# Project Management

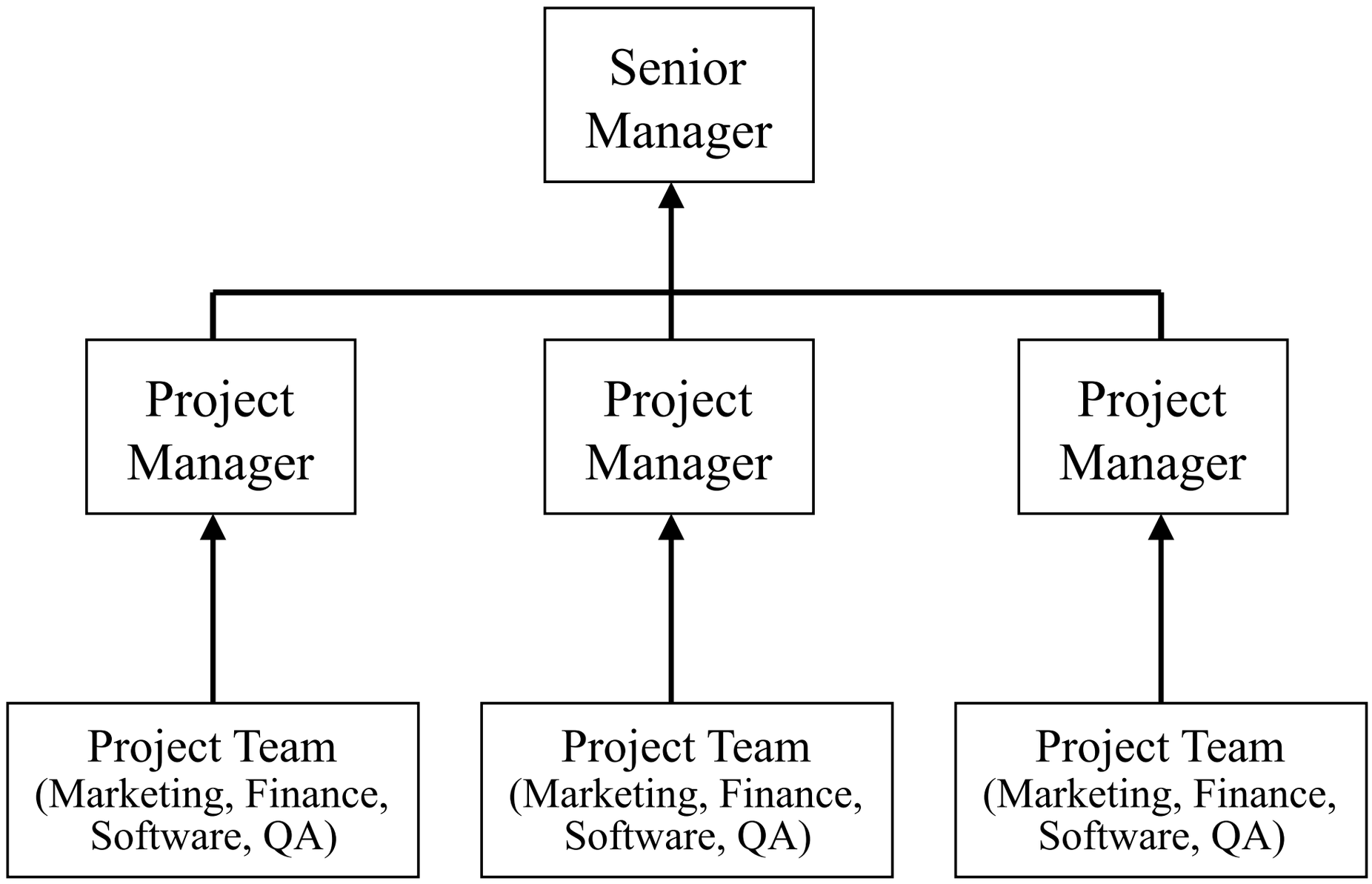
# Organizing Teams

Ragnhild Van Der Straeten - ULB - Software Engineering and Project Management - 2013/2014

