Software Team Organizations

Volkan ABUR

Deniz KILINÇ

Overview

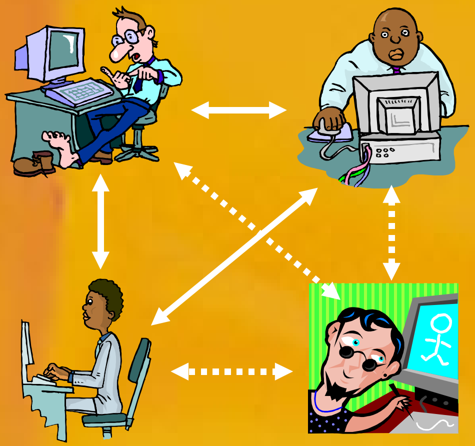
- Team organization
- Democratic team approach
- Classical chief programmer team approach
- Beyond chief programmer and democratic teams
- Synchronize-and-stabilize teams
- Extreme programming teams

Programming Team Organization

- A product must be completed within 3 months, but 1 person-year of programming is still needed
- Solution
  - If one programmer can code the product in 1 year, four programmers can do it in 3 months
- Nonsense
  - Four programmers will probably take nearly a year
  - The quality of the product is usually lower

Mythical Man/Month

- Why can't we calculate team size with?
  - **person/months * time allocated = persons**
- Teams don't scale this way because
  - some tasks can be shared, others cannot



Full task sharing is a requirement for a team to be effective in decreasing time demands

- If one farm hand can pick a strawberry field in 10 days, ten farm hands can pick same strawberry field in 1 day
- One woman can produce a baby in 9 months, but nine women cannot possibly produce that baby in 1 month

Task Sharing

- Unlike baby production, it is possible to share coding tasks between members of team
- Unlike strawberry picking, team members must interact in meaningful and effective way

Programming Team Organization

- Example:
  - Freda and Joe code two modules, mA and mB, say.
- What can go wrong?
  - Both Freda and Joe may code mA, and ignore mB
  - Freda may code mA, Joe may code mB. When mA calls mB it passes 4 parameters; but mB requires 5 parameters
  - Or, the order of parameters in mA and mB may be different
  - Or, the order may be same, but the data types may be slightly different
- This has nothing whatsoever to do with technical competency
  - Team organization is a managerial issue

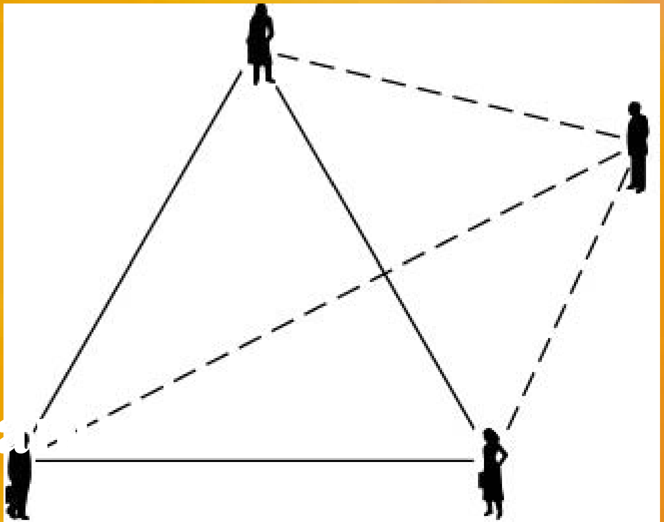
Communications Problems

- Example
  - There are three channels of communication between 3 programmers working on project. The deadline is rapidly approaching but the code is not nearly complete

"Obvious" solution:

Add a fourth programmer

to the tea



Communications Problems

- But other three have to explain in detail
  - What has been accomplished
  - What is still incomplete
- Brooks's Law (popularized in his book)
  - Adding additional programming personnel to a team when product is late has the effect of making the product even later

Team Organization

- Teams are used throughout software production
  - Especially during implementation
  - Here, the discussion is presented within the context of programming teams
- Two extreme approaches to team organization
  - Democratic teams (Weinberg, 1971)
  - Chief programmer teams (Brooks, 1971; Baker, 1972)

Democratic Team Approach

- **Basic underlying concept—*egoless programming***
- **Programmers can be highly attached to their code**
  - **They even name their modules after themselves**
  - **They see their modules as extension of themselves**
  - **They "own" the code- "Don't touch that!"**

Democratic Team Approach

- If a programmer sees a module as an extension of his/her ego, he/she is not going to try to find all the errors in "his"/"her" code
  - If there is an error, it is termed a *bug* 🕷
  - The fault could have been prevented if code had been better guarded against the "bug"
  - "Shoo-Bug" aerosol spray

Democratic Team Approach

- Proposed Solution
- Egoless programming
  - Restructure the social environment
  - Restructure programmers’ values
  - Encourage team members to find faults in code
  - A fault must be considered a normal and accepted event
  - The team as whole will develop an ethos, group identity
  - Modules will “belong” to the team as whole
  - A group of up to 10 egoless programmers constitutes a *democratic team*

Difficulties with Democratic Team Approach

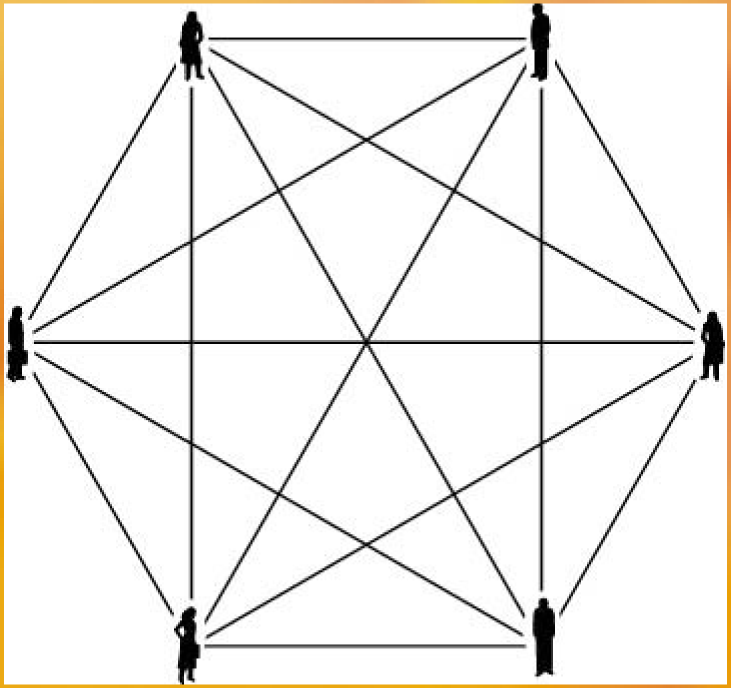
- Management may have difficulty
  - Hard to introduce into an undemocratic environment
  - Is everyone truly equal (in terms of raises, evaluation, promotion, …)???
  - Approach says "Yes"
- Difficult to impose either the composition or the approach from above
- Few teams stay intact for long
  - What happens when some members leave?
  - Will new members bond with the team as well?

Strengths of Democratic Team Approach

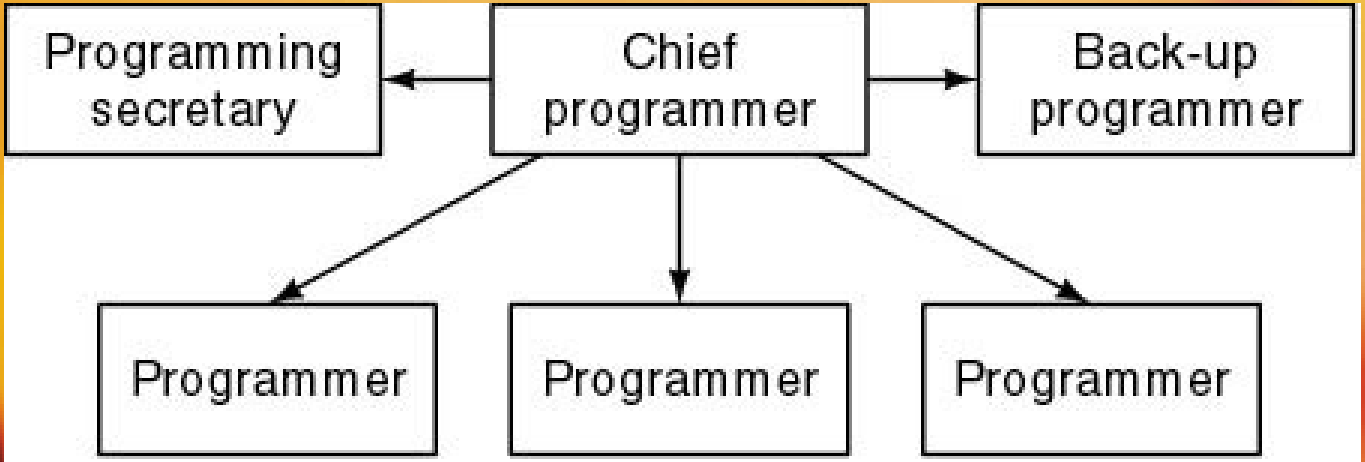
- Democratic teams are enormously productive
- They work best when the problem is difficult
- They function well in a research environment
- Major problem:
  - Democratic teams have to spring up spontaneously

