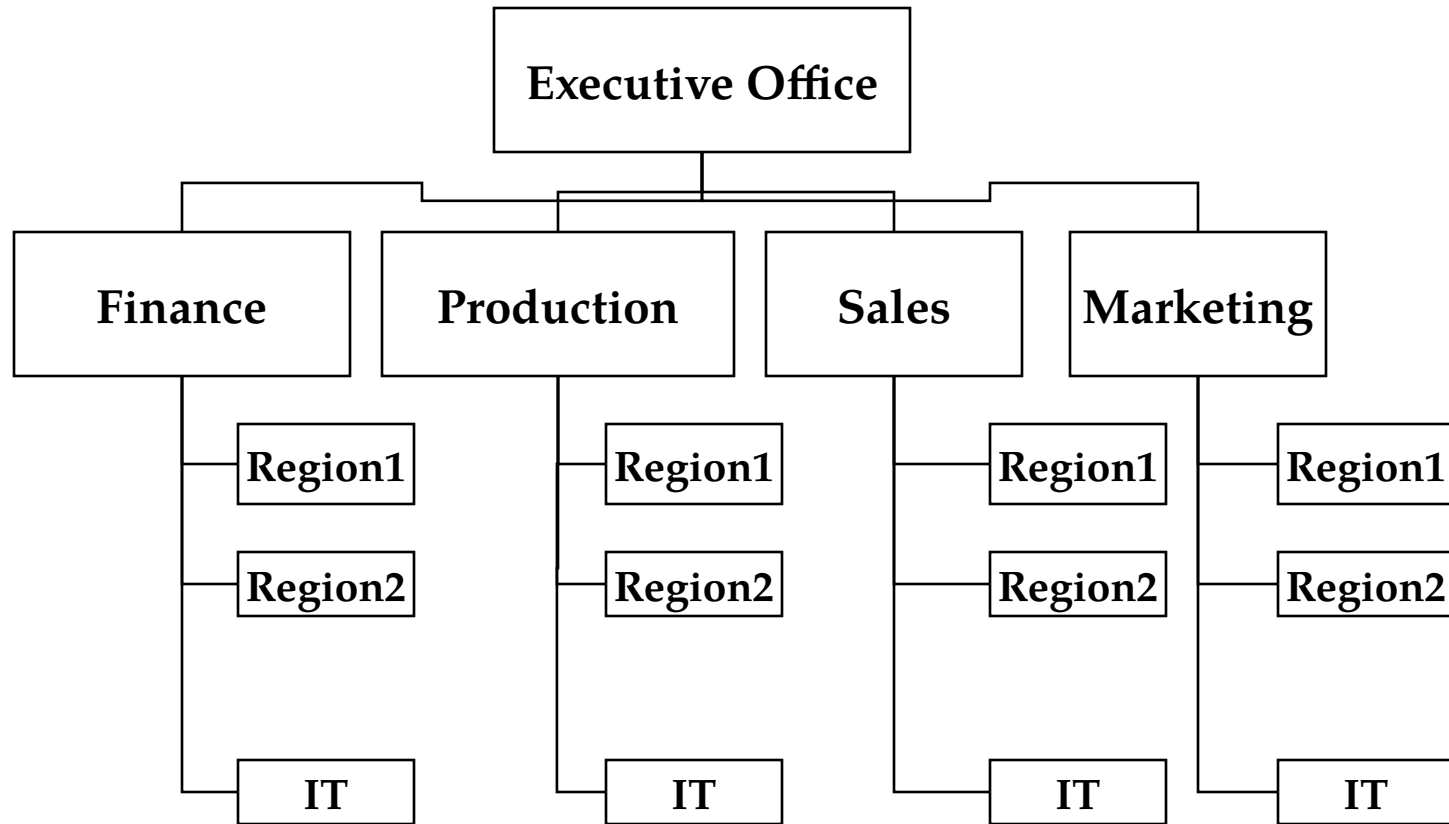
# *Roadmap for the lecture*

- ◆ We will first discuss different organization forms.
  - ◆ **Functional organization**
  - ◆ **Project organization**
  - ◆ **Matrix organization**
- ◆ Then we talk about the different roles played by people in these organizations
  - ◆ **Project manager, team member, upper management, ....**
- ◆ Then we discuss relationships between the roles
  - ◆ **Hierarchical organizations**
  - ◆ **Nonhierarchical organizations**

# *Functional Organization*

- ◆ **Definition:** In a **functional organization** participants are grouped into so-called **departments**, each of which addresses a function.
- ◆ Examples of departments:
  - ◆ **Traditional businesses:** Research, development, production, sales, finance.
  - ◆ **In software companies the departments correspond to the activities in the software process:** Analysis, design, integration, testing departments.
- ◆ Key properties:
  - ◆ **Projects are usually pipelined through the departments of a functional organization. The project starts in research, then it moves to development, then it moves to production, ....**
  - ◆ **Only a few participants are involved in the complete project.**
  - ◆ **Separate departments often address the same cross-functional needs (Examples: configuration management, IT infrastructure)**

# *Example of a Functional Organization*



Line organization of a „traditional business“

# *Properties of Functional Organizations*

- ◆ Advantages:
  - ◆ **Members of a department have a good understanding of the functional area they support.**
  - ◆ **Departments don't compete with another to get the support of their support teams**
- ◆ Disadvantages:
  - ◆ **Because each department has its own support team, different work procedures and reporting systems are the rule.**
  - ◆ **It is difficult to make major investments in equipment and facilities.**
    - ◆ **Example:** Two departments with a budget of 50,000 Euro each need a printer that costs 100,000 Euro.
    - ◆ Both need only 50% of the maximum capacity.
    - ◆ Neither department can buy it, because they don't have sufficient funds.
  - ◆ **High chance for overlap or duplication of work among departments**
  - ◆ **Conflicts between departments with different objectives**

# *Project Organization*

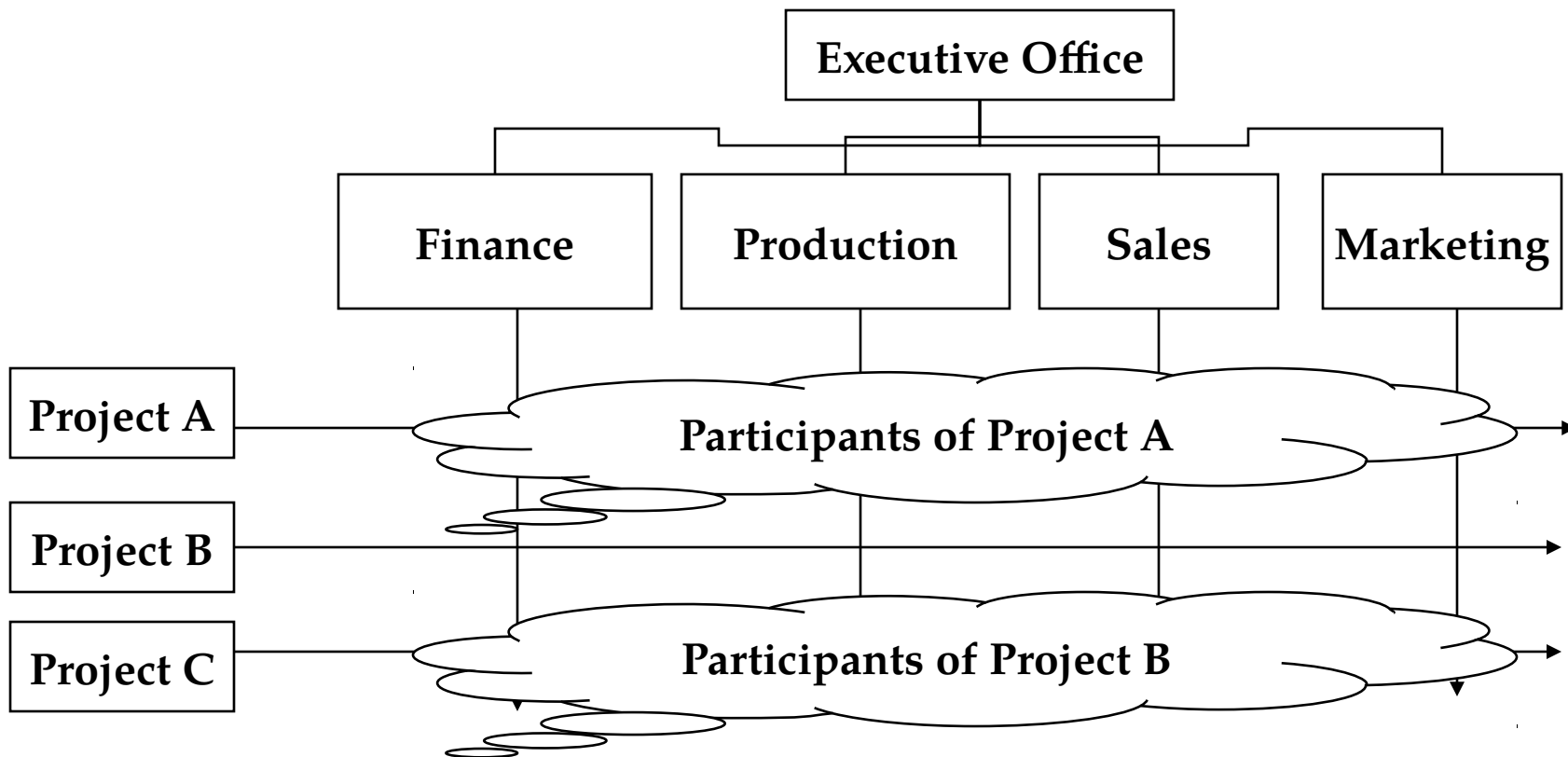
- ◆ In a **project organization** participants are grouped into **projects**, each of which has a problem to be solved within time and budget.
- ◆ **Key properties:**
  - ◆ **Teams are assembled for a project as it is created. Each project has a project leader.**
  - ◆ **All participants are involved in the complete project.**
  - ◆ **Teams are disassembled when the project terminates**

