



University of Southern California  
**Center for Software Engineering**

---

# **Theory W Software Management**

**Barry Boehm, USC**

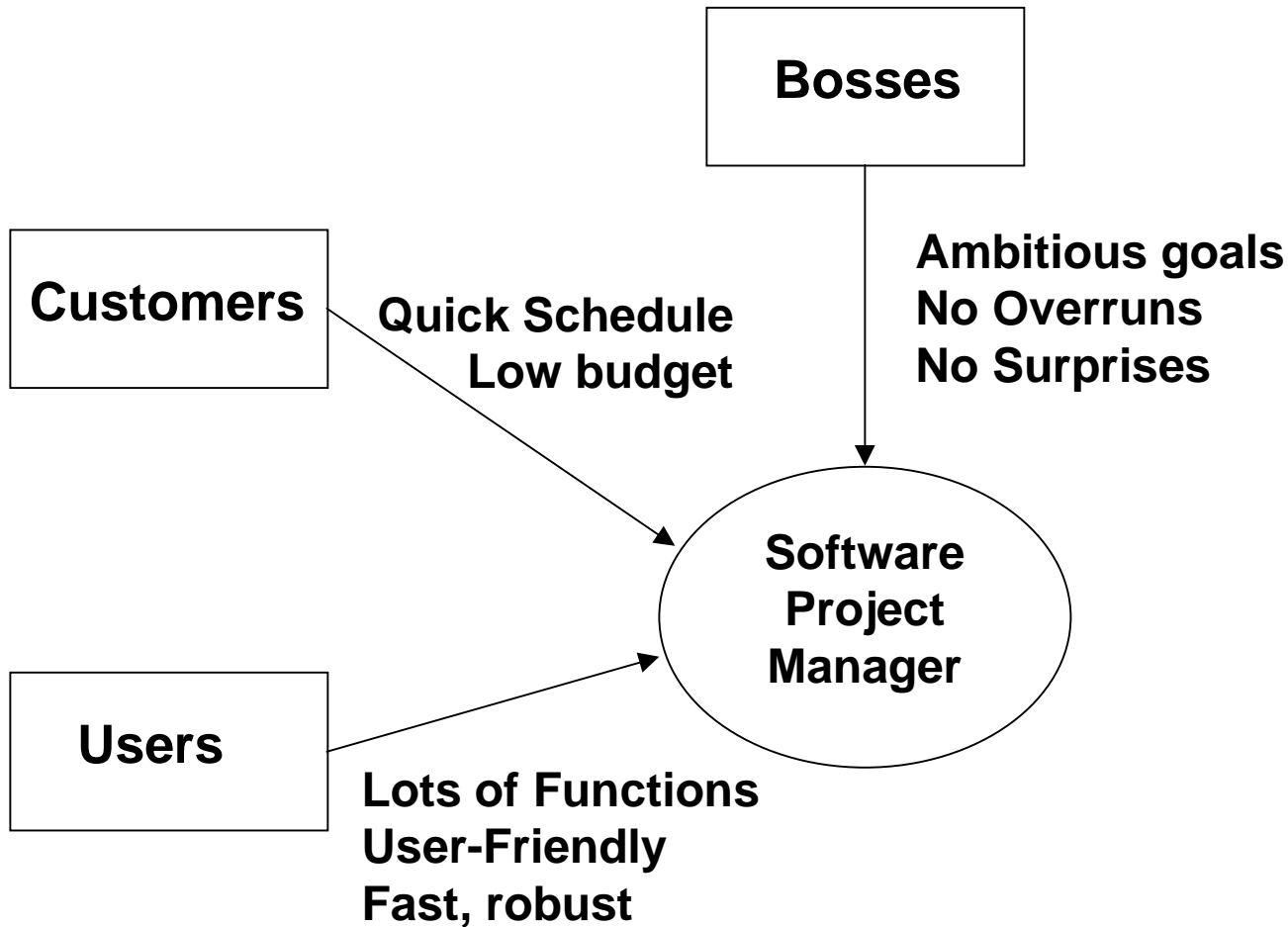


# Outline

- **The Problem**
  - For the software project manager
  - For software management theories
- **Approaches to date**
- **Theory W Principles and Practices**
- **Theory W Research Issues**
- **Conclusions**

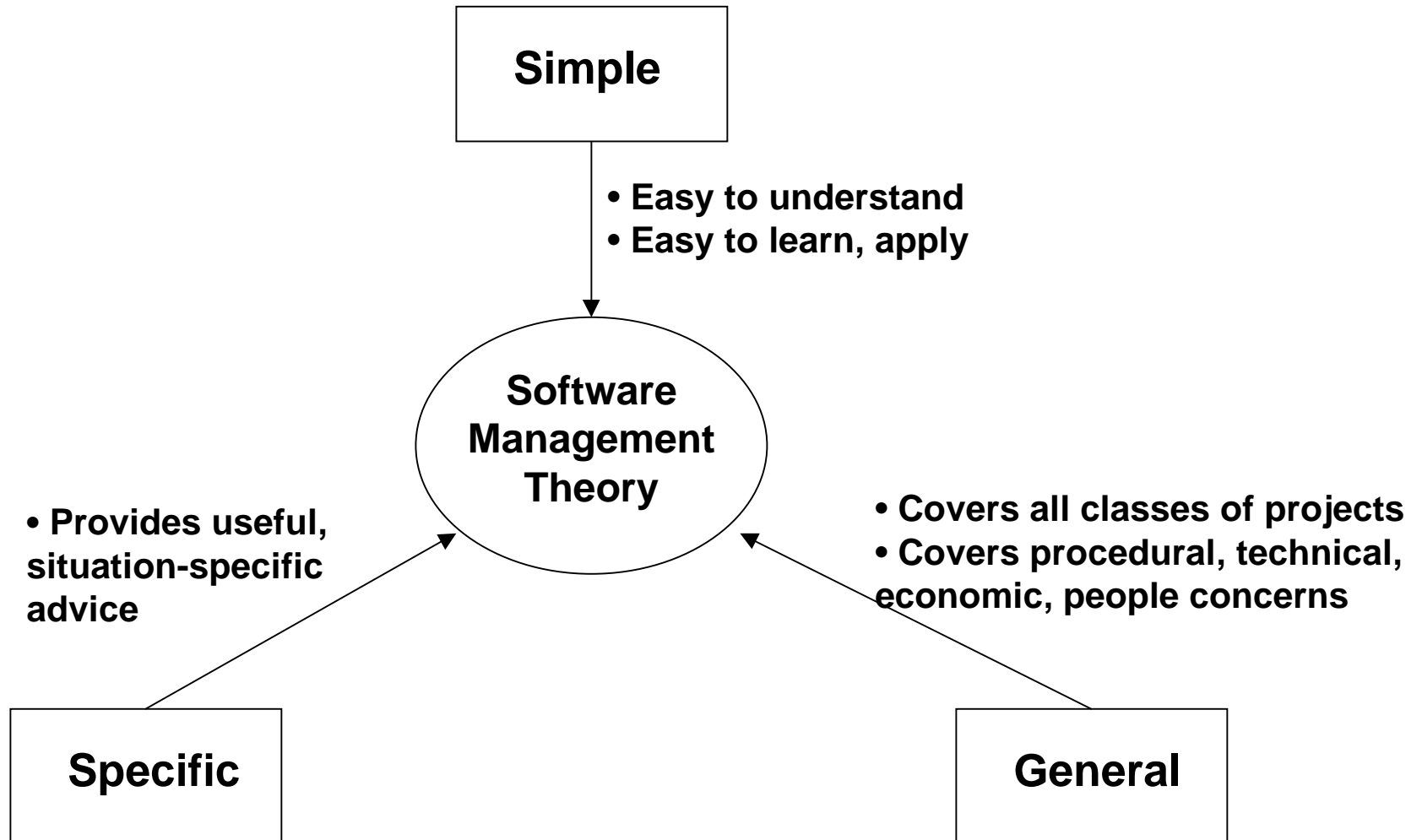


# The Software Project Manager's Problem





# The Software Management Theory Problem





# Approaches to Date

- **Objectives: Simple, General, Specific**
- **Alternatives**
  - Eclectic combinations of advice
  - DoD-STDS, Procedural Guidebooks
  - Koontz-O'Donnell Elaborations
  - One-minute manager, et al.
  - Theories X, Y, Z
  - Theory W: Make everyone a winner