Chief programmer teams- introduced in 1971

- Consider a 6-person team
  - Fifteen 2-person communication channels
  - The total number of 2-, 3-, 4-, 5-, and 6-person groups is 57
  - The team cannot do 6 person-months of work in 1 month



Chief Programmer Teams



- **Six programmers, but now only 5 lines of communication**
- **Basic idea behind the concept:**
  - **Chief surgeon directing operation, assisted by Other surgeons, anesthesiologists,Nurses, Other experts, such as cardiologists**
- **Two key aspects**
  - **Specialization**
  - **Hierarchy**

Classical Chief Programmer Teams

- Chief programmer
  - Successful manager *and* highly skilled programmer
  - Does the architectural design
  - Allocates coding among the team members
  - Writes the critical (or complex) sections of code
  - Handles all the interfacing issues
  - Reviews the work of the other team members
  - Is personally responsible for every line of code

Classical Chief Programmer Teams

- Back-up programmer
  - Necessary only because the chief programmer is human
  - The back-up programmer must be in every way as competent as the chief programmer
  - Must know as much about the project as the chief programmer
  - Does black-box test case planning and other tasks that are independent of the design process

Classical Chief Programmer Teams

- Programming secretary
  - A highly skilled, well paid, central member of the chief programmer team
  - Responsible for maintaining the program production library (documentation of project), including:
    - Source code listings
    - JCL
    - Test data
  - Programmers hand their source code to the secretary who is responsible for
    - Conversion to machine-readable form,
    - Compilation, linking, loading, execution, and running test cases (originally introduced in 1971, remember!)

Classical Chief Programmer Teams

- Programmers
  - Do nothing but program
  - All other aspects are handled by the programming secretary
  - Often fairly new to the organization and not experienced
  - Allows new personnel to "learn the ropes" under mentors

The New York Times Project

- Chief programmer team concept
  - first used in 1971
  - by IBM
  - to automate the clippings data bank (“morgue“) of *The New York Times*
- Chief programmer—F. Terry Baker

The New York Times Project

- 83,000 source lines of code (LOC) were written in 22 calendar months, representing 11 person-years
- After the first year, only the file maintenance system had been written (12,000 LOC)
- Most code was written in the last 6 months
- Only 21 faults were detected in the first 5 weeks of acceptance testing
- Only 25 further faults were detected in the first year of operation

The New York Times Project

- Principal programmers averaged one detected fault and 10,000 LOC per person-year
- The file maintenance system, delivered 1 week after coding was completed, operated 20 months before a single failure occurred
- Almost half the subprograms (usually 200 to 400 lines of PL/I) were correct at first compilation

The New York Times Project

- But, after this fantastic success, no comparable claims for the classical chief programmer team concept have been made
- Why?
- It is not unusual for "new methods" to produce a "halo effect"
- What was unusual about this project? ....

Why Was the NYT project Such a Success?

- Prestige project for IBM
  - First real trial for PL/I (developed by IBM)
  - IBM, with superb software experts, used its best people
- Very strong technical backup
  - PL/I compiler writers helped the programmers
  - JCL experts assisted with the job control language

Why Was the NYT project Such a Success?

- F. Terry Baker
  - Superprogrammer
  - Superb manager and leader
  - His skills, enthusiasm, and personality “carried” the project
- Strengths of CPT Approach
  - It works if all is right
  - Numerous successful projects have used, however, variants of CPT

Impracticality of Classical CPT

- Chief programmer must be a highly skilled programmer and a successful manager
  - Shortage of highly skilled programmers
  - Shortage of successful managers
  - Programmers and managers “are not made that way”

Impracticality of Classical CPT

- Back-up programmer must be as good as the chief programmer
  - But he/she must take a back seat (and normally this means a lower salary) waiting for something to happen to the chief programmer
  - Top programmers, top managers will not do that
- Programming secretary does only paperwork all day
  - Software professionals hate paperwork!
- Classical CPT has been found to be impractical
- However, notes it works in a surgical environment

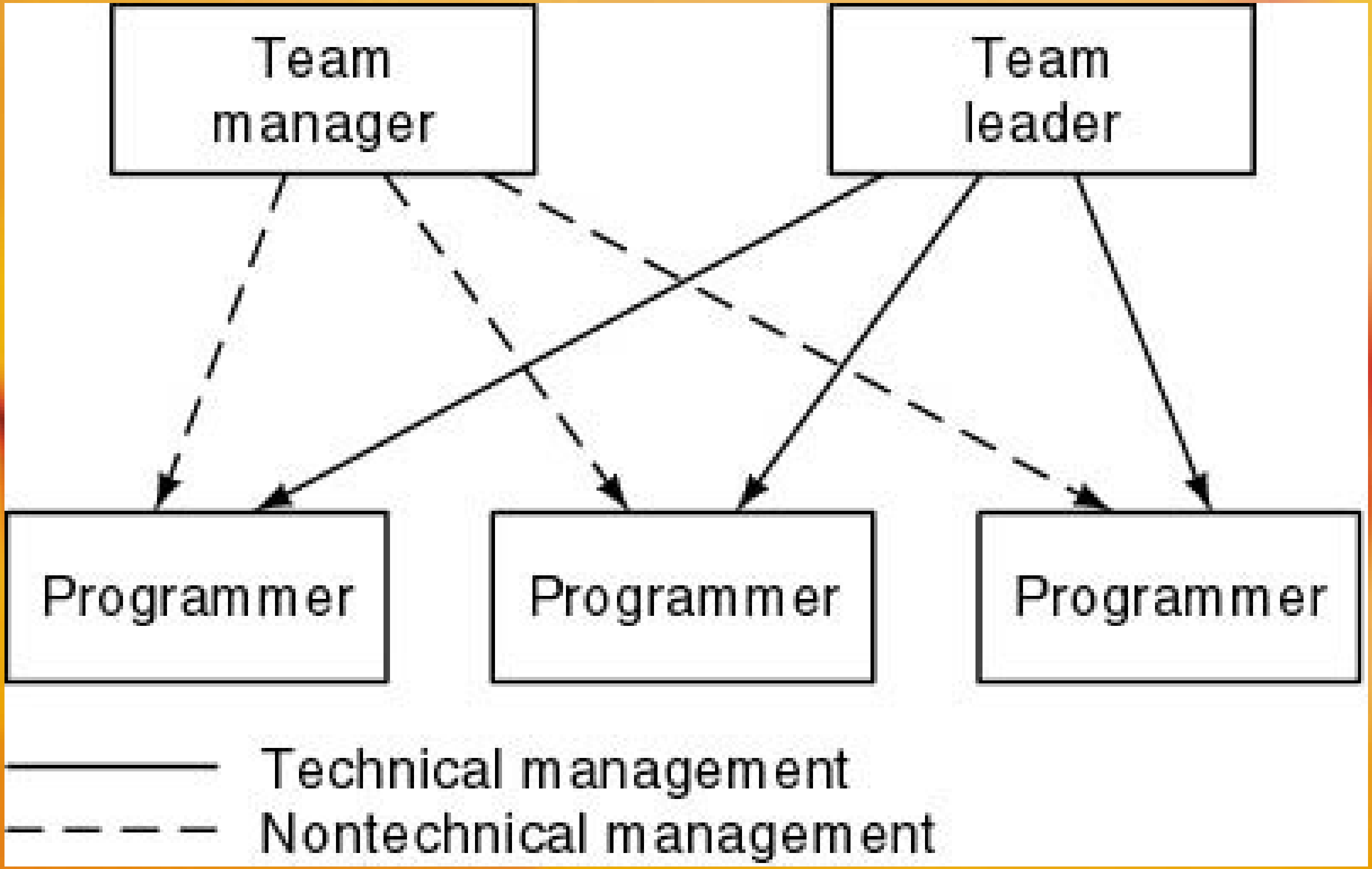
Beyond CP and Democratic Teams

- We need ways to organize teams that
  - Make use of the strengths of democratic teams and chief programmer teams, and
  - Can handle teams of 20 (or 120) programmers
- Democratic teams
  - Positive attitude to finding faults
- Use CPT in conjunction with code walkthroughs or inspections

Beyond CP and Democratic Teams

- Potential Pitfalls:
- Chief programmer is personally responsible for every line of code.
  - He/she must therefore be present at reviews
- Chief programmer is also the team manager
  - He/she must *not* be present at reviews!
- How can we handle this conflict in roles?
- One answer: split the chief programmer into two people, differentiated by the roles played.