# *Properties of Project Organizations*

- ◆ Advantages
  - ◆ **Very responsive to new project requests (because the project is newly established and can be tailored around the problem)**
  - ◆ **New people can be hired/selected who are very familiar with the problem or who have special capabilities.**
  - ◆ **There is no waste of staff workload**
- ◆ Disadvantages:
  - ◆ **Teams cannot be assembled rapidly. Often it is difficult to manage the staffing/hiring process.**
  - ◆ **Because there are „no predefined lines“, roles and responsibilities need to be defined at the beginning of the project**
  - ◆ **What people will do after project completion?**

# Matrix Organization

- ◆ In a matrix organization, participants from different departments of the functional organization are assigned to work on projects as they are created.
- ◆ The project manager and team members may be assigned to the project for less than 100 % of their time





# *Properties of Matrix Organizations*

- ◆ Advantages:
  - ◆ Teams for projects can be assembled rapidly
  - ◆ Scarce expertise can be applied to different projects as needed
  - ◆ Consistent work and reporting procedures can be used for projects of the same type.
- ◆ Disadvantages:
  - ◆ Team members usually are not familiar with each
  - ◆ Team member have different working styles
  - ◆ Team members must get used to each other
  - ◆ Conflicts between functional and project manager

# *New Challenges in Matrix Organizations*

- ◆ Team members must respond to two different bosses with different focus:
  - ◆ **Focus of the functional manager:** Assignments to different projects, performance appraisal
  - ◆ **Focus of the project manager:** Work assignments, project team support
- ◆ Team members working on multiple projects have competing demands for their time
  - ◆ **Team members working on more than one project have even more project members to report to**
  - ◆ **Some people who have claim on the team member's time may be at similar levels in the organization's hierarchy**
- ◆ Multiple work procedures and reporting systems are used by different team members
  - ◆ **Development of common procedures needs to be addressed at project kickoff time**

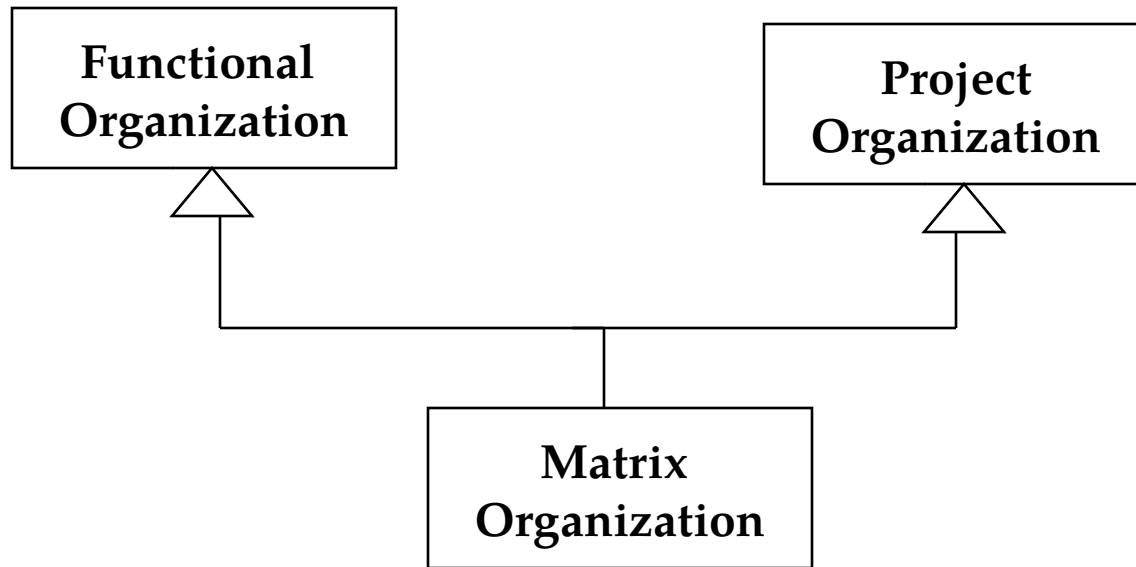
# *When to use a Functional Organization*

- ◆ Projects with high degree of certainty, stability, uniformity and repetition.
  - ◆ **Requires little communication**
  - ◆ **Role definitions are clear**
- ◆ **When?**
  - ◆ **The more people on the project, the more need for a formal structure**
  - ◆ **Customer might insist that the test team be independent from the design team**
  - ◆ **Project manager insists on a previously successful structure**

# *When to Use a Project or Matrix Organization*

- ◆ Project with degree of uncertainty
  - ◆ Open communication needed among members
  - ◆ Roles are defined on project basis
- ◆ When?
  - ◆ Requirements change during development
  - ◆ New technology develops during project

# *Metamodel for Organizations*



# *Roadmap for the Lecture*

- ✓ We discussed different organization forms.
  - ◆ **Functional organization**
  - ◆ **Project organization**
  - ◆ **Matrix organization**
- Now we will talk about the different roles played by people in these organizations
  - ◆ **Project manager, team member, upper management, ....**
- ◆ Then we discuss different types of relationships between the roles
  - ◆ **Hierarchical organizations**
  - ◆ **Nonhierarchical organizations**

## *Definition: Role*

- ◆ A **role** is a set of responsibilities
- ◆ A role is instantiated during a project and assigned to one or more persons.
- ◆ Instances of roles are often also called **players** or **stakeholders**

# *Key Roles in Organizations*

- ◆ **Project Manager:** The person ultimately responsible for the successful completion of the project
- ◆ **Project Team Member:** Participants who are responsible for performing individual activities and tasks (in a project or matrix organization)
- ◆ **Functional Manager:** The team member's supervisor in the department (in a functional organization)
- ◆ **Upper management:** People in charge of the departments or projects

In the following we focus only on roles in project and matrix organizations



# *Responsibilities of the Project Manager*

- ◆ Determine objectives, schedule and resource budgets
- ◆ Design a software project management plan (SPMP)
- ◆ Create and sustain focused and motivated teams
- ◆ Determine the team's work procedures, reporting systems and communication infrastructure.
- ◆ Accomplish project objective within time and budget
- ◆ Monitor performance against the plan
- ◆ Resolve technical conflicts and interpersonal conflicts
- ◆ Control changes in the project
- ◆ Report on project activities to upper management
- ◆ Keep the client informed and committed
- ◆ Contribute to the team members performance approval