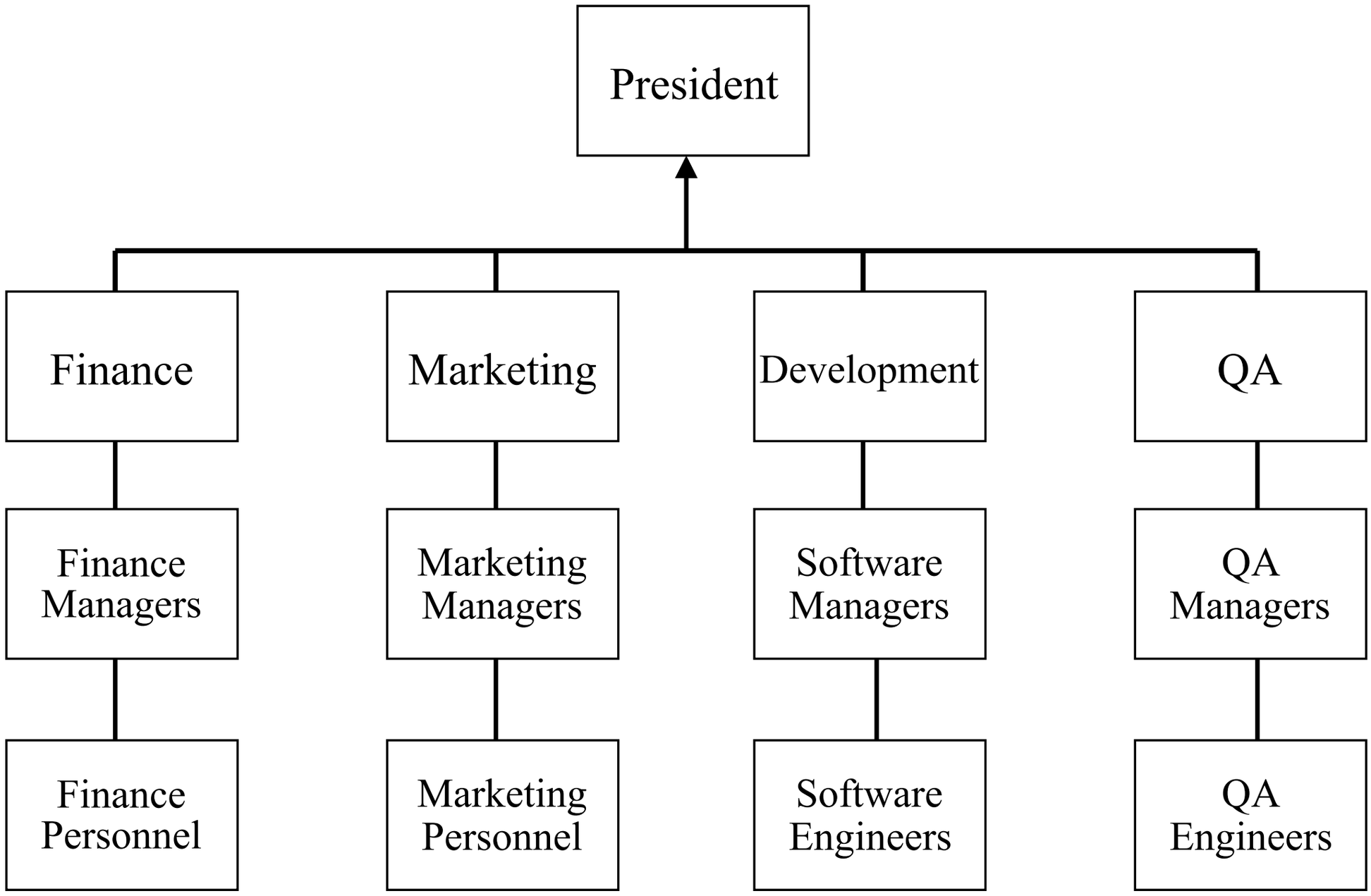
# Software Project Organisation


- three types of organisational structure:
  - project-oriented: personnel are organised around the projects of the company
  - function-oriented:company organised into groups based on their functions
  - matrix-organisation: cross between project- and function-oriented organisations