Beyond CP and Democratic Teams (c



- One solution
  - Reduce the managerial role of the chief programmer

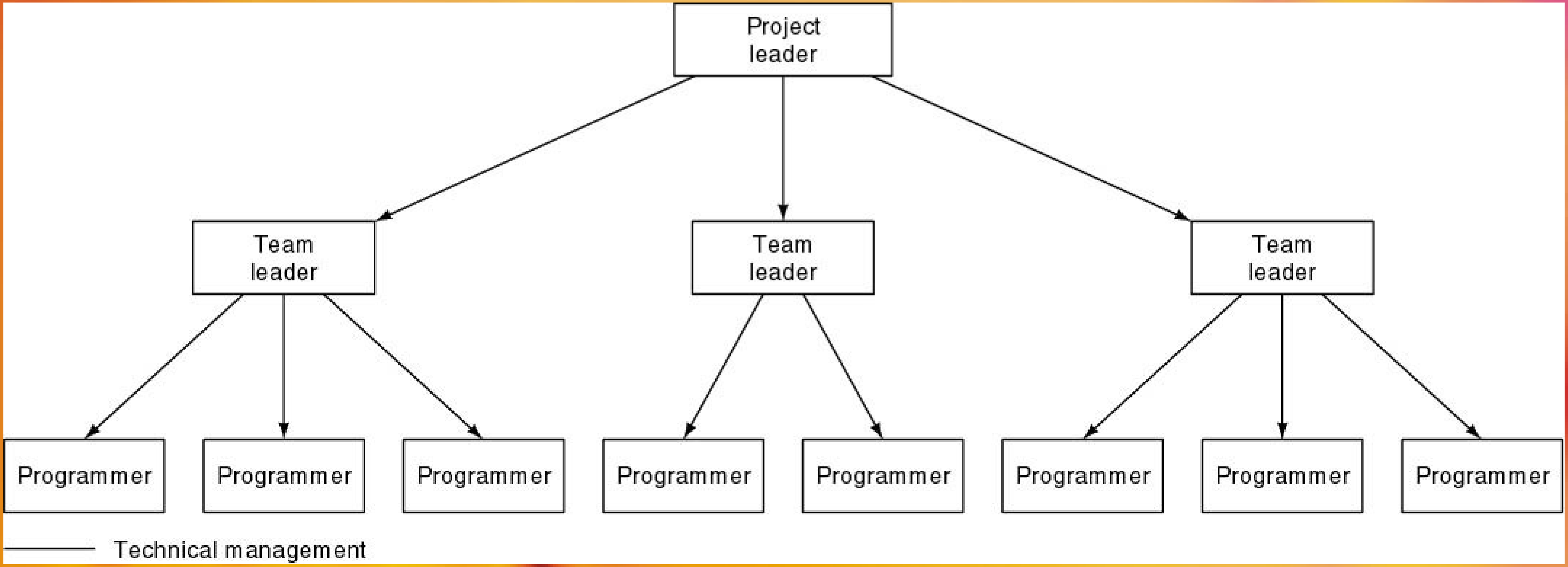
Beyond CP and Democratic Teams

- It is easier to find a team leader than a chief programmer
- Each employee is responsible to exactly one manager—lines of responsibility are clearly delineated
- Team leader is responsible for only technical management

Beyond CP and Democratic Teams

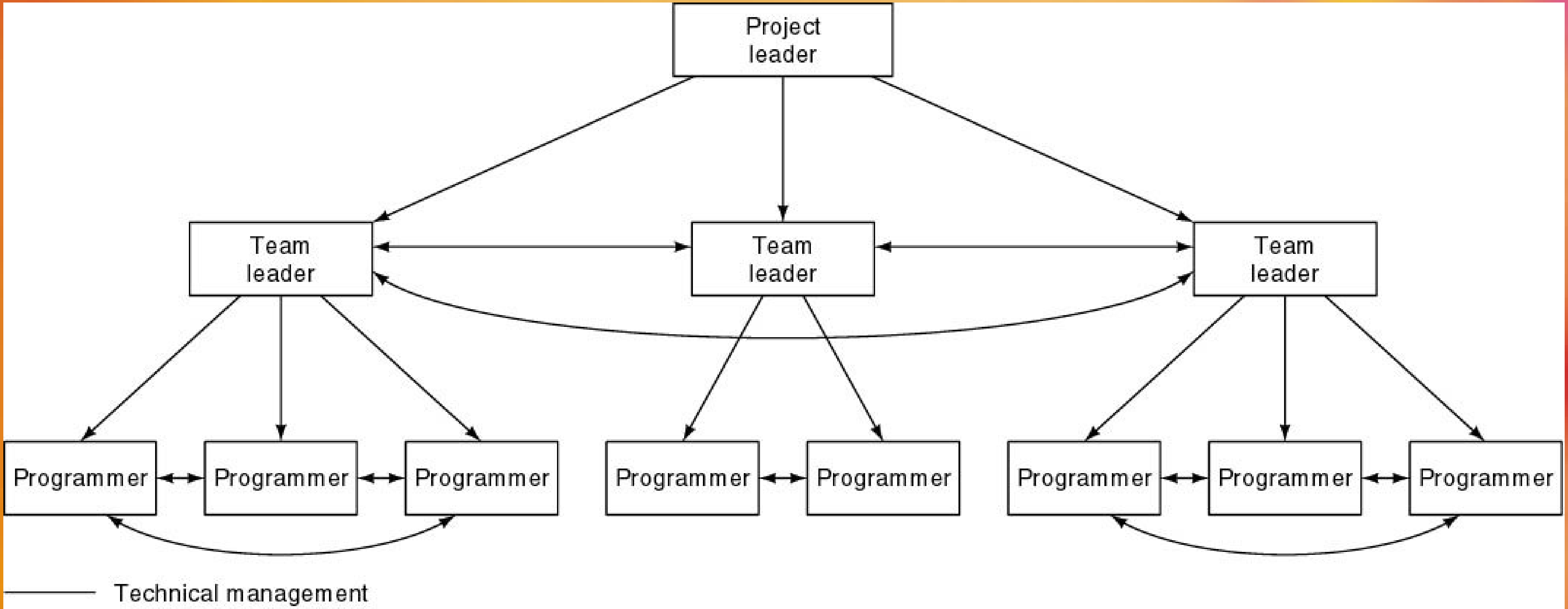
- Budgetary and legal issues and performance appraisal are not handled by the team leader
- Team leader participates in reviews—the team manager is not permitted to do so
- Team manager participates at regular team meetings to appraise the technical skills of the team members

Larger Projects- Technical Side



- **Non-technical side is similar**
- **For even larger products, add additional layers**
- **Be careful- many companies call the manager the Project Leader and this one the Technical Leader.**

Beyond CP and Democratic Teams



- Decentralize the decision-making process where appropriate
- Useful where the democratic team concept works

Synchronize-and-Stabilize Teams

- Used by Microsoft
- Products consist of 3 or 4 sequential builds
- Small parallel teams
  - 3 to 8 developers
  - 3 to 8 testers (work one-to-one with developers)
  - Team is given the overall task specification
  - They may design the task as they wish
- Why this does not degenerate into hacker-induced chaos
  - Daily synchronization step enforced strictly
  - Individual components must work together or work continues
  - Let children do what they like all day...... but with a 9 P.M. bedtime!

Synchronize-and-Stabilize Teams

- Will this work in all companies?
  - Perhaps if the software professionals are as good as many at Microsoft
  - Again, more research is needed
  - But, research is *extremely* difficult to carry out!

Extreme Programming Teams

- Feature of XP
  - All code is written by two programmers sharing a computer
  - “Pair programming”
- Advantages of pair programming
  - Test cases drawn up by one member of team
  - Knowledge not all lost if one programmer leaves
  - Inexperienced programmers can learn from others
  - Centralized computers promote egoless programming

Teams vs. Groups

- Groups
  - strong leader
  - individual accountability
  - organizational purpose
  - individual work products
  - efficient meetings
  - measures performance by influence on others
  - delegates work

- Teams
  - shared leadership
  - individual & mutual accountability
  - specific team purpose
  - collective work products
  - open-ended meetings
  - measures performance from work products
  - does real work together

- Teams & good performance are inseparable
  - a team is more than the sum of its parts

Keys to Team Success

- Common commitment
  - requires a purpose in which team members can believe
    - "prove that all children can learn", "revolutionizing …"
- Specific performance goals
  - comes directly from the common purpose
    - "increasing the scores of graduates form 40% to 95%"
  - helps maintain focus – start with something achievable
- A right mix of skills
  - technical/functional expertise
  - problem-solving & decision-making skills
  - interpersonal skills
- Agreement
  - action item assignment, when to meet & work, schedules

Final Remarks

- There is no one solution to the problem of team organization
- The "correct" way seems to depends too much on
  - The product
  - The outlook of the leaders of the organization
  - Previous experience with various team structures
- More research is needed to truly compare structures
- Most of the research has concentrated on the programming teams.