# *General Responsibilities of Team Members*

- ◆ Technical responsibilities:
  - ◆ Perform assigned tasks within time and budget
  - ◆ Acquire technical skills and knowledge needed to perform the work
- ◆ People responsibilities
  - ◆ Identify situations and problems that might affect your team members's tasks
  - ◆ Keep your team members informed of your progress and problems you encounter

# *Other Team Member Roles*

- ◆ **Project Management**
  - ◆ **Coach**
  - ◆ **Team leader**
  - ◆ **API Liaison**
  - ◆ **Planner**
- ◆ **Meeting Management**
  - ◆ **Minute Taker**
  - ◆ **Scribe**
  - ◆ **Primary facilitator**
- ◆ **Development**
  - ◆ **Analyst**
  - ◆ **Designer (Software Architect)**
  - ◆ **Programmer**
  - ◆ **Tester**
  - ◆ **Maintainer**
  - ◆ **Trainer**
  - ◆ **Document Editor**
  - ◆ **Web Master**
  - ◆ **Configuration Manager**

# *Responsibilities of the Coach*

- ◆ Listen to gripes from individual team members
- ◆ Attend weekly project meetings
- ◆ Review weekly team status reports
- ◆ Schedule and prepare meetings with project manager
- ◆ Insist that project guidelines are followed
- ◆ Assign presentations to team members (in-class project meetings, client review, client acceptance test)
- ◆ Resolve team member conflicts if they cannot be resolved otherwise

# *Responsibilities of the Team Leader*

- ♦ Responsible for intra-team communication
  - ♦ **Run the weekly project meeting**
  - ♦ **Post the agenda before the meeting**
  - ♦ **Define and keep track of action items assigned to team members (who, what, when)**
  - ♦ **Measure progress (Enforce milestones)**
  - ♦ **Deliver work packages for the tasks to the project manager**
  - ♦ **Present team status to project manager**
- ♦ *Heuristics: The team leader should to be rotated among members of the team.*

# Team Leader: Create an Agenda

- ◆ Purpose of Meeting
- ◆ Desired Outcome
- ◆ Information Sharing
- ◆ Information Processing
- ◆ Meeting Critique

Action Items  
(Check Previous Meeting)

Issues  
(Check Previous Meeting & BBoards)

New Agenda

Agenda for Database Group

Date: 06/19/96

Location: Primary Facilitator: Bernd Bruegge

Start Time: 12:00 PM Minute Taker:

End Time: 12:00 PM Time Keeper:

Purpose of the Meeting

2. Desired Outcome

3. Information Sharing (15 Minutes)

Include Action Item Text: Yes Update Action Item Text

To exclude Action Items, choose 'No' and press 'Update Action Item Text'.  
To include Action Items, choose 'Yes' and press 'Update Action Item Text'.  
These two fields form a toggle switch for including Action Items in this Agenda.  
The 'Update Action Item Text' button can also be used to refresh the Action Items in an Agenda.

4. Information Processing (40 Minutes)

Include Issue Text: Yes Update Issue Text

To exclude Issues, choose 'No' and press 'Update Issue Text'.  
To include Issues, choose 'Yes' and press 'Update Issue Text'.  
These two fields form a toggle switch for including Issues in this Agenda.  
The 'Update Issue Text' button can also be used to refresh the set of Issues in an Agenda.

## *Responsibilities of the API Liaison*

- ◆ Responsible for inter-team communication
  - ◆ **API Liaison: Make available public definitions of subsystem developed by the team to the architecture teams (ensure consistency, etc)**
  - ◆ **Coordinate tasks spanning more than one group with other teams**
- ◆ Responsible for service negotiations

## *Responsibilities of the Planner*

- ◆ Plans the activities of an individual team
- ◆ Define project plan for team:
  - ◆ **Work Breakdown Structure**
  - ◆ **Dependency graph and schedule showing work packages**
- ◆ Make project plan available to management
- ◆ Report team project status to team leader

No explicit planner in many teams. Responsibility usually assumed by team leaders or project manager



## *Responsibilities of the Document Editor*

- ◆ Collect, proofread and distribute team documentation
- ◆ Submit team documentation to documentation team
- ◆ Collect agendas
- ◆ Take minutes at meetings

# *Responsibilities of the Web Master*

- ◆ Maintain team home page
- ◆ Keep track of meeting history
- ◆ Keep track of design rationale

# *Web Master*

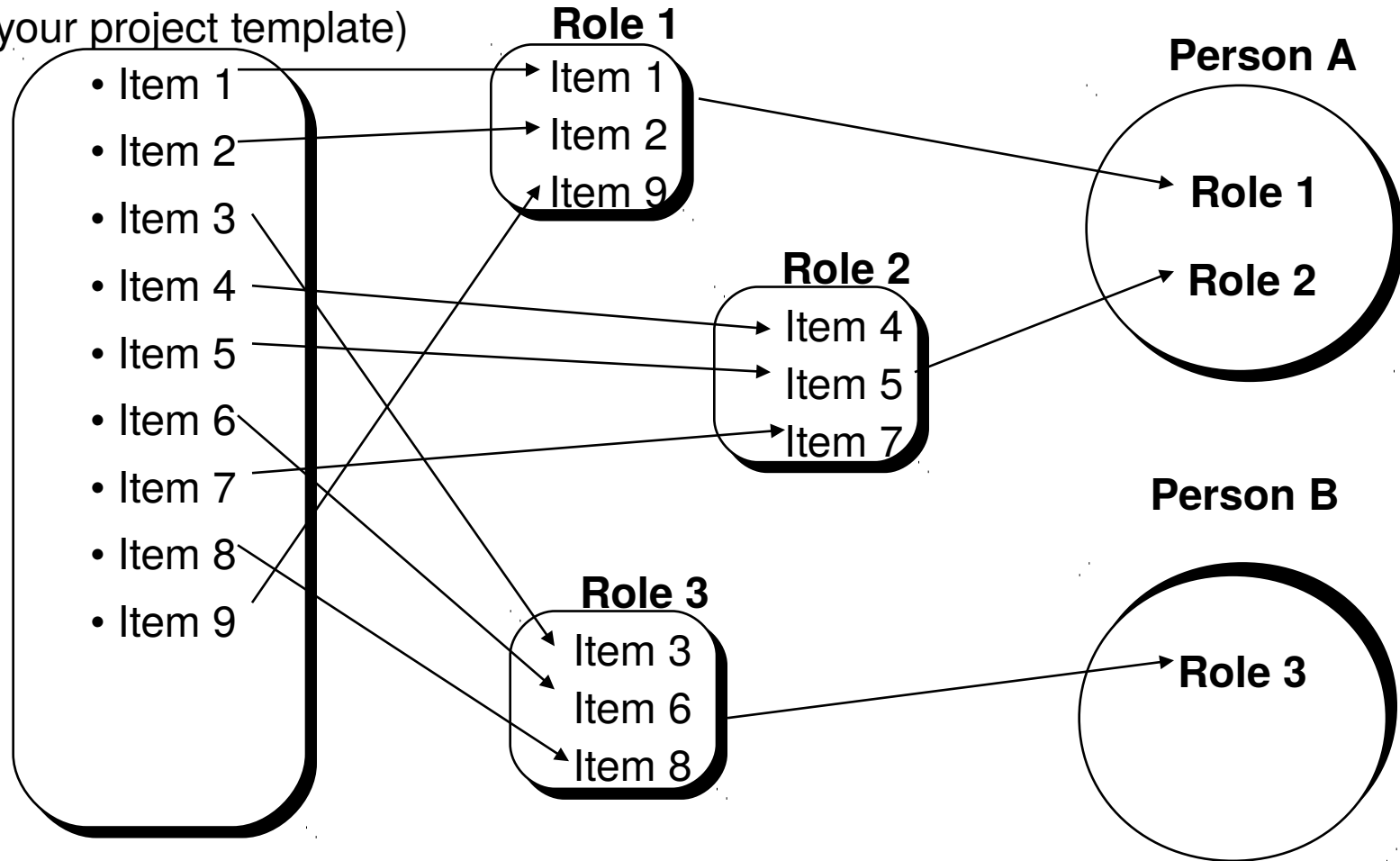
- ◆ Publish Meeting Information on Team Homepage
  - ◆ Should contain agenda, minutes, action items and issues
  - ◆ Possibilities:
    - ◆ One HTML document per meeting, with anchors (maintained by one role)
    - ◆ Separate HTML documents for Agenda, Minutes, etc (maintained by several roles)

Date	Agenda	Minutes	Action Items	Issues
9/9/96	<u>Agenda</u>	<u>Minutes</u>	<u>Action Items</u>	<u>Issues</u>
9/16/96	<u>Agenda</u>	<u>Minutes</u>	<u>Action Items</u>	<u>Issues</u>

# Assigning Responsibilities To People

Project To Do List

(from your project template)



Bindings made During  
Project-Initiation Phase

Bindings made during  
Hiring, Initial Planning phase  
(First team meeting, etc . . .)