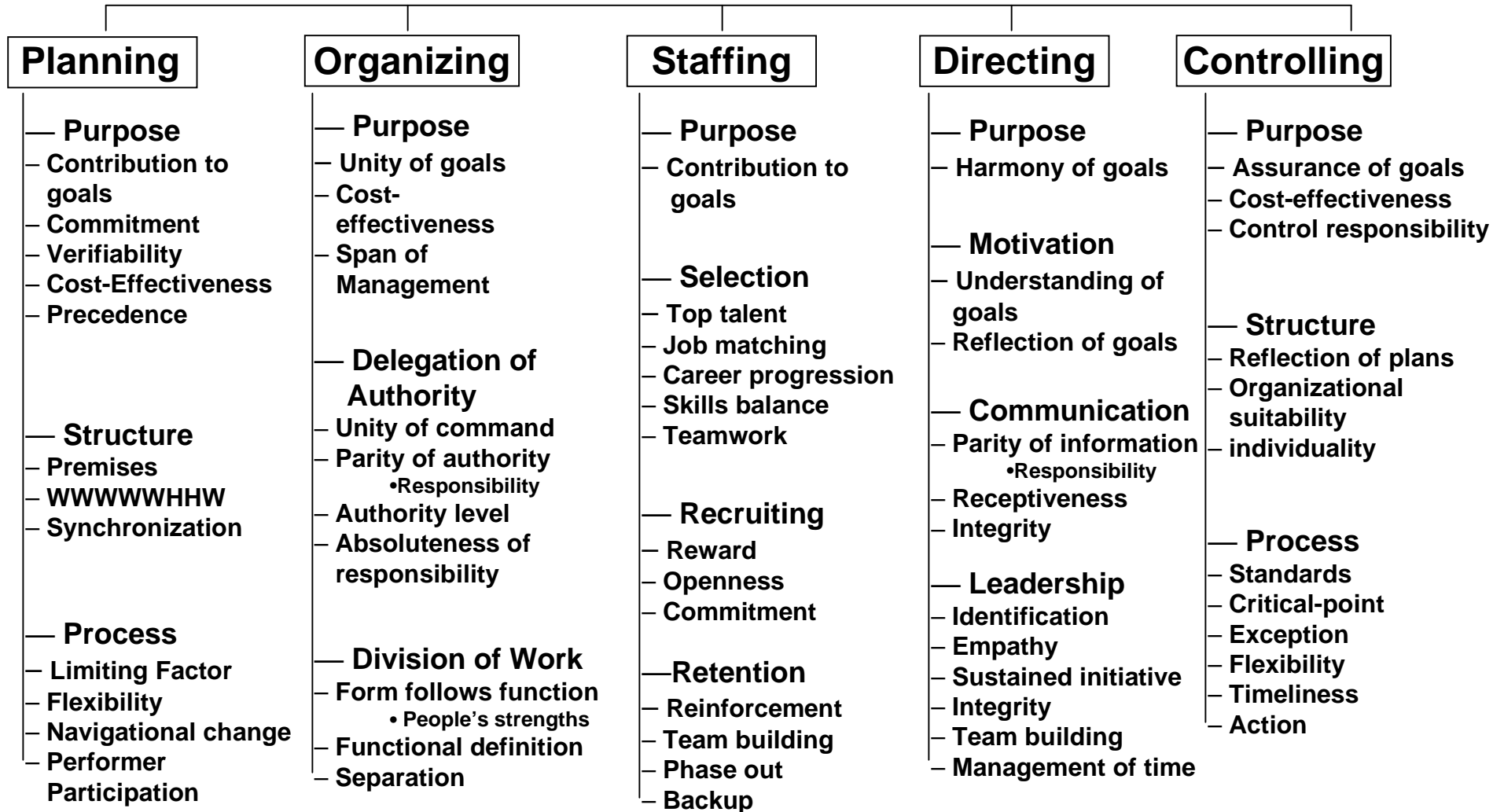


# Sorting out software advice

Do it top-down  
Thorough test planning  
Build It twice  
Use disciplined reviews  
Prove everything correct  
Do it outside-in  
Programming standards  
Independent test teams  
Use walk-throughs  
Chief Programmer teams  
Early requirements baseline  
Measurable milestones  
Program Library  
Involve the user  
Design verification  
Structured Programming  
Configuration management  
Project work authorizations  
Automated aids  
Unit development folders  
End-item acceptance plan



# Koontz-O'Donnell Management Framework





# Theory X and Theory Y\*

- **Theory X**
  - People inherently dislike work
  - They have to be coerced into working
  - They prefer being told what to do
- **Theory Y**
  - People don't inherently dislike work
  - People can exercise self-direction
  - Commitment to objectives depends on resulting rewards
  - People can learn to seek responsibility
  - Work creativity is widely distributed
  - People's potential is only partially utilized

\* D. McGregor, *The Human Side of Enterprise*, 1960.



## **Theory Z: Japanese-Style Management**

- **People work best toward goals which they have helped establish**
- **Once people have bought into goals, you can trust them to perform**
- **If people share a common set of values, they can develop workable project goals**



# Theory W Software Management Steps

- **Establish a set of win-win preconditions**
  - Understand how people want to win
  - Establish a set of win-win objectives
    - Establish reasonable expectations
    - Match people's objectives to their win conditions
  - Provide a supportive environment
- **Structure a win-win software process**
  - Establish a realistic process plan
    - Highlight potential win-lose, lose-lose risk items
  - Keep people involved
  - Provide feedback
    - Confront, resolve new win-lose, lose-lose situations
- **Structure a win-win software product**
  - Match product to users', maintainers' win conditions



# Theories X, Y, Z, and W: An Example

- **Problem**
  - **George and Harry want same system analysis job**
  - **Both well-qualified deserving**
- **Solutions**
  - **Theory X: Boss makes arbitrary choice**
  - **Theory Y: Boss asks for proposals, picks most ambitious one**
  - **Theory Z: Boss pre-builds consensus on team objectives, chooses based on Qualifications rating**



# Theory W Solution to Problem

- **Understand how people want to win**
  - **George: career path to marketing**
  - **Harry: Extensive travel to Boston; Daughter in college there**
- **Establish a set of win-win objectives by realigning expectations or expanding option space**
  - **Find comparable marketing-oriented job for George**
  - **Find comparable job with Boston travel for Harry**