WHERE NOW?

- **At any given moment, there are often many possible projects waiting to be chosen.**
- **The selection of which project to move forward will be an upper management decision based on many factors.**
  - P the need to improve performance
  - I the need to improve information (and data)
  - E the need to improve economics, control costs, or increase profits
  - C the need to improve control or security
  - E the need to improve efficiency of people and processes
  - S the need to improve service to customers, suppliers, partners, employees, etc.

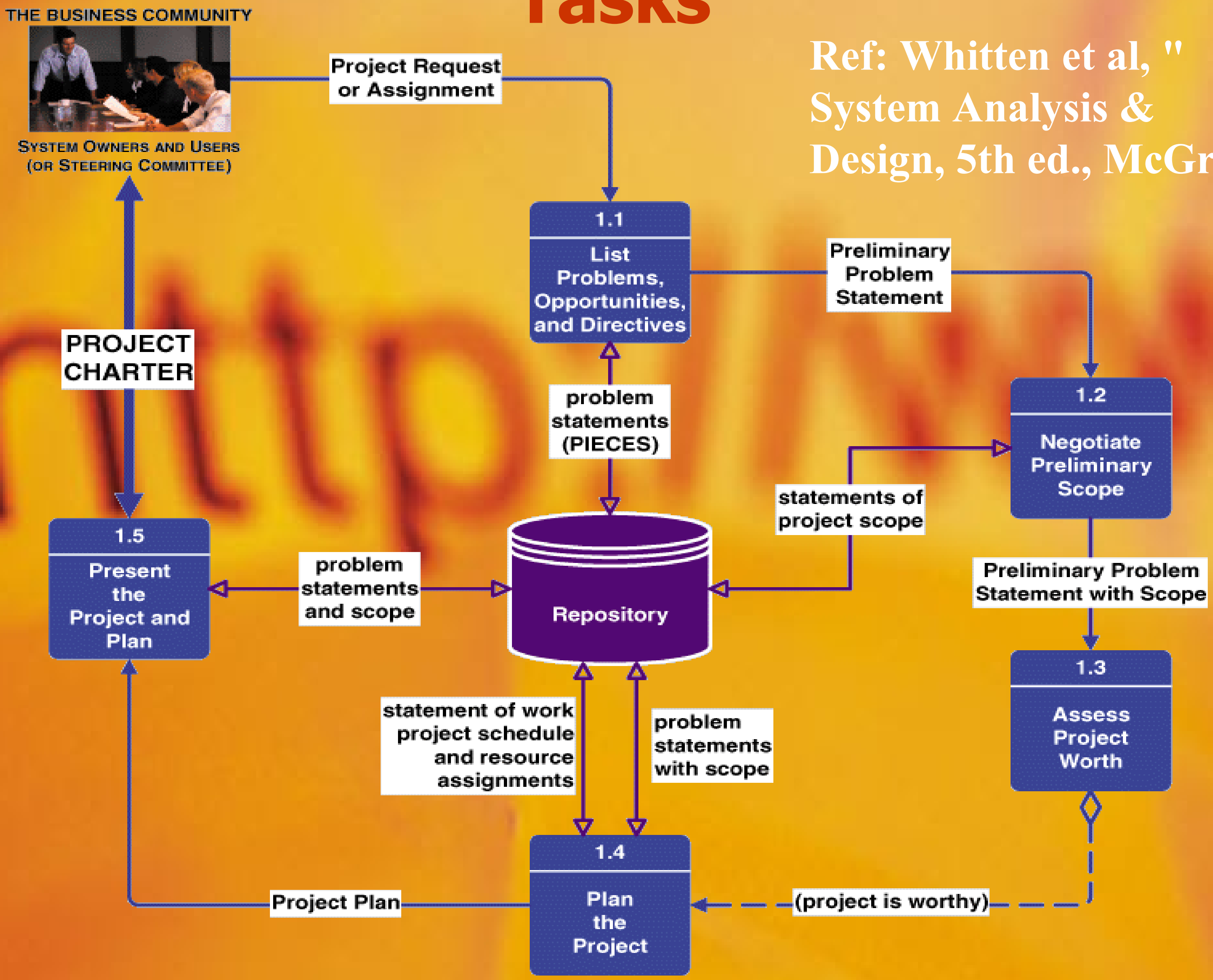
EACH OF THESE CAN GENERATE PROJECTS

- **Problems** are undesirable situations that prevent the organization from fully achieving its purpose, goals, and/or objectives.
- **Opportunities** are chances to improve the organization even in the absence of specific problems.
- **Directives** are new requirements that are imposed by management, government, or some external influence.

Every organization should have a well-defined strategy for selecting which projects to initiate.

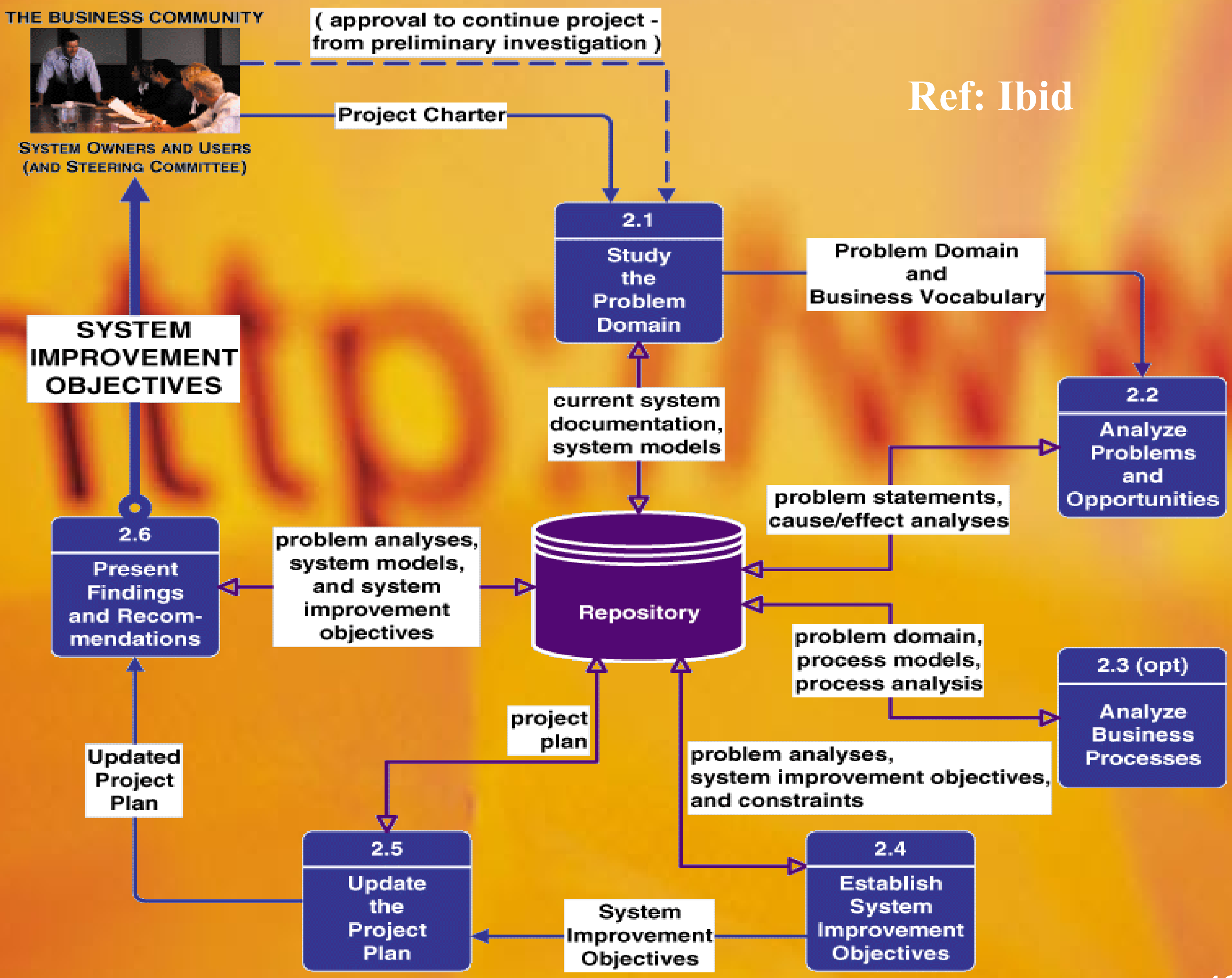
Preliminary Investigation Phase Tasks

Ref: Whitten et al, " System Analysis & Design, 5th ed., McGraw


THE BUSINESS COMMUNITY
SYSTEM OWNERS AND USERS
(OR STEERING COMMITTEE)
Project Request or Assignment
1.1
List Problems, Opportunities, and Directives
Preliminary Problem Statement
1.2
Negotiate Preliminary Scope
PROJECT CHARTER
problem statements (PIECES)
statements of project scope
Preliminary Problem Statement with Scope
1.5
Present the Project and Plan
problem statements and scope
Repository
1.3
Assess Project Worth
statement of work project schedule and resource assignments
problem statements with scope
1.4
Plan the Project
Project Plan
(project is worthy)