# *Mapping Responsibilities to People*

- ◆ **One-to-One**
  - ◆ **Ideal but often not worth to be called a project**
- ◆ **Many-to-Few**
  - ◆ **Each project member assumes several roles ("hats")**
  - ◆ **Danger of over-commitment**
  - ◆ **Need for load balancing**
- ◆ **Many-to-"Too-Many"**
  - ◆ **Some people don't have significant roles**
  - ◆ **Bystanders**
  - ◆ **People loose the touch with project**

# *Towards A Project Role Taxonomy*

- ◆ Management roles
  - ◆ **Organization and execution of the project within constraints. Examples: project manager, team leader.**
- ◆ Development roles
  - ◆ **Specification, design and construction of subsystems. Examples: Analyst, software architect, programmer.**
- ◆ Cross functional roles
  - ◆ **Execute project functions. Example: API Liaison, configuration manager**
- ◆ Consultant roles
  - ◆ **Supports in areas where the project participants lack expertise. Examples: End user, client, application domain specialist ( problem domain), technical consultant (solution domain).**
- ◆ Promoter roles
  - ◆ **Deals with change in the organization, application/solution domain or process.**

# *Promoter Roles*

- ◆ Promoter are self appointed individuals who identify themselves with the outcome of the project.
  - ◆ **They are member of the corporate organization and may not necessarily be directly involved with the project.**
  - ◆ **Instead, they are the interface to the rest of the corporate organization.**
- ◆ Because of their power, knowledge of technology, or familiarity with the project's processes, they are able to promote and push specific changes through an existing organization which are needed to make the project a success.

# *Power Promoter*

- ◆ Also called executive champion or project champion
- ◆ Pushes the change through the existing organizational hierarchy.
  - ◆ not necessarily at the top of the organization, but must have protection from top level management, otherwise project opponents might be able to prevent the success of the project.
- ◆ **Tasks:**
  - ◆ Constantly identify difficulties, resolve issues, and communicate with the project members, especially with the developers.
- ◆ **Example at project level:** Project Leader.
- ◆ **Example at corporate level:** Chief Executive Officer (CEO).



# *Knowledge Promoter*

- ◆ Also called the technologist
- ◆ Promotes change arising in the application domain or the solution domain. Usually closely associated with the power promoter.
- ◆ **Tasks:** Acquire information iteratively, understand the benefits and limitations of new technologies, and argue its adoption with the other developers.
- ◆ **Example at project level:** System architect.
  - ◆ Reports to project manager
  - ◆ Does not have any direct subordinate in the reporting hierarchy
  - ◆ Has final say over all technical decisions in the system.
- ◆ **Example at corporate level:** Chief Technical Officer (CTO).

# *Process Promoter*

- ◆ The process promoter has intimate knowledge of the projects processes and procedures.
- ◆ The process promoter is in constant interaction with the power promoter to get consensus on the overall goals.
- ◆ **Tasks:** Bridge between the power and knowledge promoters, who often do not speak or understand the same language.
- ◆ **Example at project level:** Development lead. Responsible for the administrative aspects of a project, including planning, milestones definition, budgeting and communication infrastructure.
- ◆ **Example at corporate level:** Chief Information Officer (CIO)

# *Roadmap for the Lecture*

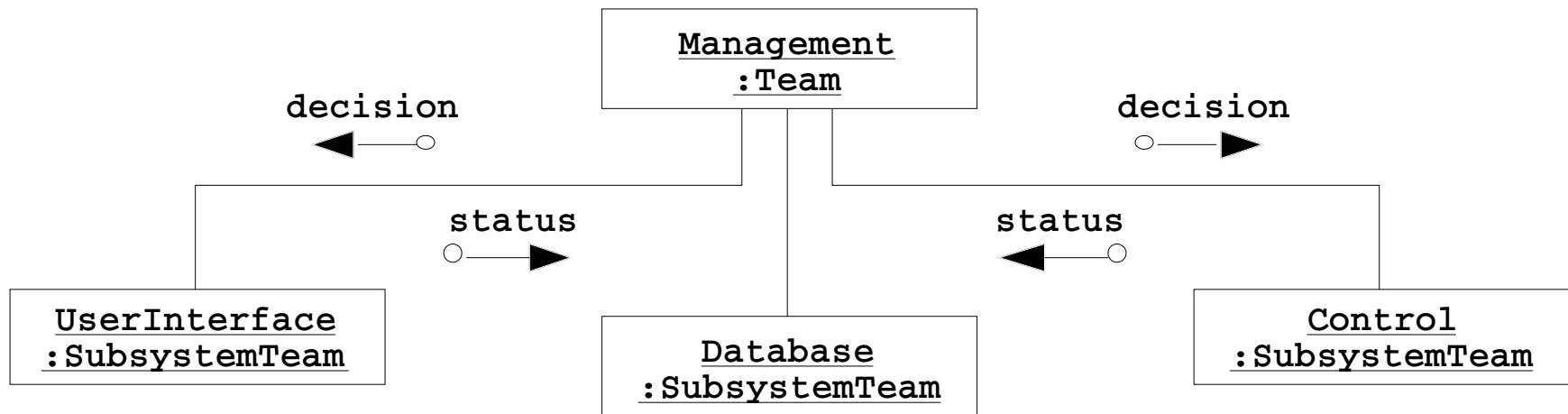
- ✓ We first discussed different organization forms.
  - ◆ **Functional Organization**
  - ◆ **Project Organization**
  - ◆ **Matrix Organization**
- ✓ Then we talked about the different roles played by people in these organizations
  - ◆ **Project Manager, Team Member, Upper Management, ....Promoters**
- ➔ Now we discuss different types of relationships between the roles
  - ◆ **Hierarchical Organizations**
  - ◆ **Nonhierarchical Organizations**

# *Relationships between Roles*

- ◆ Organizations can have many different types of associations between roles
- ◆ The three most important associations for project organizations are: *Reporting*, *decision making* and *communicating*
- ◆ Reporting association:
  - ◆ **Used for reporting status information**
- ◆ Decision association
  - ◆ **Used for propagating decisions**
- ◆ Communication association
  - ◆ **Used for exchanging information needed for decisions (e.g., requirements, design models, issues).**