# Strategic Guidelines Derived from Win-Win Preconditions

Win-Win Precondition	Users	Maintainers	Customers	Developer Team
Understand win conditions	<ul style="list-style-type: none"> <li>• Mission Analysis</li> <li>• Operations concept</li> <li>• Prototyping</li> <li>• Requirements spec</li> <li>• Early users' manual</li> </ul>	<ul style="list-style-type: none"> <li>• Operations concept</li> <li>• Operations procedures</li> </ul>	<ul style="list-style-type: none"> <li>• Cost-benefit analysis</li> </ul>	<ul style="list-style-type: none"> <li>• Career path development</li> </ul>
Reasonable expectations	<ul style="list-style-type: none"> <li>• Requirements scrub</li> </ul>	<ul style="list-style-type: none"> <li>• Team building, negotiating, communicating</li> <li>• Resource allocation</li> </ul>		
Match tasks to win conditions	<ul style="list-style-type: none"> <li>• User-spec reviews</li> <li>• Prototype exercise</li> </ul>	<ul style="list-style-type: none"> <li>• Quality assurance</li> </ul>	<ul style="list-style-type: none"> <li>• Change control participation</li> <li>• Status tracking</li> <li>• Early error detection</li> </ul>	<ul style="list-style-type: none"> <li>• Staffing, organizing</li> </ul>
Supportive environment preparation	<ul style="list-style-type: none"> <li>• User training</li> <li>• Cutover preparation</li> </ul>	<ul style="list-style-type: none"> <li>• Maintenance training</li> <li>• Conversion</li> <li>• Deliverable support environment</li> <li>• Configuration management</li> </ul>	<ul style="list-style-type: none"> <li>• Customer training</li> <li>• Modern programming practices</li> </ul>	<ul style="list-style-type: none"> <li>• Developer training</li> <li>• Support environment</li> <li>• Configuration management</li> </ul>



# Strategic Guidelines Derived from Product, Process Guidelines

Guideline	Users	Maintainers	Customers	Development Team
<b>Process Planning</b>	<ul style="list-style-type: none"> <li>•Operational plan</li> <li>•Installation and training plans</li> </ul>	<ul style="list-style-type: none"> <li>•Life-cycle support plan</li> </ul>	<ul style="list-style-type: none"> <li>•Development plans</li> <li>•Risk management plans</li> </ul>	
<b>Process Involvement</b>	<ul style="list-style-type: none"> <li>•System engineering plan participation</li> <li>•Review participation</li> <li>•Prototype exercise</li> </ul>	<ul style="list-style-type: none"> <li>•System engineering plan participation</li> <li>•Review participation</li> <li>•Quality assurance</li> </ul>	<ul style="list-style-type: none"> <li>•Cost-benefit reviews, approvals</li> </ul>	<ul style="list-style-type: none"> <li>•Delegation</li> <li>•Planning participation</li> </ul>
<b>Process Feedback</b>	<ul style="list-style-type: none"> <li>•Reviews</li> </ul>	<ul style="list-style-type: none"> <li>•Team building, negotiating, communicating</li> <li>•Reviews</li> <li>•Status tracking, controlling</li> <li>•Performance feedback</li> </ul>		
<b>Product Structuring</b>	<ul style="list-style-type: none"> <li>•Service-oriented</li> <li>•Efficient</li> <li>•Easy to learn</li> <li>•Easy to use</li> <li>•Tailorable</li> </ul>	<ul style="list-style-type: none"> <li>•Easy to modify</li> <li>•Programming standards</li> </ul>	<ul style="list-style-type: none"> <li>•Efficient</li> <li>•Correct</li> <li>•Feasible</li> </ul>	<ul style="list-style-type: none"> <li>•Easy to modify</li> <li>•Balanced</li> <li>•Correct</li> </ul>



# Outline

- **The Problem**
  - For the software project manager
  - For software management theories
- **Approaches to date**
- ➔ • **Theory W Principles and Practices**
- **Theory W Research Issues**
- **Conclusions**

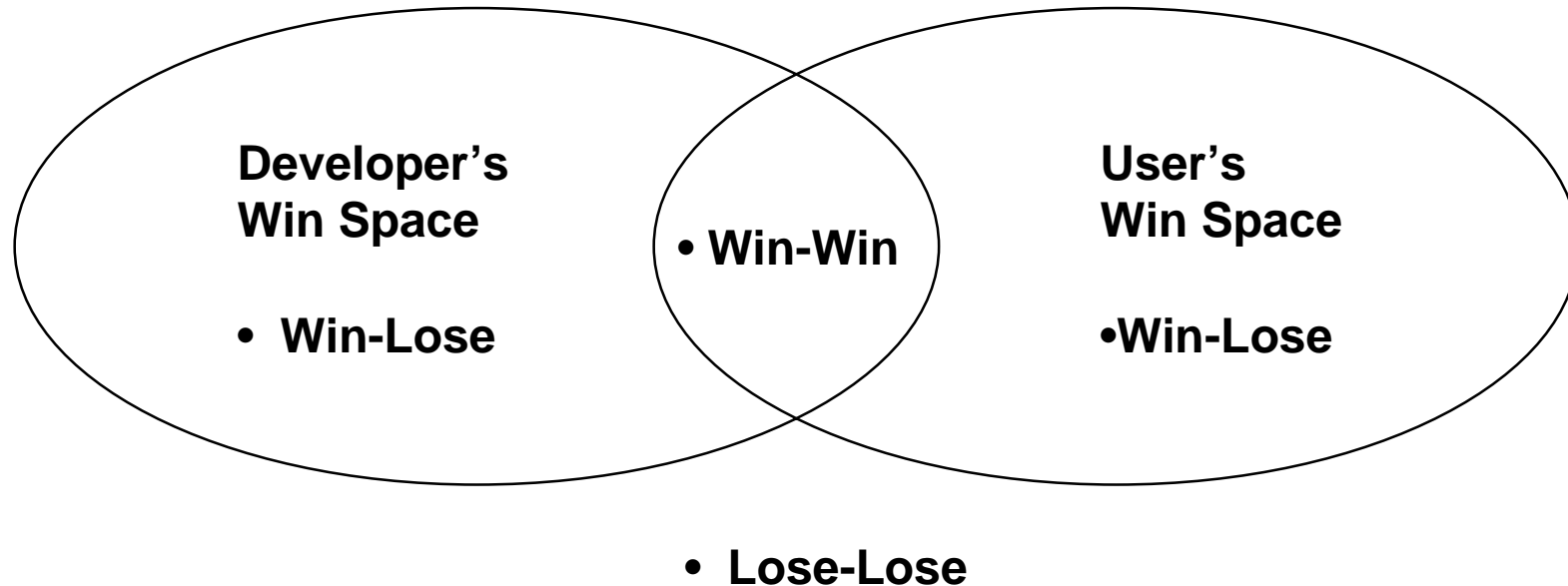


# Theory W Principles and Practices

- **Principles**
  - Win-win, win-lose, and lose-lose situations
  - **Getting to win-win**
    - Getting to yes: Principles of negotiations
- **Practices: Examples**
  - Understanding win conditions
  - Establishing win-win objectives
  - Structuring win-win software process
  - Structuring win-win software products