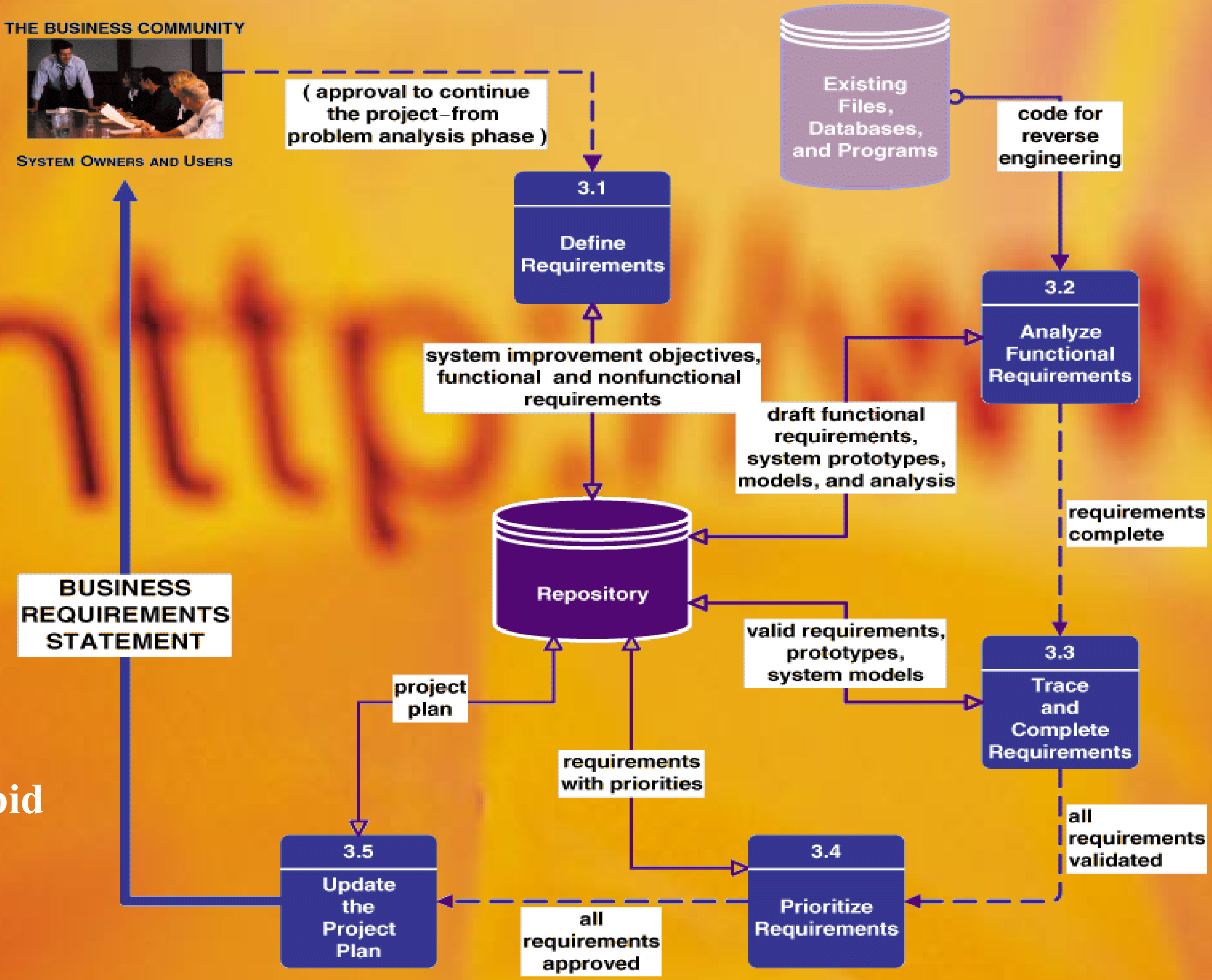
Problem Analysis Phase Context

Ref: Ibid



THE BUSINESS COMMUNITY
SYSTEM OWNERS AND USERS (AND STEERING COMMITTEE)
(approval to continue project - from preliminary investigation)
Project Charter
2.1
Study the Problem Domain
Problem Domain and Business Vocabulary
2.2
Analyze Problems and Opportunities
current system documentation, system models
problem statements, cause/effect analyses
SYSTEM IMPROVEMENT OBJECTIVES
2.6
Present Findings and Recom-mendations
problem analyses, system models, and system improvement objectives
Repository
problem domain, process models, process analysis
2.3 (opt)
Analyze Business Processes
project plan
Updated Project Plan
problem analyses, system improvement objectives, and constraints
2.5
Update the Project Plan
System Improvement Objectives
2.4
Establish System Improvement Objectives

Requirements Analysis Phase Tasks


THE BUSINESS COMMUNITY
System Owners and Users
(approval to continue the project–from problem analysis phase)
Existing Files, Databases, and Programs
code for reverse engineering
3.1
Define Requirements
3.2
Analyze Functional Requirements
system improvement objectives, functional and nonfunctional requirements
draft functional requirements, system prototypes, models, and analysis
requirements complete
Repository
BUSINESS REQUIREMENTS STATEMENT
valid requirements, prototypes, system models
3.3
Trace and Complete Requirements
project plan
requirements with priorities
all requirements validated
3.5
Update the Project Plan
3.4
Prioritize Requirements
all requirements approved

Ref: Ibid

The End

Questions?

MSF
(Microsoft Solutions Framework)

Team Model

Problems, problems, problems…

"The project was late and over budget"

"It doesn't meet our expectations – we're not happy"

"What was built really isn't what we needed"

"We didn't understand clearly what we were supposed to do"

"We couldn't get the information we needed to do our work"

"We were unaware of how the work of other team members affected our work"

"It's just too difficult to use"