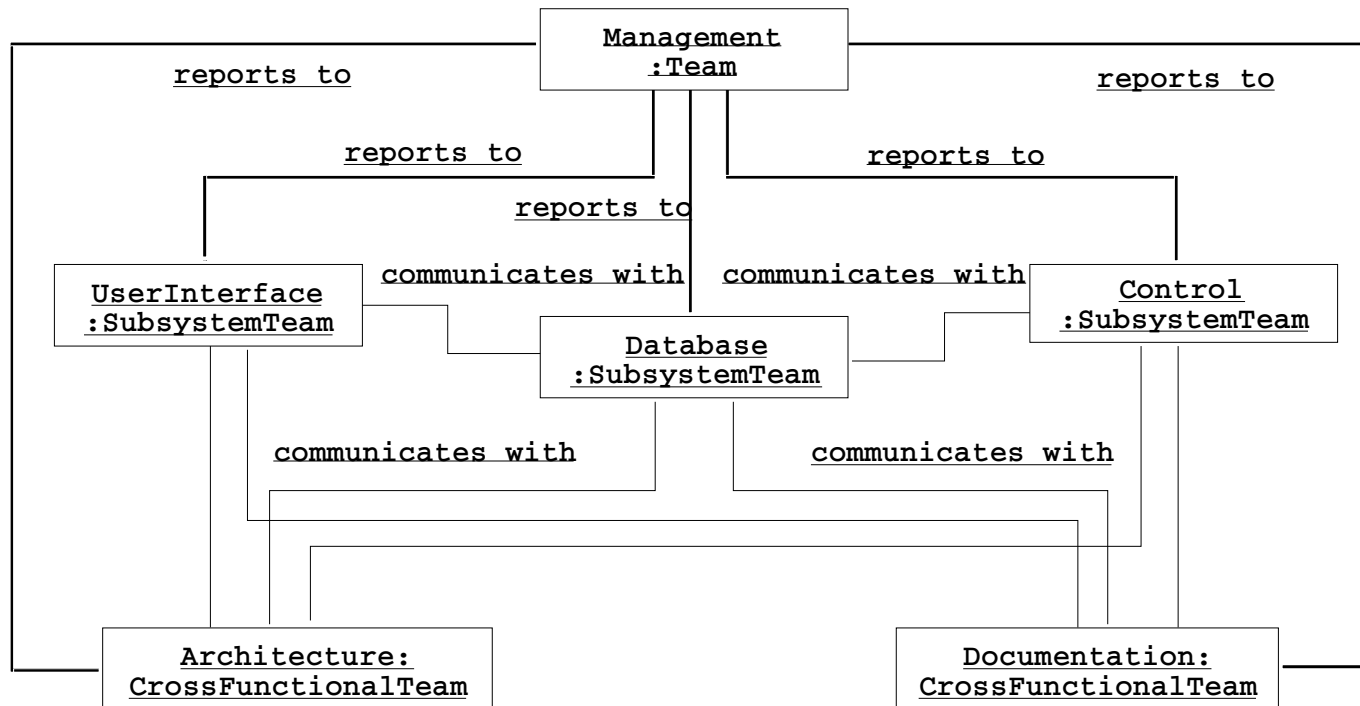
# *An Organization with a Reporting and Decision Structure*

- ♦ The **developers** make local decisions and reports them via a status report to the leader (team leader, project manager)
- ♦ The **team leader**, who has a local overview of the subsystem, can override these decisions. She reports them to the project manager.
- ♦ The **project manager**, who has a global view of the project, can virtually override any decision.



# *An Organization with Distinct Reporting, Decision and Communication Structures*

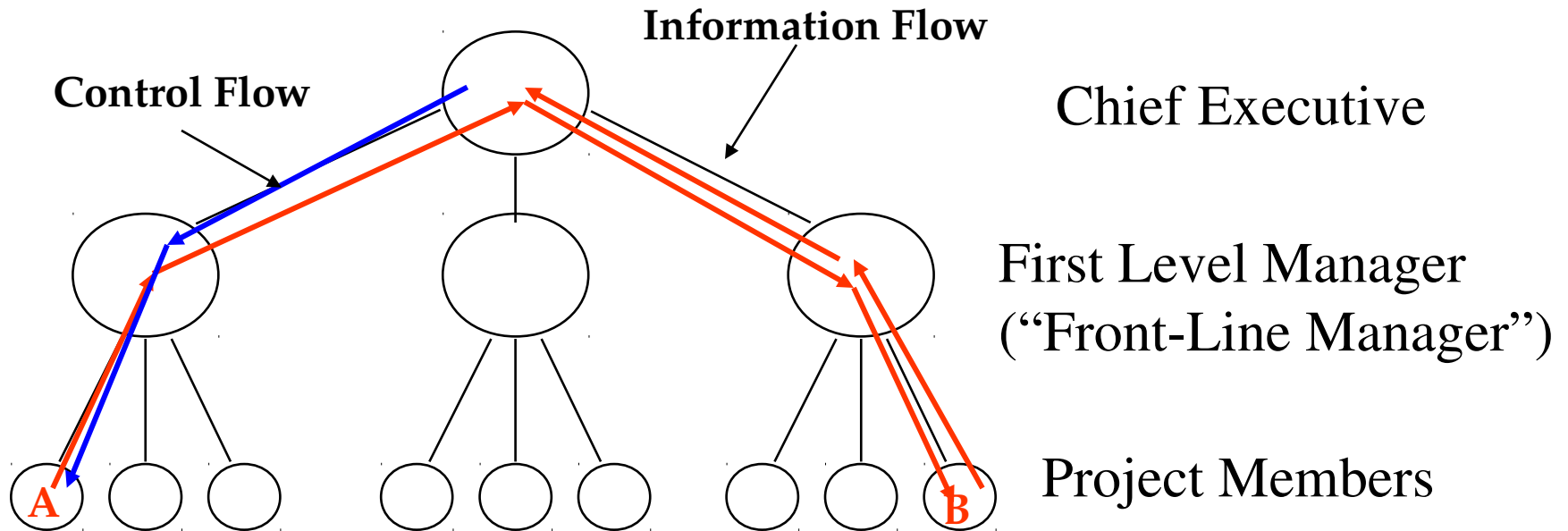
- ◆ **Developers** communicate with each other without having to communicate with the team leader or project leader.
- ◆ **Developers** make local decisions and report them to the leader
  - ◆ **The team leader, who has a local overview of the subsystem, can override these decisions. She reports them to the project manager.**
  - ◆ **The project manager, who has a global view of the project, can virtually override any decision.**



# *Hierarchical Organization*

- ◆ Often also called *centralized organization*. Examples: Military, church, traditional businesses.
- ◆ **Key property:** The organization has a tree structure. Decisions are made at the root and communicated to the leaf nodes. The decision association is also used for reporting and communication.
- ◆ Advantages:
  - ◆ **Centralized control over project selection**
  - ◆ **One set of management and reporting procedures for all project participants across all projects**
  - ◆ **Established working relationships among people**
  - ◆ **Clearly established lines of authority to set priorities and resolved conflicts**
  - ◆ **Authority to pressure people to honor their action items**
  - ◆ **Clearly defined career path**