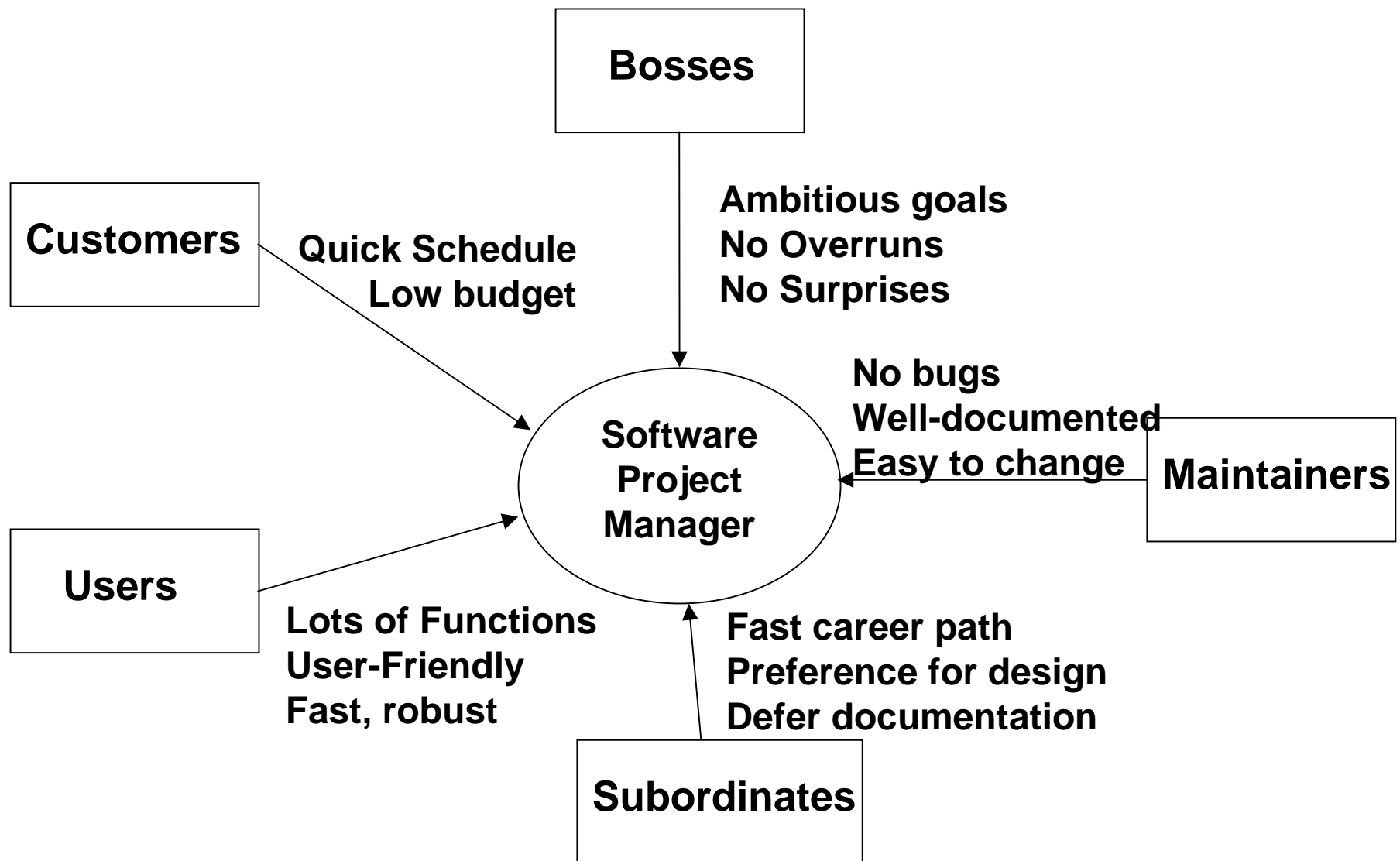


# Win-Win, Win-Lose, and Lose-Lose Situations





# The Software Project Manager's Problem





# Making Everyone a Winner: Potential Conflicts

Proposed Solution	“Winner”	Loser
Quick, Cheap, Sloppy Product	Developer & Customer	User
Lots of bells and whistles	Developer & User	Customer
Driving too hard a bargain	Customer & User	Developer

**Actually, nobody wins in these situations**



# Negotiation Principles\*

- **Don't bargain over positions**
- **Use 4-step solution approach**
  - **Separate the people from the problem**
  - **Focus on interests, not positions**
  - **Invent options for mutual gain**
  - **Insist on using objective criteria**

\* Fisher & Ury, *Getting to Yes*, 1981.



# Theory W Principles and Practices

- **Principles**

- Win-win, win-lose, and lose-lose situations
- Getting to win-win
  - Getting to yes: Principles of negotiations

- ➔ • **Practices: Examples**

- Understanding win conditions
- Establishing win-win objectives
- Structuring win-win software process
- Structuring win-win software products



# Understanding People's Win Conditions

- **Principles**
  - People in general
  - Software people
- **Practices**
  - Reaching out
  - Studying the culture
  - Projection
  - Mutual Exploration



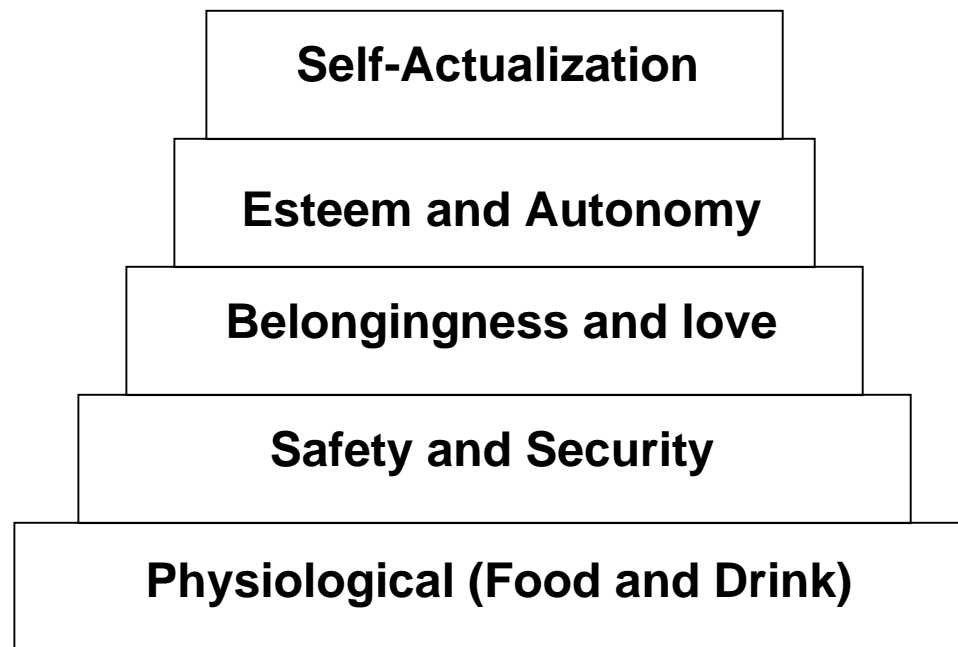
# Win Conditions: People in General

- **Maslow Need Hierarchy**
- **Motivating Factors**
- **Theories X, Y, Z, and W**