# Project-oriented organisation



# Function-oriented Organisation


# Matrix Organisation

| | | Project | | | |
|---|---|---|---|---|---|
| | | **Airline reserv. project** | **Bank accounting project** | **Molecular analysis project** | **Fluid mechanics project** |
| **Func-tional Unit** | **Project manage-ment dept** | Al Pruitt<br>Full time | Quinn Parker<br>Full time | Ruth Pella<br>Full time | Fred Parsons<br>Full time |
| | **Marketing dept (Julia Pitt – mgr)** | Oscar Mart<br>Full time | Pete Merrill<br>Full time | Sue More<br>Half time | Elton Marston<br>Full time |
| | **Engineer-ing dept (Joe Roth – mgr)** | Hal Egberts<br>..... | Ben Ehrlich<br>..... | Mary Ericson<br>..... | Len Engels<br>..... |

# Agile Organisation


- agile teams consist of:
  - technical people
- other functions required too:
  - e.g. finance people
    - need to be agile too!!

# Software Teams

- Teams should be relatively small (< 8 members)
  - minimize communication overhead
  - team quality standard can be developed
  - members can work closely together
  - programs are regarded as team property ("egoless programming")
    - continuity can be maintained if members leave
- Break big projects down into multiple smaller projects (hierarchical decomposition)
- Small teams may be organised in an informal, democratic way (ego-less teams)
- Chief programmer teams try to make the most effective use of skills and experience

# Team Roles

- Belbin introduced the notion of Team Roles
  - "A tendency to behave, contribute and interrelate with others in a particular way."
  - need people from different roles to be successful
    - Team of *only* experts will not work
    - Neither will team of only managers

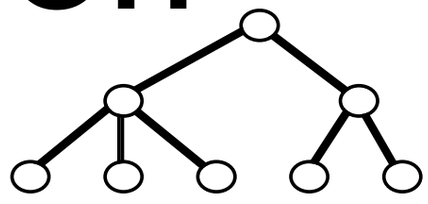
# What roles are proposed?

- Coordinator (co-ordinator and social leader)
- Shaper (gives drive and impetus)
- Plant/Innovator (ideas person)
- Monitor/evaluator (stopping over enthusiasm, missing key points)
- Resource investigator (delicate external negotiations)
- Implementer (turns ideas into practical action)
- Teamworker (diffuses friction)
- Completer/Finisher (progress chaser)
- Specialist (in-depth knowledge of a key area)

# Team Roles in TSP (Humphrey)

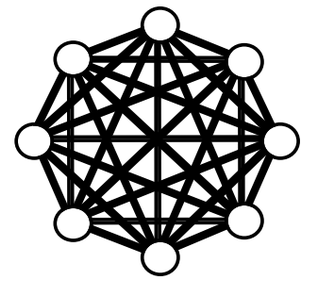

- Team leader
  - build and maintain effective team, motivate, resolve all issues, keep instructor fully informed, perform effectively as meeting facilitator
- Development manager
  - produce a superior product, fully use the team members' skills and abilities
- Planning manager
  - produce a complete, precise, and accurate plan for the team and every team member, accurately report team status every week
- Quality/Process manager
  - properly moderate and report inspections, meetings, control whether team follows TSP process
- Support manager
  - team has suitable tools, no unauthorised changes, team meets its reuse goals all the risks and issues are recorded and reported

# Hierarchical Decomposition



- Properties
  - each member reports to a manager and is responsible to carry out the tasks delegated by the manager
  - because every member is only responsible for his/her tasks, problems could remain unnoticed
  - best suited for big projects with a strict schema and where each person is competent and each role is well-defined

# Ego-less Teams

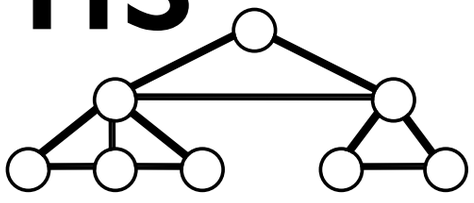


- Properties
  - group is more important than an individual
  - team works together to achieve a common goal
  - decisions are taken by consensus
  - every information is available for each member
  - suited for
    - difficult projects with lots of challenges
    - teams where each member is competent and has lots of experience.