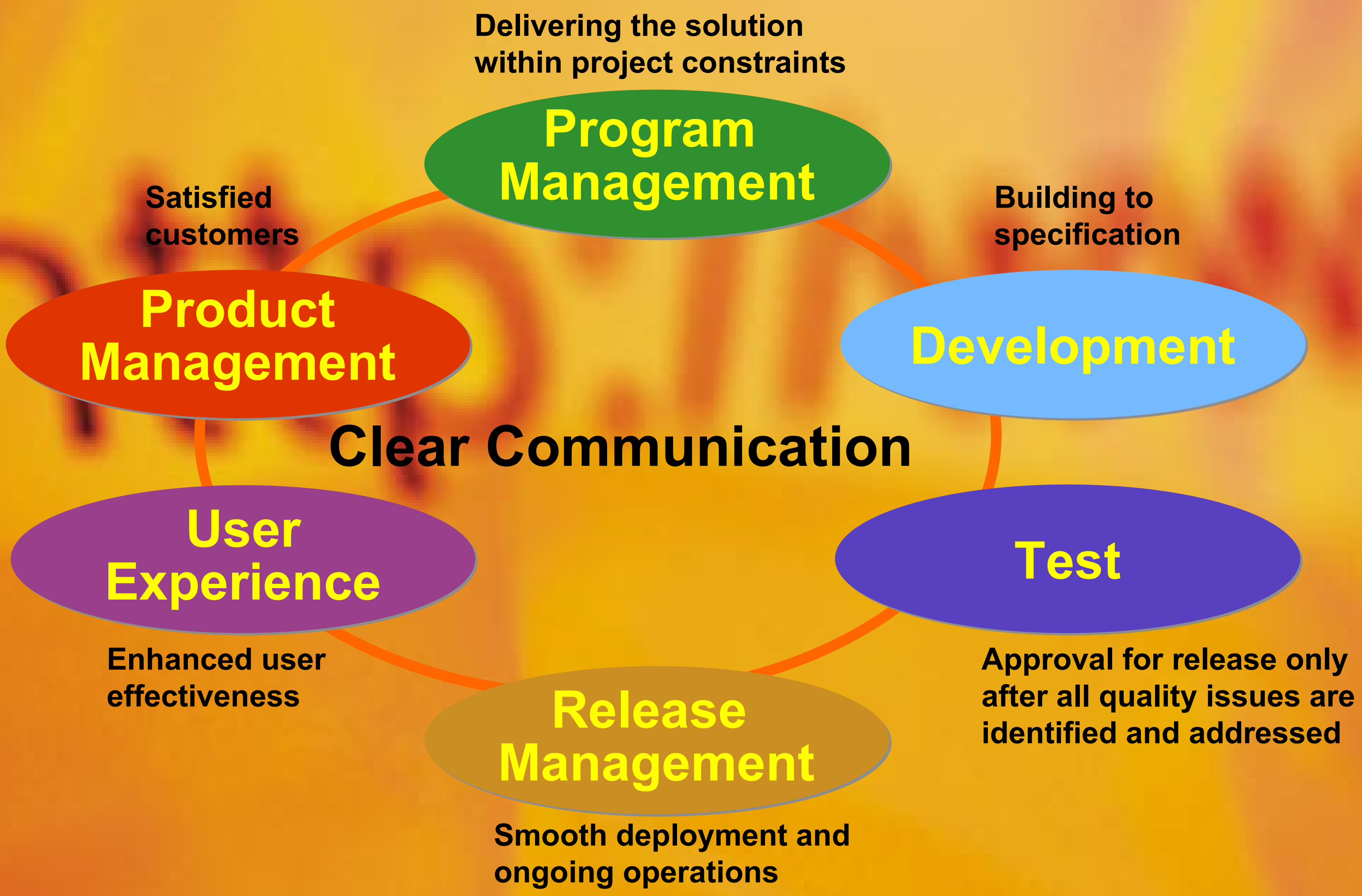
"We can't get it to operate well in our environment"

"This thing is unpredictable – we keep discovering new problems"

2W, 1H (What, Who, How)

Problems	Goals to Success	Owner
"The project was late and over budget"	***Deliver within project constraints***	?
"What was built really isn't what we needed"	***Build to specifications***	?
"This thing is unpredictable – we keep discovering new problems"	***Release with issues identified and addressed***	?
"We can't get it to operate well in our environment"	***Deploy smoothly and prepare well for ongoing operations***	?
"It's just too difficult to use"	***Enhance user effectiveness***	?
"It doesn't meet our expectations – we're not happy"	***Satisfy customers***	?
"Needed information is not shared timely to all who need it"	***Establish good communications***	?

MSF Team Model



Delivering the solution within project constraints

Program Management

Satisfied customers

Product Management

Building to specification

Development

Clear Communication

User Experience

Test

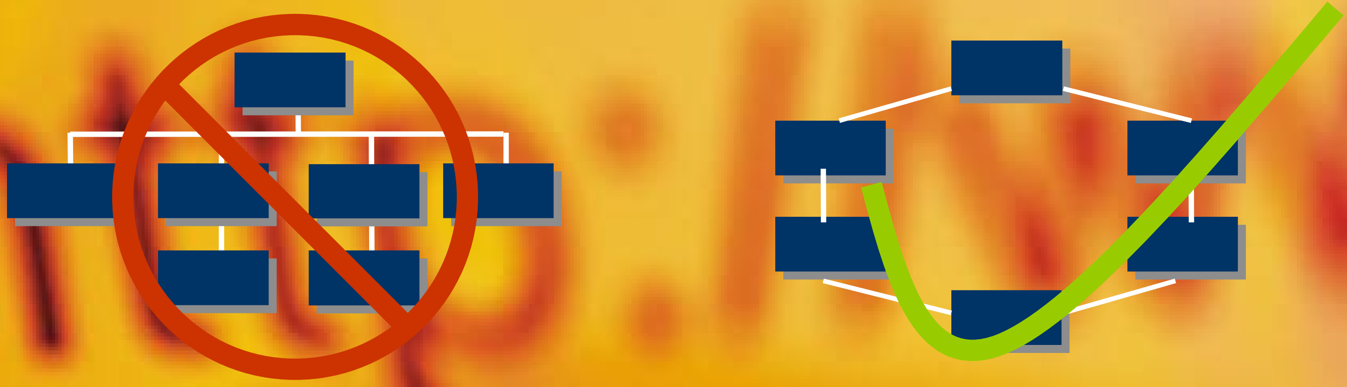
Enhanced user effectiveness

Approval for release only after all quality issues are identified and addressed

Release Management

Smooth deployment and ongoing operations

MSF Team Model Hierarchy



-No hierarchy between project members

-Everyone is equal

External Stakeholders

- Project sponsors
- Customers (business sponsors)
- End users
- Operations
- …

Team Model – Principles

- Work toward a shared vision
- Focus on business value
- Stay agile, expect change
- Empower team members
- Foster open communications
- Establish clear accountability, shared responsibility

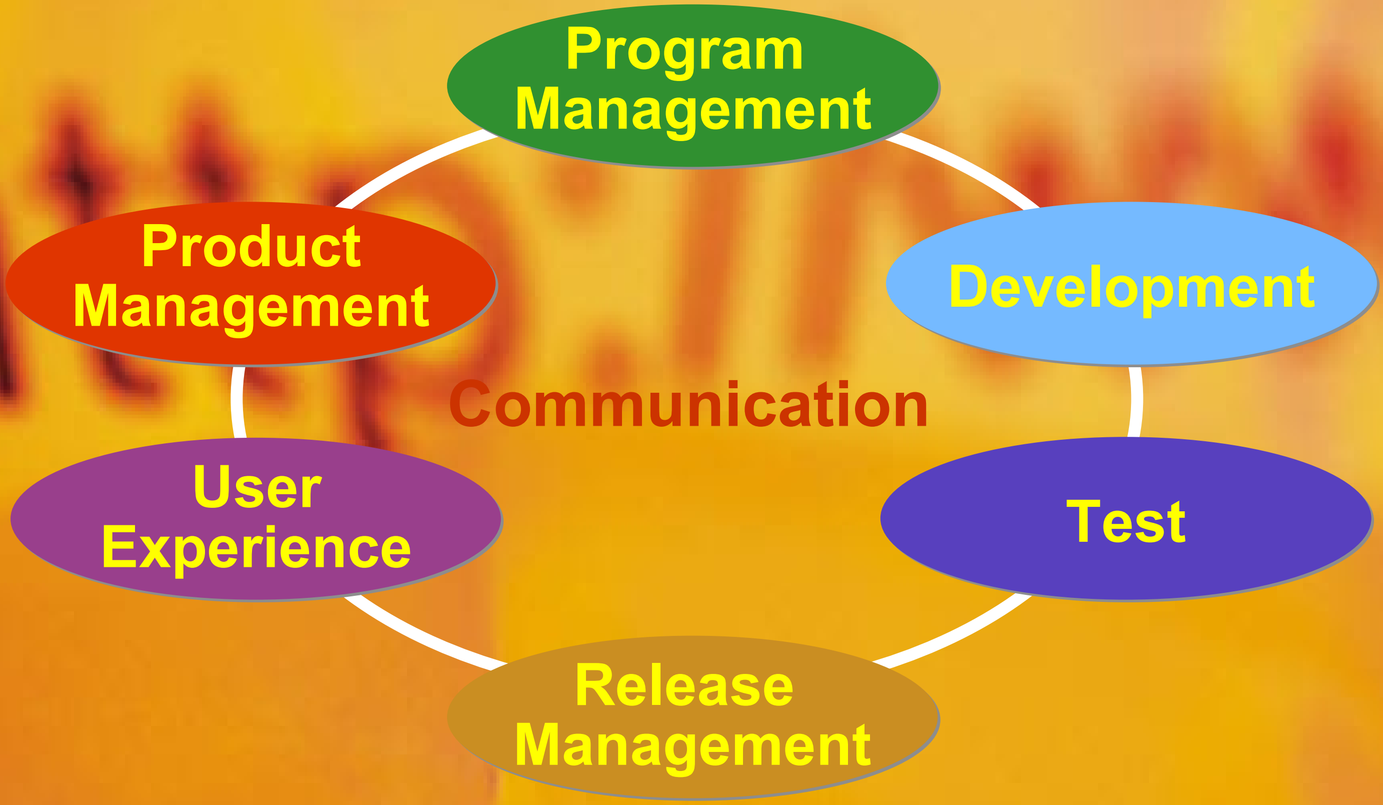
Team Model – Key Concepts

- Team of peers
- Customer-focused mindset
- Product mindset
- Zero defect mindset
- Willingness to learn