# Maslow Human Need Hierarchy

A. Maslow, *Motivation and Personality*, 1954.





# Maslow Need Hierarchy

- **Satisfied needs aren't motivators**
- **Unsatisfied lower-level needs dominate higher-level needs**
- **Management implications**
  - **Create environment and subculture which satisfies lower-level needs**
    - **Stability**
    - **Shared values, community**
    - **Match to special needs**
  - **Tailor project objectives, structure to participants' self-actualization priorities**



# People Self-Actualize in Different Ways

- **Becoming a Better Manager**
- **Becoming a Better Technologist**
- **Helping Other Developers**
- **Helping Users**
- **Making People Happy**
- **Making People Unhappy**
- **Doing New Things**
- **Increasing Professional Stature**

**Project Managers Must be Very Sensitive to these Differences-  
Remember the Modified Golden Rule**



# Win Conditions: Software People

- **Overall motivating factors**
- **Growth needs vs. social needs**
- **Responsiveness to objectives**
- **Some management implications**



# Ranking of Motivating Factors

## General (Herzberg)

1. Achievement
2. Recognition
3. Work Itself
4. Responsibility
5. Advancement
6. Salary
7. Possibility for growth
8. Relations, subordinate
9. Status
10. Relations, superior



# Ranking of Motivating Factors

## General (Herzberg)

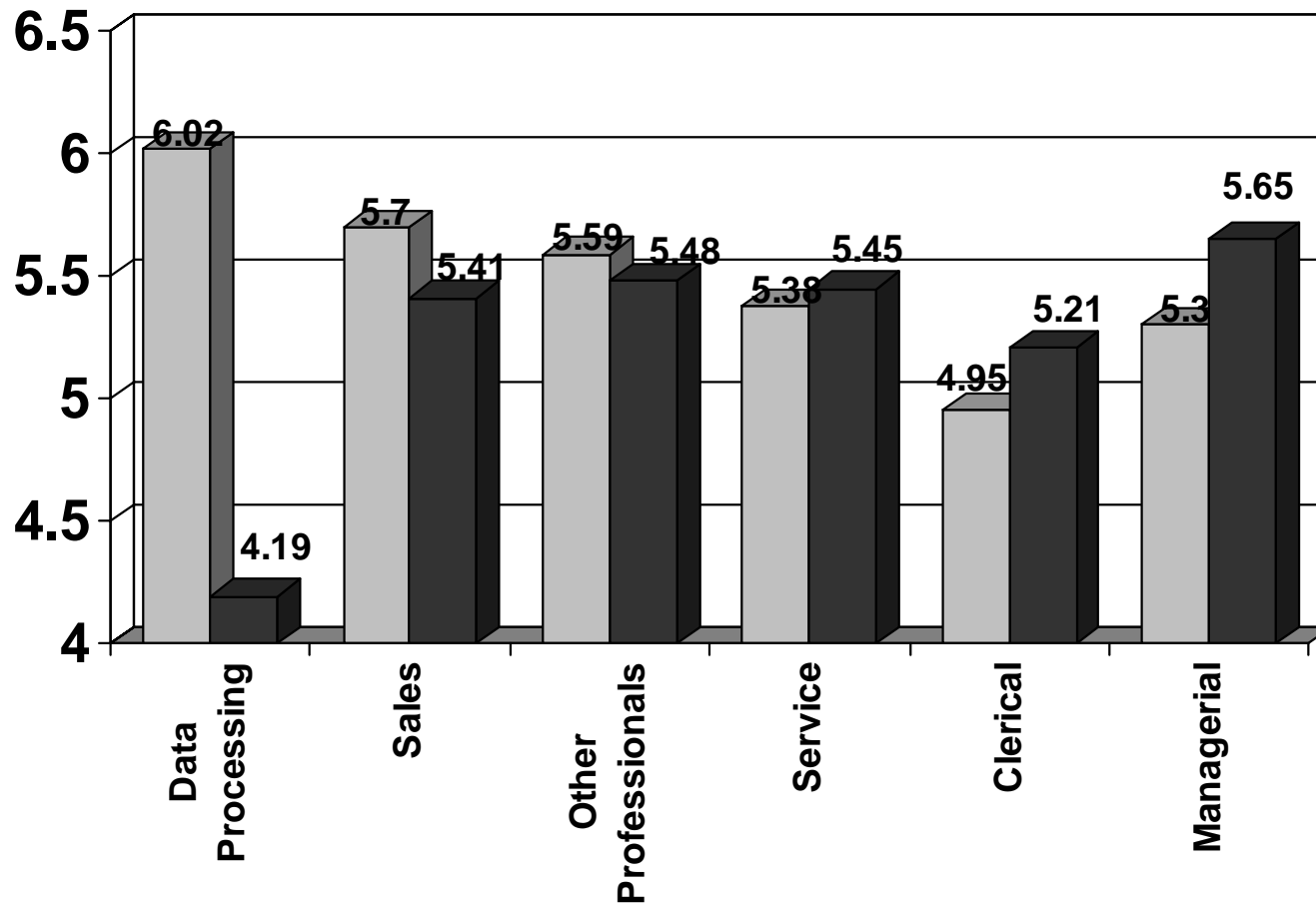
1. Achievement
2. Recognition →
3. Work Itself
4. Responsibility →
5. Advancement →
6. Salary →
7. Possibility for growth
8. Relations, subordinate
9. Status
10. Relations, superior ↘<sup>14</sup>

## DP Professionals (Fitz-Enz)

1. Achievement
- ↗ 2. Possibility for growth
3. Work Itself
4. Recognition
5. Advancement
- ↗ 6. Tech. Supervision
- ↗ 7. Responsibility
- ↗ 8. Relations, peers
- ↗ 9. Relations, subordinate
10. Salary



# Comparative Growth Needs and Social Needs





## Experiments Show that Programming Team Performance is Highly Sensitive to Given Objectives\*

Team Objective: Optimize	Resulting Rank on Performance**				
	Time To Complete	No. of Statements	Memory Required	Program Clarity	Output Clarity
Time To Complete	1	4	4	5	3
No. of Statements	2-3	1	2	3	5
Memory Required	5	2	1	4	4
Program Clarity	4	3	3	1-2	2
Output Clarity	2-3	5	5	1-2	1

\*Weinberg, 1972

\*\* 1=Best



# Effect of Objectives on Productivity

## (Weinberg-Schulman, 1974)

<b>Team Objective: Optimize</b>	<b>Number of Statements</b>	<b>Man- hours</b>	<b>Productivity (State M-H)</b>
<b>Core Memory</b>	<b>52</b>	<b>74</b>	<b>0.7</b>
<b>Number of Statements</b>	<b>33</b>	<b>30</b>	<b>1.1</b>
<b>Execution Time</b>	<b>100</b>	<b>50</b>	<b>2.0</b>
<b>Program Clarity</b>	<b>90</b>	<b>40</b>	<b>2.2</b>
<b>Programming, Man-hours</b>	<b>126</b>	<b>28</b>	<b>4.5</b>
<b>Output Clarity</b>	<b>166</b>	<b>30</b>	<b>5.5</b>



# **Incorporating People's Goals in Management Decisions: DP People**

- **Concentrate on Meaningful Work, Growth Opportunities**
  - **Carefully Define Objectives, Priorities**
- **Downplay Status as a Primary Motivator**
  - **Watch Out for Peter Principle**
- **Try to Develop Responsibility**
  - **Participation in Planning, Goal-Setting**
- **Increase Feedback**
- **Remember the Modified Golden Rule**