# Ego-less Teams

- Advantages
  - there is collaboration
  - quality standards of the group can be developed
  - each member knows the other members
  - each member knows what the other members are doing
  - each member can improve the programs of the other members.

# Chief Programmer Teams



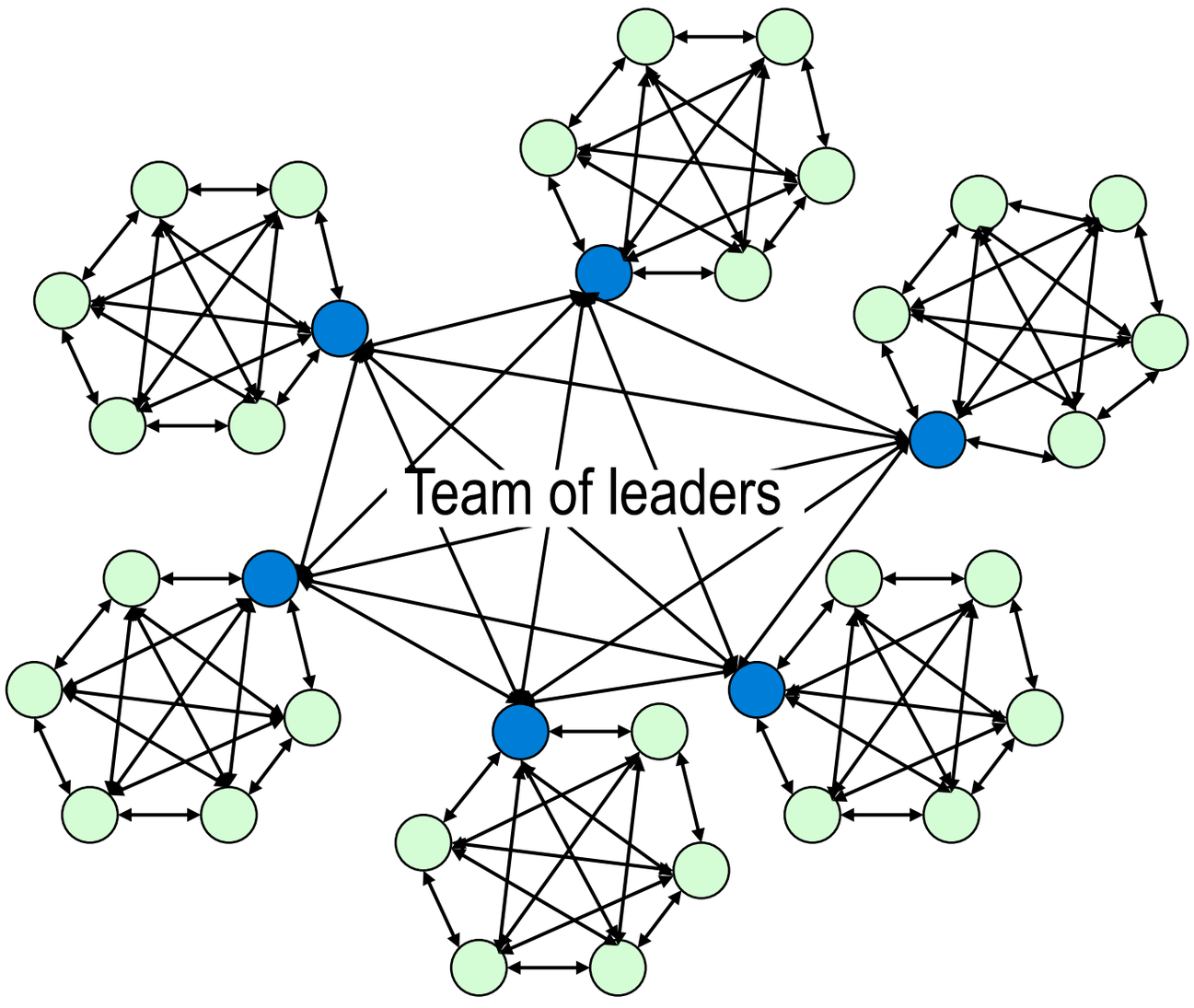
b) Programmeur-chef

- Consist of a kernel of specialists helped by others as required
  - chief programmer takes full responsibility for design, programming, testing and installation of system
  - backup programmer keeps track of CP's work and develops test cases
  - librarian manages all information
  - others may include: project administrator, toolsmith, documentation editor, language/system expert, tester, and support programmers …

# Chief Programmer Teams

- Reportedly successful but problems are:
  - Can be difficult to find talented chief programmers
  - Might disrupt normal organisational structures
  - May be de-motivating for those who are not chief programmers


# Peer Organisation for Larger Projects



# Geographically Distributed Development


- Same office area
  - + ideal for group communication
  - - labor rates sub-optimal
- Same city, different offices
  - + reasonably good communcations
  - + common culture
  - - labor rates suboptimal
- Same country, different cities
  - - communication difficult
  - + common culture
- Multi-country
  - - communication most difficult
  - - culture issues problematical
  - + labor rates optimal

# To Offshore or Not To Offshore


- Benefits of Offshoring
  - lower offshore labor rates
  - work continues during night
  - flexible use of staff
  - use of specialist staff
- Costs of Offshoring
  - communications hampered
  - cultural differences
  - increased project management costs
  - assessment of quality
  - formal procedures

# Task Allocation Example


- In-house team
  - business analysis
  - new feature development and related testing
  - requirements: capturing, documenting and managing
  - system architecture, modeling
  - project management
- Remote subcontractor
  - Testing on pre-release builds
  - Testing maintenance releases of current version
  - build and deployment
  - selected project management
  - unit testing components during maintenance