# Hierarchical Project Organization

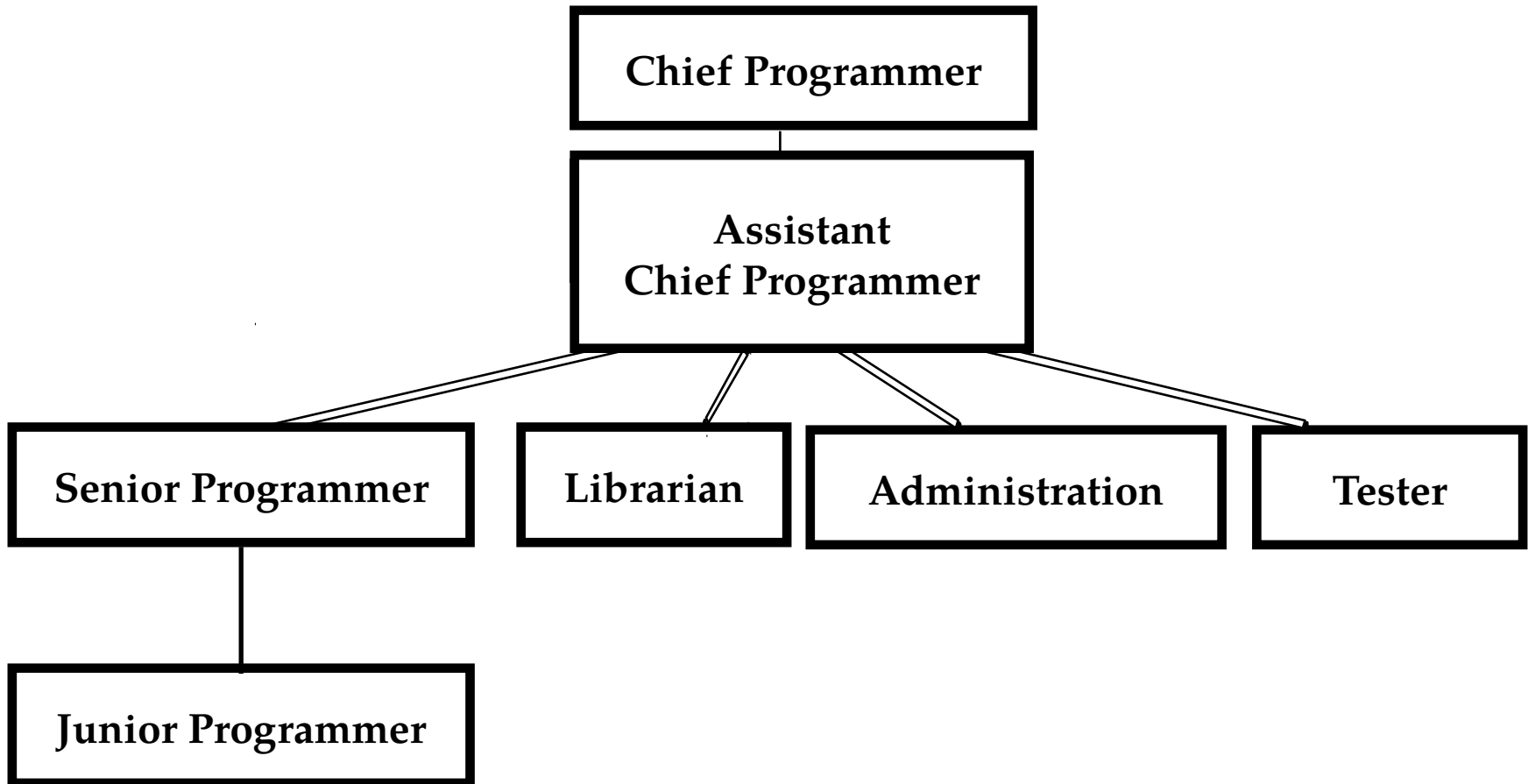


**A wants to talk to B: Complicated Information Flow** interdipendenza  
**B wants to make sure A does a certain change: Complicated Controlflow**

Basis of organization:  
Complicated information and control flow  
across hierarchical boundaries



# *Example of a Hierarchical Organization: Chief Programmer Team [Brooks 1995]*



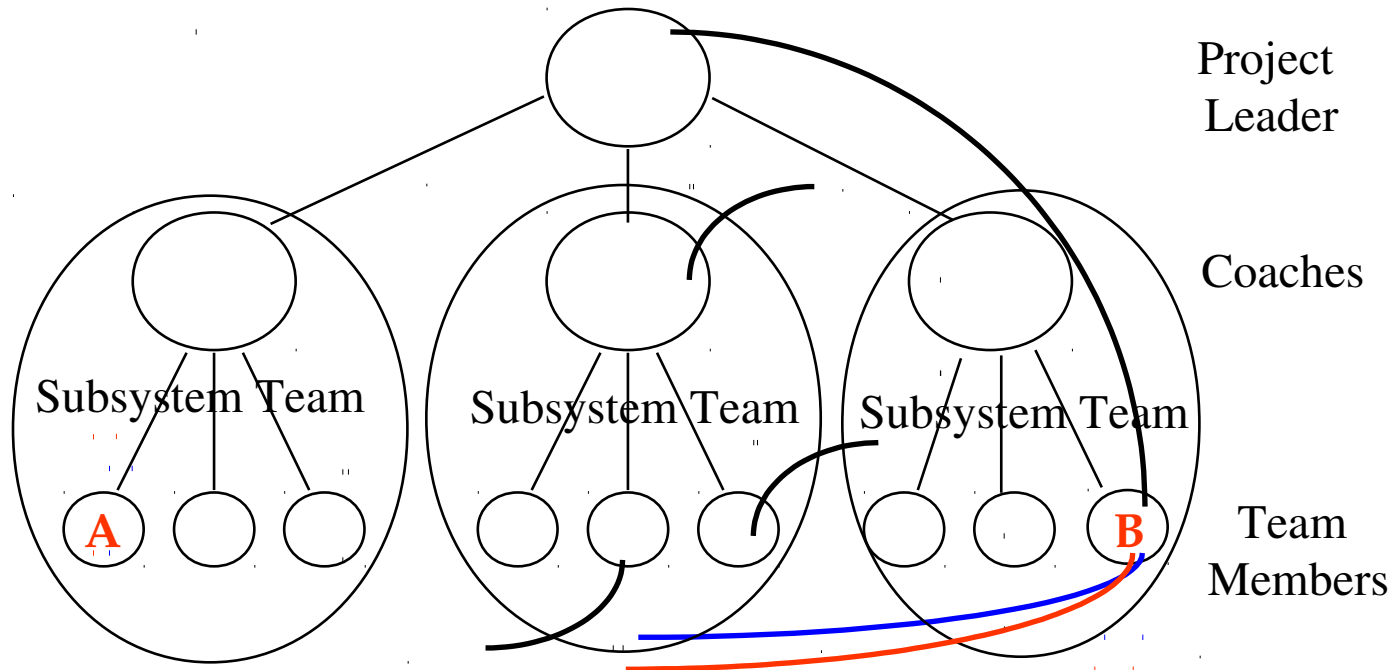
# *Disadvantages of Hierarchical Organizations*

- ◆ Slow response time
  - ◆ **The process of evaluating and approving change requests often takes too long because of long reporting/decision lines.**
- ◆ Difficult to manage the workload of the people:
  - ◆ **People are assigned fulltime to the organization, but projects don't come in a smooth stream.**
  - ◆ **Project request might not require the people who are available or their expertise.**
- ◆ Unfamiliarity with application or solution domain area
  - ◆ **People are usually hired for their technical proficiency in a specialty that the organization normally performs.**
  - ◆ **They often have only limited experience, if the problem to be solved is outside of their field of expertise.**

# *Nonhierarchical Organizations*

- ♦ **Key property:** The organization has a general graph structure with different edges for the decision, reporting and communication flows. Decisions can be made at various nodes in the graph.