# Understanding People's Win Conditions

- **Principles**
  - People in general
  - Software people

- • **Practices**
  - Reaching out
  - Studying the culture
  - Projection
  - Mutual Exploration



# Reaching Out

- **Interviews, conversations**
- **Surveys, questionnaires**
- **Tours of duty**
- **Hypothesis testing**
  - **Prototypes, scenarios**
  - **OPS-concept document**
  - **Draft users' manuals**



# Studying the Culture

- **Background reading**
  - User shared values, taboos
- **Operations analysis**
  - Example: Accounting for funds, man-hours
- **Previous experiences with automation**
  - Scars and bruises
  - Previous winners



# Projecting Yourself Into Others' Win Situations

## Counterexample: The Golden Rule

- **Do unto others**
  - .. **As you would have others do unto you**
- **Build computer systems to serve users and operators**
  - .. **Assuming users and operators like to write programs, and know computer science**
- **Computer sciences world (compilers, OS, etc.)**
  - **Users are programmers**
- **Applications world**
  - **Users are pilots, doctors, tellers**



# The Modified Golden Rule

**Do unto others  
as you would have others do unto you  
– if you were like them**



# Mutual Exploration

- **Users:** How can software technology help them become more effective?
  - Prototypes, demonstrations
- **Owners:** How can the software product enhance their value to the ongoing mission?
  - Ease of change; diagnostics
- **Subordinates:** How can the project help them achieve career goals?
  - Training, breadth of experience
- **General:** Helping people find out and demonstrate they are winners



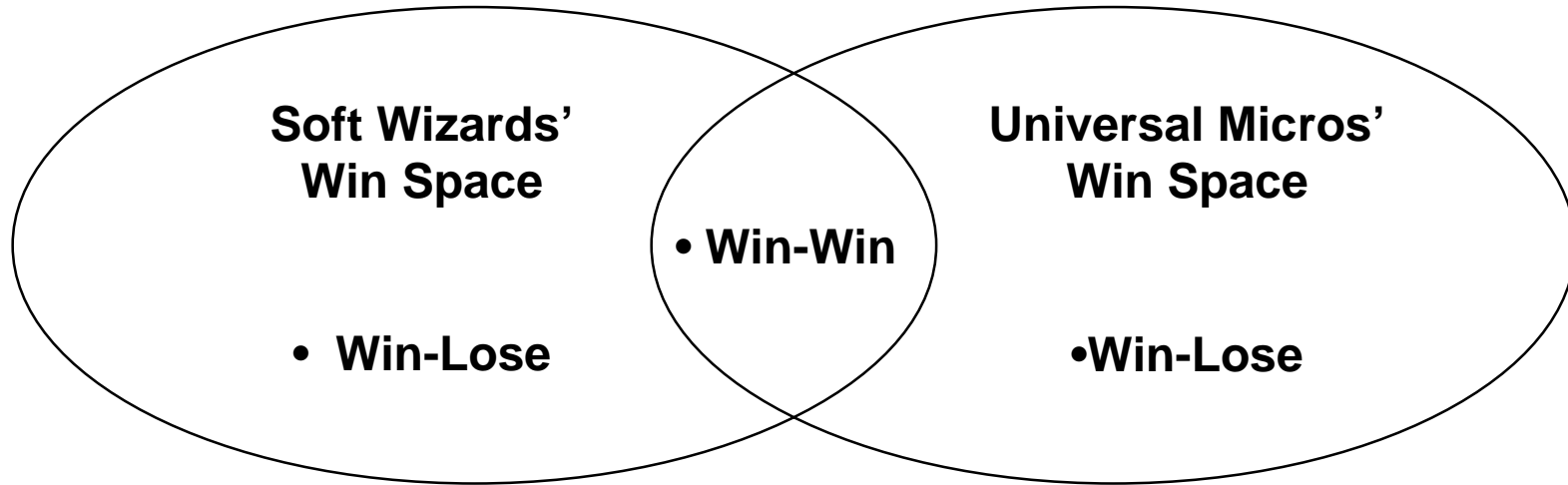
# Reconciling People's Win Conditions

- **Principles**
  - Win-Win, Win-Lose, and Lose-Lose situations
  - Negotiation principles
- **Practices**
  - Searching out Win-Win situations
  - Expanding the option space
  - Teambuilding and shared values
  - Setting expectations