Team Model – Proven Practices

- Use small, interdisciplinary teams
- Enable teams to work together at a single site
- Create a soultion design through total team participation

Team Model – Role Clusters



Role cluster (role)

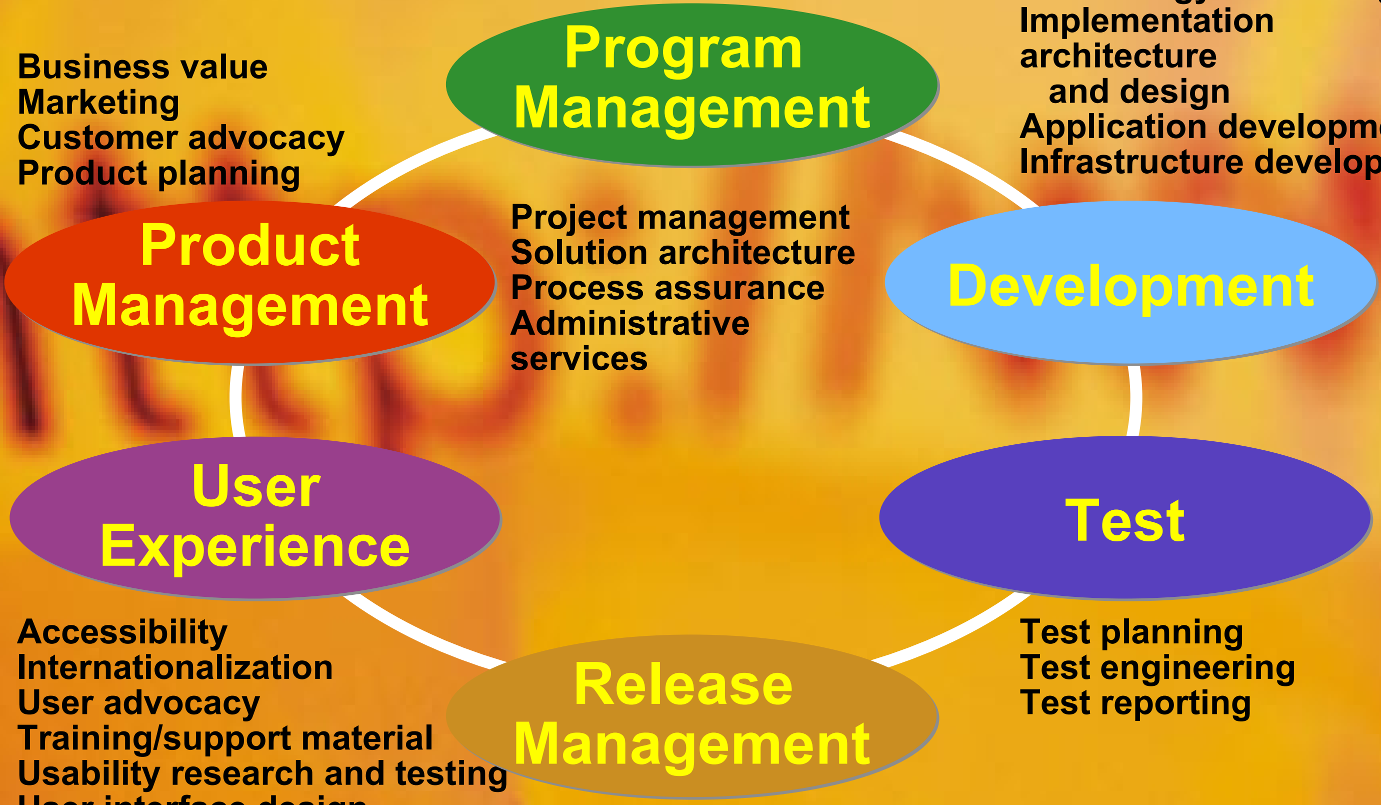
- Functional areas
  - Responsibilities
    - Tasks

Example

Program management

- Project management
- Solution architecture
  - Drive overall solution design
  - Manage functional specification
    - Maintain traceability map
    - Liaise with other project teams on interoperability issues

Functional Areas of Role Clusters



Program Management

Project management
Solution architecture
Process assurance
Administrative services

Development

Technology consulting
Implementation architecture and design
Application development
Infrastructure development

Test

Test planning
Test engineering
Test reporting

Release Management

Infrastructure
Support
Operations
Logistics
Commercial release management

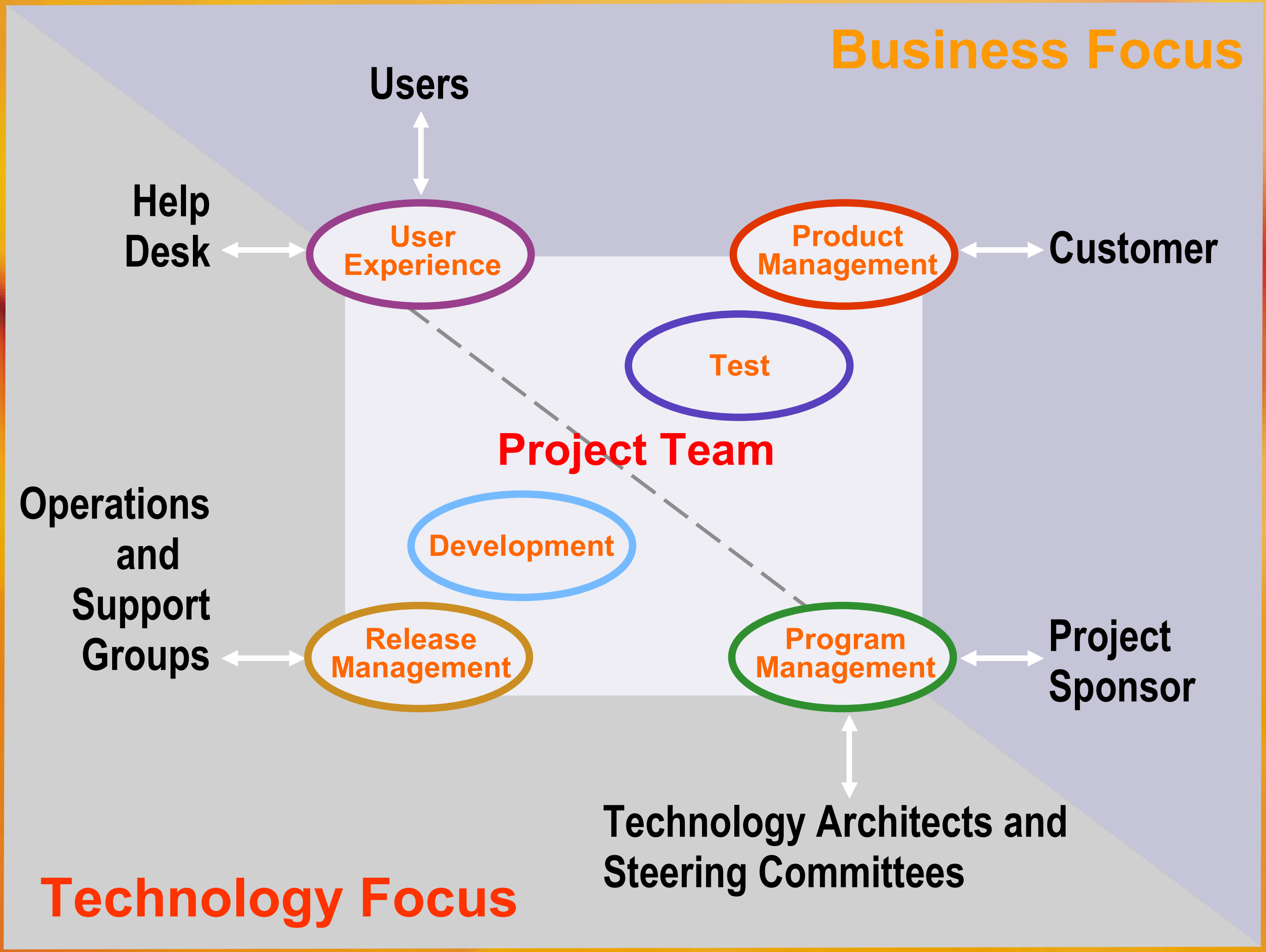
User Experience

Accessibility
Internationalization
User advocacy
Training/support material
Usability research and testing
User interface design