# Nonhierarchical Project Organization

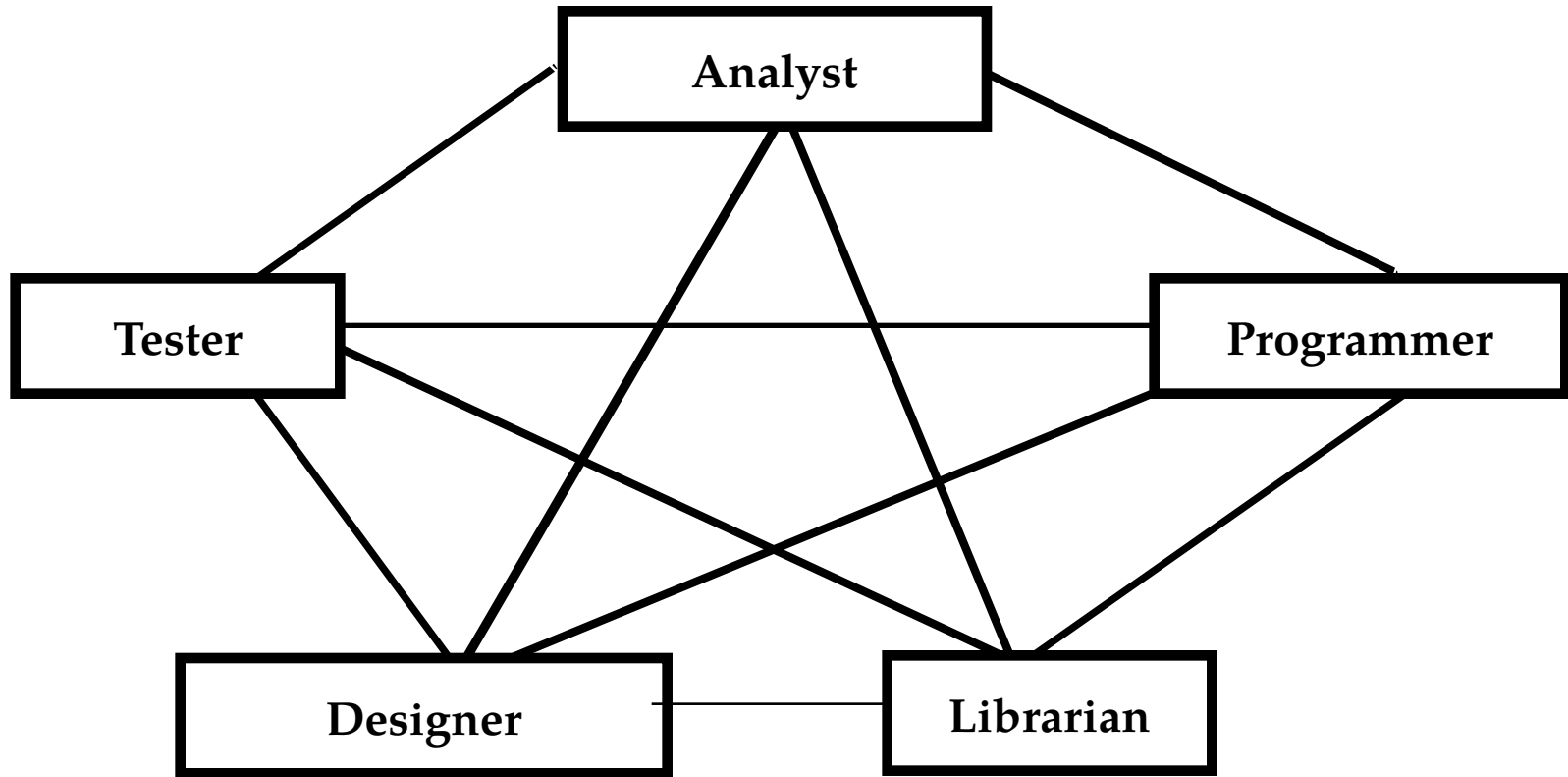


**A wants to talk to B: Communication Flow**

**B wants to make sure A does a certain change: Decision Flow**

**Basis of organization:  
Nonlinear information flow across dynamically formed units**

# *A Nonhierarchical Organization: Egoless Programming [Weinberg 1971]*



# *Observations on Organizational Structures*

- ◆ Hierarchical structure
  - ◆ “Reports”, “Decides” and “Communicates-With” all mapped on the same association
  - ◆ Does not work well with iterative and incremental software development process
  - ◆ Manager is not necessarily always right
  - ◆ Innovative proposals can be lost at any level
- ◆ Project-based structures
  - ◆ “Reports”, “Decides” and “Communicates-With” are different associations
  - ◆ Cuts down on bureaucracy
  - ◆ Reduces development time
  - ◆ Decisions are expected to be made at each level
  - ◆ Hard to manage

# *Flexibility of Organizations*

- ◆ An organization is flexible, if it allows “late” or “dynamic” bindings between roles and people and information flows between roles.
- ◆ Late binding (Cannot be changed after project kickoff):
  - ◆ **Organizational units and information flows are established for the project. (Example: The top level design influences the team structure: At kickoff each subsystem is assigned to a team)**
- ◆ Dynamic binding (Can be changed anytime):
  - ◆ **The organizational relationship changes over time (Example: We start with a hierarchical organization at project kickoff and end with a nonhierarchical organization at project finish time.)**
  - ◆ **We recognize the fact that organizational units change over time**
    - ◆ **New teams can be formed**
    - ◆ **Existing teams can be merged**
    - ◆ **An existing team can be removed from the organization**

# *Heuristics for Project Managers*

## 1. Create team identity

- ♦ **Clarify team vision and working relationships**
- ♦ **Define team procedures (meeting management, configuration management, system integration strategy)**
- ♦ **Clarify each participant's authority**
- ♦ **Make sure your team is functioning**
- ♦ **Be sure only one person is assigned as project manager**

## 2. Create team membership buy-in

- ♦ **Get commitment to the project goals (tough in matrix environment)**
- ♦ **Get to know other people's style**

## 3. Get support from the environment

- ♦ **Get a project champion (for example a power promoter)**

## 4. Develop general procedures for

- ♦ **Conflict resolution**
- ♦ **Communication between teams and project leaders, communication with upper management and for communication with the client**



# *Outline of this class*

- ✓ Organizational Structures
  - ◆ **Functional, Project and Matrix Organizations**
- ✓ Key project roles in organizational structures
  - ◆ **Project Manager, Team members, upper management, ...**
- ✓ Relationships between roles
- ✓ Information flows between roles
  - ◆ **Decision making, status reporting, communication**
- ▣ Identifying people
  - ◆ **Audience List, Drivers, Supporters, Observers**
  - ◆ **Involvement of audience members during the lifetime of a project**
- ❖ Properties of roles
  - ◆ **Responsibilities, Authority and Delegation**
- ◆ **Micromanagement (and how to avoid it)**

# *Identifying People*

- ◆ As soon as you start thinking about a project, you should start an audience list.
- ◆ Project Lists:
  - ◆ **Project Audience Lists**
  - ◆ **Stakeholder lists**
  - ◆ **Distribution lists**
  - ◆ **Team Member lists**
- ◆ **Audience List:** A list of people or groups of people that support, is affected by or is interested in the project.

## *Other Project Lists*

- ◆ **Stakeholder list** : Identifies people and groups who support or are affected by your project. This list does not include people outside of the organization or those who are merely interested in the project.
- ◆ **Distribution Lists**: Identifies people who receive copies of written project communication. The presence of people on distribution lists does not ensure that they actually support the project (Often out of date)
- ◆ **Team member lists**: People whose work is directed by the project manager.

# *Categories for an Audience List Template*

- ◆ **Internal**
  - ◆ **Upper Management**
  - ◆ **Project Manager**
  - ◆ **Team members**
  - ◆ **People with special knowledge**
- ◆ **External:**
  - ◆ **Clients or customers**
  - ◆ **Collaborators**
  - ◆ **Vendors, suppliers and contractors**
  - ◆ **Regulators**
  - ◆ **The Public**
- ◆ **Support Groups**
  - ◆ **Human Resources**
  - ◆ **Legal services**
  - ◆ **Contracting**
  - ◆ **Finances**
  - ◆ **Security**
  - ◆ **Computing Facilities**
- ◆ **End users of your project's deliverables**
- ◆ **People who will maintain or support your deliverables**

# *Guidelines for Establishing the Audience List*

- ◆ Develop your audience list from a template that worked well in a previous project
- ◆ Eventually instantiate instances from each category with position and name
- ◆ When in doubt, **ADD** a person's name
- ◆ Separately include a person's name for every different role played by him or her
- ◆ Speak with a wide range of people
- ◆ Allow sufficient time to developing your audience list (mainly during project initiation time)
- ◆ Continue to maintain the audience list during the project (remove names, add names)
- ◆ Encourage project participants to identify new candidates

# *Another Categorization of the Audience List*

- ◆ Drivers:
  - ◆ **People who have some say in defining the results of the project**
- ◆ Supporters:
  - ◆ **People who help to perform the activities and tasks of the project. Supporters include those who authorize resources for the project as well as those who work on it.**
- ◆ Observers:
  - ◆ **People who are interested in the activities and results of the project. Observers have no say in the project and they are not actively involved. However, the project may affect them at some point in the future.**
- ◆ Project Champion (Power Promoter):
  - ◆ **A Person who strongly supports the project, advocates it in disputes, takes whatever is necessary to help ensure the successful completion of the project.**

# *Key Concepts for Mapping Roles to People*

- ◆ Authority:
  - ◆ **The ability to make binding decisions between people and roles**
- ◆ Responsibility:
  - ◆ **The commitment to achieve specific results.**
- ◆ Accountability: (rispondere di, rendere conto)
  - ◆ **Tracking a task performance to a participant.**
- ◆ Delegation:
  - ◆ **Binding a responsibility assigned to one person (including yourself) to another person.**

# *Authority vs Responsibility vs Accountability*

## ◆ Authority vs Responsibility

- ◆ **They are similar:** Both are upfront agreements. Before you start a project, you agree on who can make decisions and who will ensure that particular results are achieved.
- ◆ **They are different:** Authority focuses on process such as activities and tasks, responsibility focuses on outcome such as work products and deliverables
- ◆ Good leaders delegate authority; they never delegate responsibility

## ◆ Responsibility vs Accountability:

- ◆ **Similarity:** Both focus on results
- ◆ **Difference:** Responsibility is a before-the-fact agreement, accountability is an after-the-fact process.
- ◆ If you are responsible you should be held accountable.
- ◆ If you are not responsible you should not be held accountable.
- ◆ **Scapegoating:** Making somebody accountable who was not responsible



# *Delegation*

- ◆ **Delegation:** Rebinding a responsibility assigned to one participant (including yourself) to another project participant.
- ◆ Three reasons for delegation:
  - ◆ **Time Management:** To free yourself up to do other tasks
  - ◆ **Expertise:** To have the most qualified person make decisions
  - ◆ **Training:** To develop another person's ability to handle additional assignments.
- ◆ You can delegate authority, but not responsibility
- ◆ You can share responsibility
  - ◆ **Shared relationship between activities and roles can be described in a linear responsibility chart**

- ◆ For successful Lean leadership we need to separate responsibility and authority. This seems strange because we normally think that authority and responsibility are linked together. Could this be another Lean thinking paradox!
- ◆ The focus in a Lean organisation has shifted from “who has the authority” to “what is the right thing to do”. This is achieved by getting each person to take initiative to actually solve problems that improve his or her job, by placing individual responsibility at the lowest possible level where the work is actually done. and ensuring that every person’s job is aligned with providing value for the customer that ultimately leads to prosperity for the company.
- ◆ Our job as a Lean leader is to help expose problems and then make sure people have the skills and the tools to solve these problems. It is more a philosophy of “let’s figure this out together” and creating an environment where learning from mistakes is an accepted part of our continuous improvement process.
- ◆ To help expose problems we must spend more time in the process asking why, and then focus on giving people the responsibility and ownership for developing and implementing the solution. Lean leaders avoid relying on authority, instead leading by influence and example, as if they have no authority.