# Win-Win, Win-Lose, and Lose-Lose Situations

- Lose-Lose



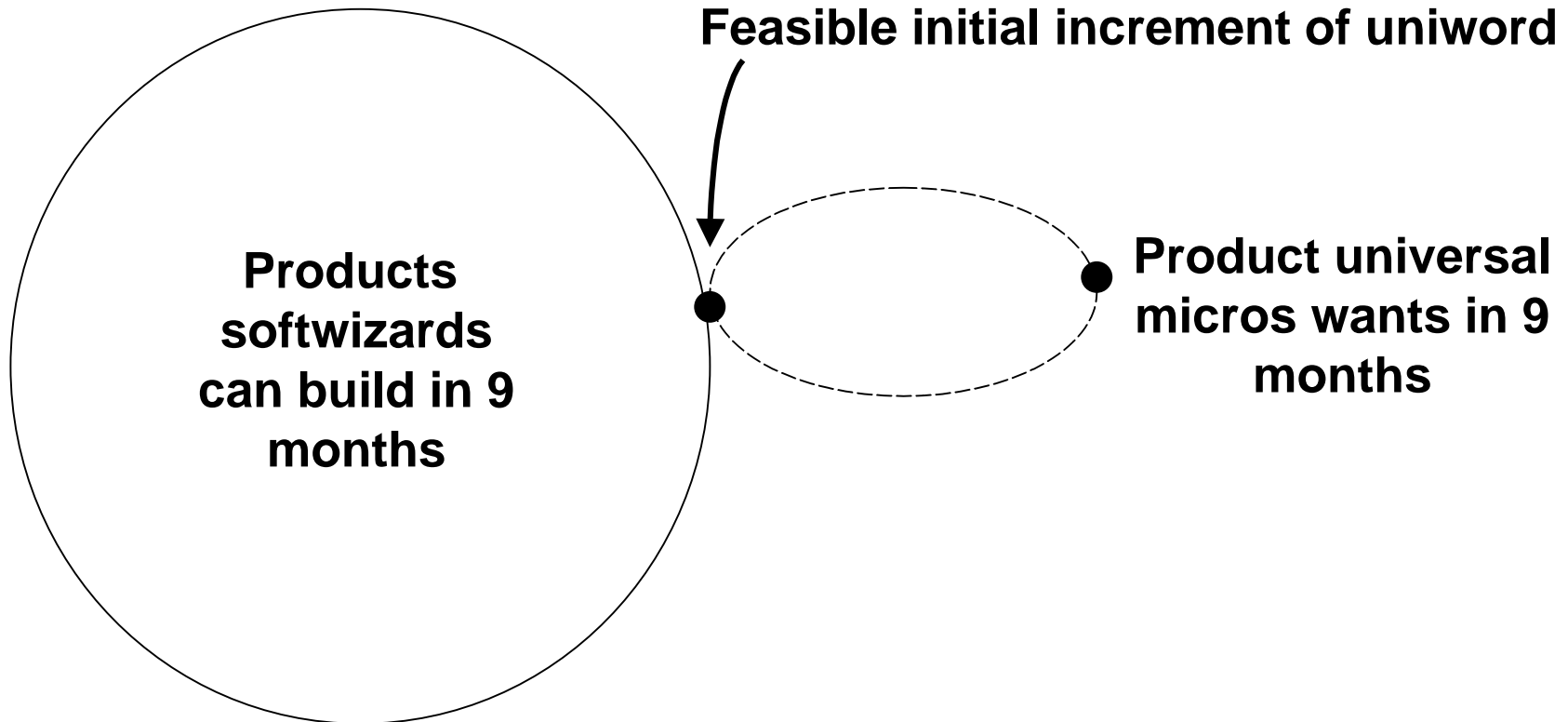


# Win-Win, Win-Lose, & Lose-Lose Examples: Uniword

- **Win-Win**
  - License fee for soft wizards
  - Structured programming
- **Win-Lose**
  - Best and final offer
  - Independent user-interface designs
  - Gold-plated DBMS
- **Lose-Lose**
  - Establishing unrealistic schedule
  - Staffing with incompatible people
  - Poor planning
  - Adding people to catch up
  - No concurrence on product features



# Getting to Win-Win





# Negotiation Principles

- Fisher & Ury, “*Getting to Yes*,” 1981
- Don’t bargain over positions
- Use 4-step solution approach
  - Separate the people from the problem
  - Focus on interests, not positions
  - Invent options for mutual gain
  - insist on using objective criteria



# Separate the People From the Problem

- **Put yourself in their shoes**
- **Recognize and understand emotions**
- **Present proposals in terms of their values**
- **Make sure they participate in the process**
- **Face the problem, not the people**



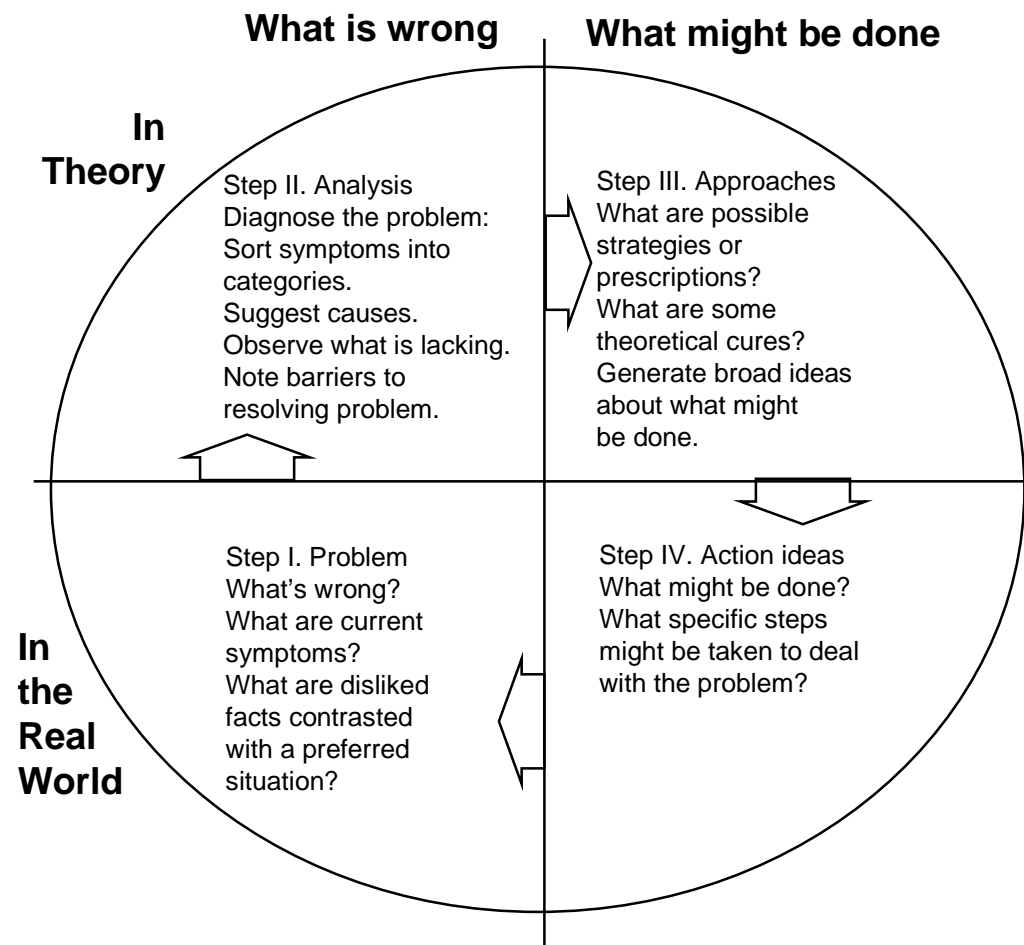
# Focus on Interests, Not Positions

- **Ask “Why?”**
- **Ask “Why not?”**
- **Look forward, not back**
- **Be concrete but flexible**
- **Be hard on the problem, soft on the people**



# Inventing Options for Mutual Gain

- The four basic steps: Fisher and Ury





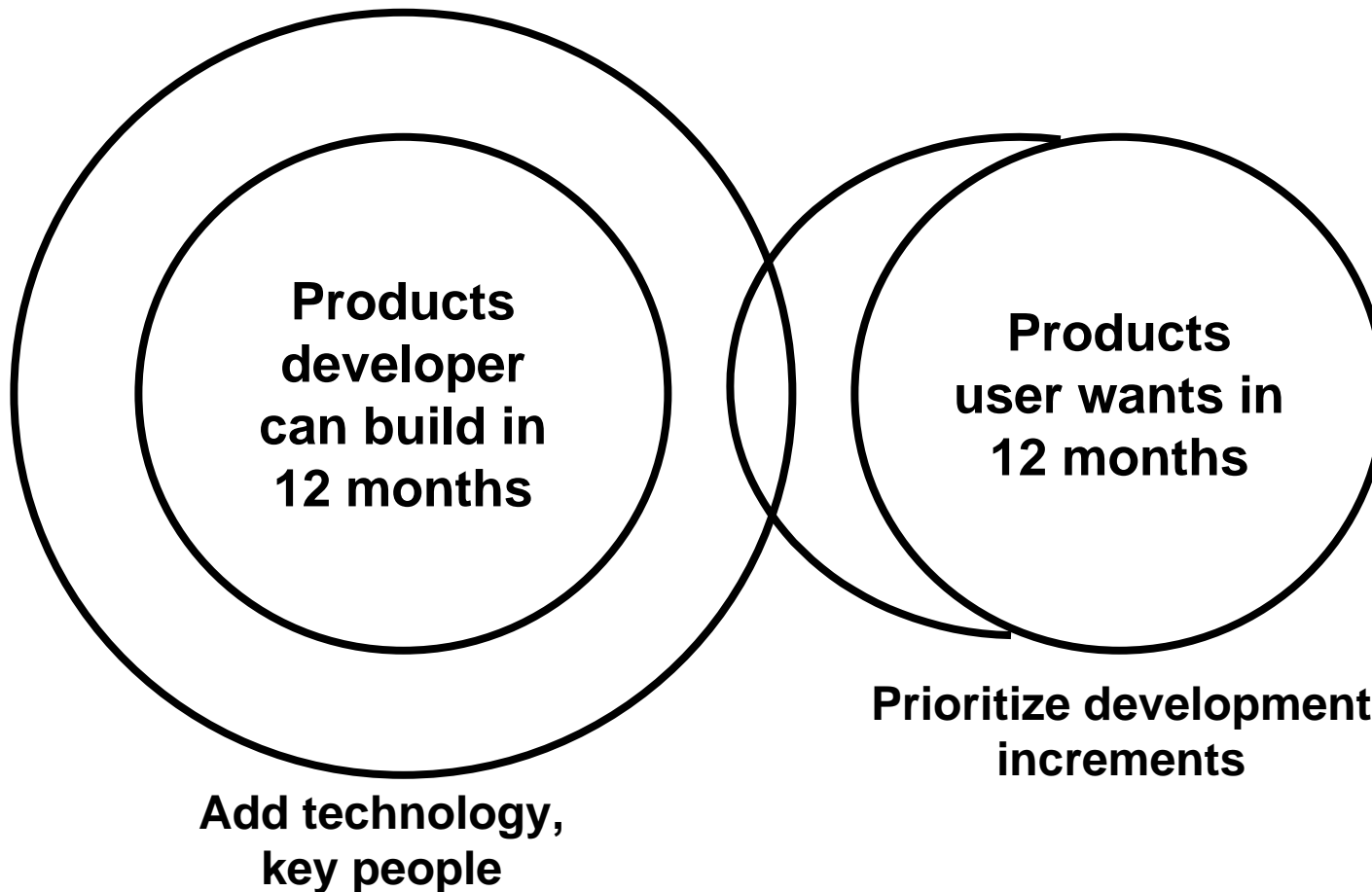
# Getting to Win-Win: COCOMO F-16 Example