Product Management

Business value
Marketing
Customer advocacy
Product planning

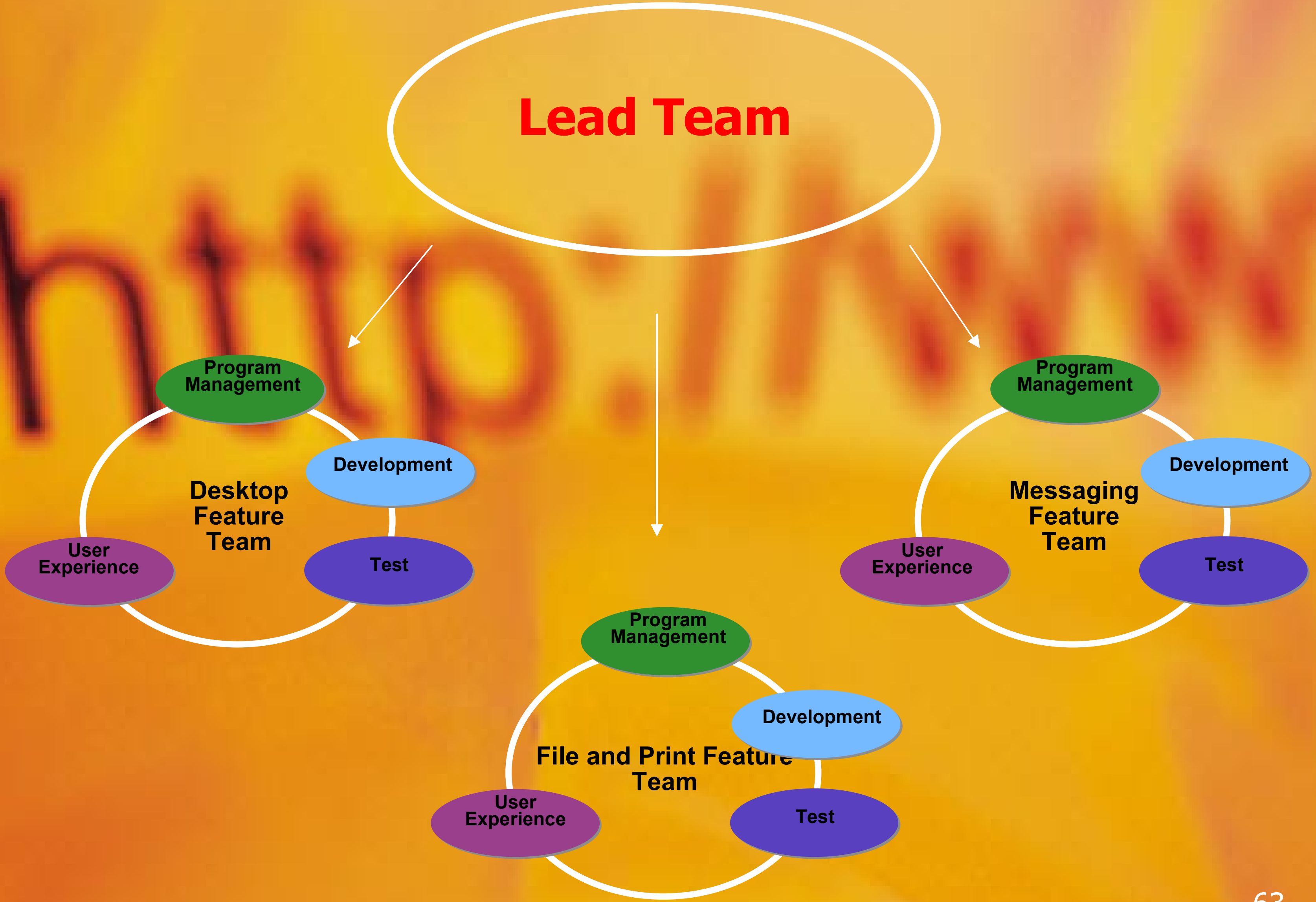
Extended Team



Ways to Scale Up Teams

- Use factors such as complexity, size, risk, and skills for scaling
- Divide large teams into smaller teams, which have lower process, management, and communication overhead and allow faster implementation
- Designate team leads for sub-teams
- Use core team to manage overall project
  - Core team is composed of team leads and program management
  - Core team coordinates and synchronizes sub-teams

Lead and Feature Teams



Lead Team

Program Management
Development
Desktop Feature Team
User Experience
Test

Program Management
Development
Messaging Feature Team
User Experience
Test

Program Management
Development
File and Print Feature Team
User Experience
Test

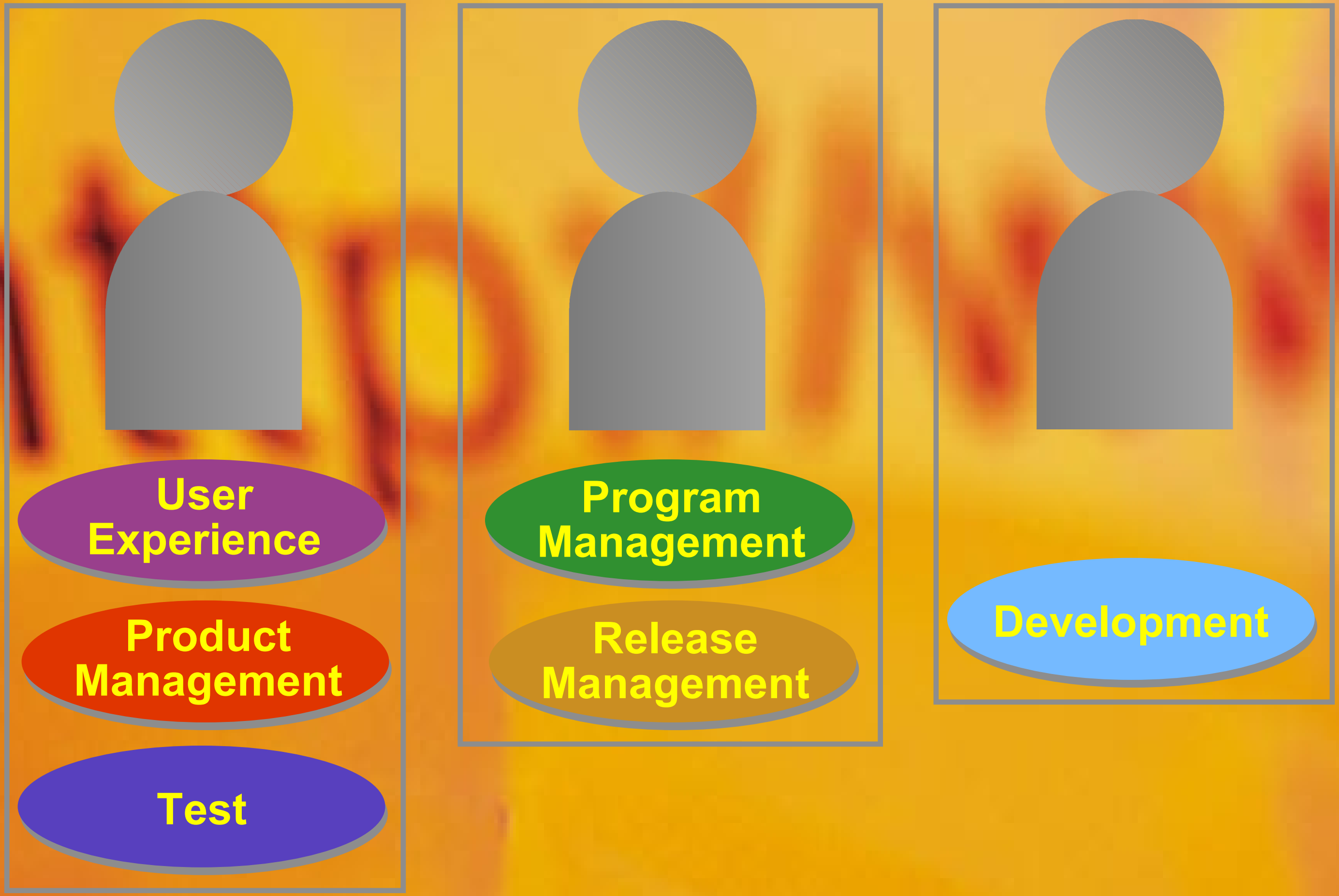
Combining Roles for Small Teams

Roles *may* be combined, but some combinations pose *risks*

	Product Management	Program Management	Development	Test	User Experience	Release Management
Product Management		N	N	P	P	U
Program Management	N		N	U	U	P
Development	N	N		N	N	N
Test	P	U	N		P	P
User Experience	P	U	N	P		U
Release Management	U	P	N	P	U	

P Possible **U** Unlikely **N** Not Recommended

Small Team Example



User Experience

Product Management

Test

Program Management

Release Management

Development