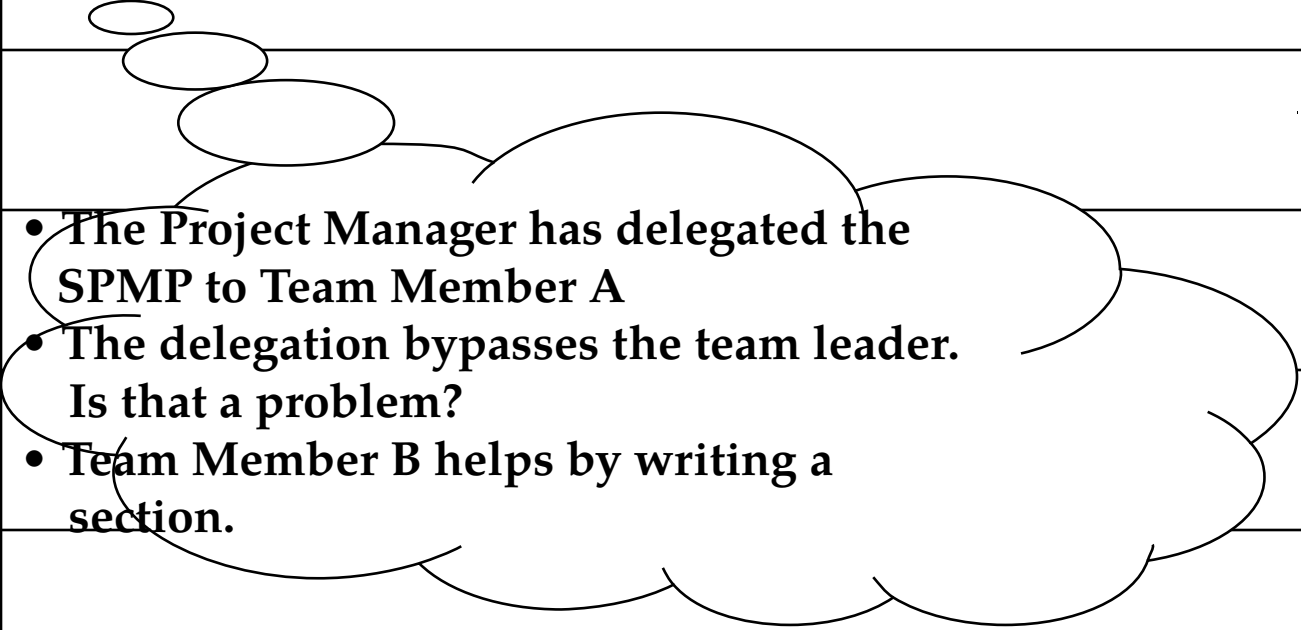
# *Linear Responsibility Chart*

- ◆ A **linear responsibility chart** is a matrix that depicts the role that each project participant will play in different activities identified in the work breakdown structure.
- ◆ **Rows: Project activities**
- ◆ **Columns: Roles/Project participants**
- ◆ **Entries: Type of responsibility**
  - ◆ *P (Primary responsibility)*: You have committed to ensure that the desired result is achieved
  - ◆ *S (Secondary responsibility)*: You have committed to some portion of the result
  - ◆ *A (Approval)*: You are not doing the work, but you will approve what has been done
  - ◆ *R (Review)*: You will review and comment on the work product of an activity
  - ◆ *O (Output)*: You will receive the work product of an activity
  - ◆ *I (Input)*: You will provide input for a task or activity

# *Example of a Responsibility Chart*

	Project Manager	Team Leader	Team Member A	Team Member B
<b>Develop SPMP</b>	<b>P</b>			
<b>Run weekly meeting</b>		<b>A</b>	<b>P</b>	<b>S</b>
<b>Write SDD</b>	<b>P</b>	<b>S</b>	<b>S</b>	<b>S</b>
	<p><i>Legend:</i>  <b>P</b> = Primary responsibility  <b>S</b> = Secondary responsibility)  <b>A</b> = Approval</p>			

# Another Example of a Responsibility Chart

	Project Manager	Team Leader	Team Member A	Team Member B
Develop SPMP	A		P	S
	 <ul style="list-style-type: none"> <li>• The Project Manager has delegated the SPMP to Team Member A</li> <li>• The delegation bypasses the team leader. Is that a problem?</li> <li>• Team Member B helps by writing a section.</li> </ul>			

# *Analysing Responsibility Charts identifies Risks*

- ◆ Problem: Somebody is heavily committed.
  - ◆ **Possible Project Management Issues: Not enough time to handle all duties, making too many key decisions, What if this person leaves during the project**
- ◆ Problem: The project manager has no direct responsibilities
  - ◆ **Issues: Will the project manager fully understand status reports?**
- ◆ Problem: An activity requires many approvals
  - ◆ **Issue: Does anyone else have to approve the activity. Are there too many people involved approvals? Is your estimated duration of the activity too optimistic, because the approval is out of your hands?**
- ◆ After you identify an issue, you should address it in your risk management plan.

# *Micro Management*

- ◆ Micromanagement is the excessive involvement of a manager in the details of a task assigned to a team member.
- ◆ Micromanagement is inefficient use of the time and energy of all project participants.
- ◆ It leads to tension and low morale among all project members.
- ◆ Why do people micromanage?

# *Reasons for Micro Management*

- ◆ The manager is interested in and enjoys the work
- ◆ The manager is a technical expert and feels he/she can do the job best.
- ◆ The manager may feel they did not explain the assignment clearly.
- ◆ The manager is looking for a way to stay involved with the person and or the team.
- ◆ The manager feels threatened because you have more technical knowledge.
- ◆ The manager does not have a clear understanding on how to spend project time.
- ◆ The manager wants to stay up-to-date in case somebody else asks about the work.



# *Overcoming Micro Management*

- ◆ Don't be defensive when the manager asks questions.
  - ◆ **Doing so make it appear as if you are hindering something and the manager will worry even more.**
- ◆ Thank the micromanager for the interest and time.
  - ◆ **Complaining about micromanagement will cause the micromanager to do it even more.**
- ◆ Offer to explain to the micromanager how you will approach your tasks
- ◆ Work with the micromanager to develop a scheme for sharing progress and accomplishments.

# Summary

- ◆ **Organization:** A graph with nodes (organizational units) and different type edges (information structures)
  - ◆ **Functional Organization:** Organizational units are business functions or software process activities („functional model of the organization“)
  - ◆ **Project Organization:** Organizational units are teams. („object model of the organization“)
  - ◆ **Matrix Organization:** Organization that inherits the properties of both, functional and project organizations.
  - ◆ **Hierarchical organization:** Tree with only one type of information structure used for everything (decisions, status, communication).
- ◆ **Project roles in project organizations**
- ◆ **Authority, Responsibility, Accountability, Delegation** („dynamic model of the organization“)
- ◆ **Flexible organization:** Dynamic binding of responsibilities to people
- ◆ **Linear Responsibility Chart:** Shows team roles and responsibilities. Can help to identify and avoid potential difficulties during a project