  - creating and modifying requirements for maintenance

# Tools and Methods for Distributed Development


| REQUIREMENTS | IBM RESPONSE |
|---|---|
| United teams despite diverse languages and cultures | • Enable browser-based access to the same knowledge base for all teams<br>• Provide easy access to guidelines, template and tool mentors based on underlying best practices<br>• Visually communicate discipline workflow and interactions with the Unified Modeling Language (UML) |
| Reduction in work transfer issues | • Implement a framework based on core process workflows from business modeling through deployment<br>• Use a phased approach to software development that details execution for each discipline |
| Easy-to-navigate process that is not overwhelming for users | • Jump-start planning and get new team members up to speed fast with knowledge assets and guidance<br>• Allow users to create personal process views that are central to individual needs<br>• Provide intuitive navigation with a browser-based interface |
| Demonstrated progress toward expected return on investment for GDD projects | • Track metrics throughout the project life cycle<br>• Report on variance measurements and adjust process to achieve desired results<br>• Track project progress and quality through quantitative analysis |
| Ability to assess and manage distributed resources efficiently | • Maintain a broad and deep understanding of an organization's capacity, skills inventory, total workload and resource demand<br>• Optimize skill usage with resource planning to align mission-critical resources with high-priority projects |

http://www3.software.ibm.com/ibmdl/pub/software/rational/web/guides/GC34-2500-00.pdf

# Directing Teams

***Managers serve their team***

Managers ensure that team has the *necessary information and resources*

> *"The manager's function is not to make people work, it is to make it possible for people to work"*
>
> — Tom DeMarco

***Responsibility demands authority***

Managers must *delegate*

Trust your own people and they will trust you.

# Directing Teams

***Managers manage***

Managers cannot perform tasks on the *critical path*

Especially difficult for technical managers!

***Developers control deadlines***

A manager cannot meet a deadline to which the developers have not agreed