**Products  
developer  
can build in  
12 months**

**Products  
user wants in  
12 months**



# Getting to Win-Win: COCOMO F-16 Example





# Insist on Using Objective Criteria

- **Fair standards**
- **Fair procedures**
- **Establish joint search for criteria**
- **Don't yield to pressure**
  - **Develop best alternative to negotiated agreement**



# Searching out Win-Win Situations

- **Breaking options into parts**
  - **Functionality**
    - **A take lead on user I/F; B on comm. Proc.**
  - **Increments**
  - **Phases**
- **Realigning options**
  - **OS, DBMS, applications**
  - **Input, process, output**
  - **Inventory, production, distribution**



# Expanding the Option Space

- **Linking to future options, career paths**
- **Linking to extra rewards**
- **Providing extra support**
- **Surfacing new technical options**
- **Creating ownership**
  - **Can be easily overdone, though**



# **Incorporating People's Goals in Management Decisions: Users**

- **Give Users Opportunities for Achievement, Responsibility**
  - Swedish Bank
- **Minimize User Difficulties With Product**
  - Help Messages
  - Avoid Lock-Step Controls
- **Don't Assume Users Have Urge to be Computer Scientists**
  - Data Entry Language
- **Remember the Modified Golden Rule**



# Teambuilding

- **Build appreciation for others' win conditions**
- **Establish shared values**
- **Group planning, issue resolution**
- **Offsites**



# Setting Up Reasonable Expectations

- **User: functionality**
- **Customer: budget, schedule**
- **Performer: Lead design role**

**Research content**

**Better to establish low expectations and come up  
than to establish high expectations and come down.**



# Theory W: A Large-Project Example

- **Scope of Phase I contract**
  - Develop Ada object-oriented design approach
  - Use on representative system CSCI
  - Demonstrate requirements satisfaction
    - Functions, portability, performance, reliability
  - Learn lessons; incorporate into Phase II development
- **Customer expectations**
  - No problems with Phase I
  - Phase I CSCI fully usable in Phase II
- **External PDR, CDR reviewer comments**
  - Design not object-oriented
  - Should consider PDR, CDR not passed
    - Would cause major slip in Phase I completion
- **Review team called in to assess situation, make recommendations**



# Review Team Procedures

- **Review team composition**
  - Key PDR, CDR external reviewers
  - Contractor non-project Ada experts
  - External Ada Experts
- **Review charter**
  - Determine if design is/isn't object-oriented?
  - Determine if PDR, CDR are/aren't passed?
  - Determine how to get best system design & plan
- **Output ground rules**
  - Full consensus; no minority reports
- **Initial activities**
  - Find out how people want to win
    - Customer, contractor, external reviewers
  - Establish reasonable expectations
    - Determine how well design satisfies rqts.
    - Identify risks
    - Reconcile with expectations



# Review Findings

- **Software rqts. not traceable to OPS-concept**
- **Design faithfully followed project OOD guidelines**
- **Design would have major problems in meeting full-scale performance & reliability rqts.**
- **Design would make significant classes of changes difficult**



# Review Recommendations

- **Consider Phase I CSCI a throwaway prototype**
  - A win for external reviewers
  - Revised expectations for customers
- **Congratulate customer for foresight in establishing a 2-phase, lessons-learned approach**
  - A win for customer
- **Consider PDR, CDR passed**
  - A win for customer and contractor
- **Establish risk management plan to address risk items identified**
  - A downstream win for customer, contractor, external reviewers, and users



# Structuring a Win-Win Software Process - FAA/AAS Risk Management Plan

## Users winners $\Rightarrow$

- **Reliability:** Use Ada; risk-manage exceptions, elaboration, RTS
- **Performance:** Risk-manage tasking, RTS, non-Ada use

## Customers winners $\Rightarrow$

- **Cost, schedule:** Risk-manage compiler limits; host, target support; developer readiness (exercise); key personnel

## Maintainers winners $\Rightarrow$

- Train maintenance personnel in Ada
- Have maintenance personnel develop maintenance plans
- Get maintenance personnel involved in development

## Developers winners $\Rightarrow$

- Re-evaluate fixed-price strategy
- Require developer risk management plan (RMP)
- Base selection on RMP, exercise as well as proposal



# Applying Win Conditions: Tactical Example

- **Situation** : Susan, a new 1st-level manager, proposes that project use a new test tool she worked on
- **Understanding win conditions:**
  - **Susan** : Help project avoid errors  
Justify time spent on tool
  - **Others** : New tool immature, adds risk  
Project considered, rejected similar tool
- **Establishing reasonable expectations:**
  - Meeting to explore tool use options, concerns
  - Others more sympathetic to tool's value
  - Susan more sympathetic to project risks



# Tactical Example - II

- **Matching tasks to win conditions**
  - Use tool experimentally on Susan's work package
  - Others review experience
  - If successful, use on entire project
- **Provide supportive environment**
  - Training on tool usage
  - Budget for tool improvement
- **Keeping people involved**
  - Periodic reviews positive: integration errors down 45%
  - Agreement to use on entire project
- **Providing feedback**
  - Bonus award for Susan, key subordinates
  - Division recognition of project contribution
  - Preparation for division-wide use of tool



# Structuring A Win-Win Software Product - Student COCOMO Programs

- **Requirement: Enter N input attributes**
- **Solution**
  - **Precondition:** 0 acceptable inputs
  - **Post-condition:** N acceptable inputs
  - **Invariant:** i acceptable inputs,  
add an acceptable input  
⇒ i+1 acceptable inputs
- **Program: Loop through input attributes**
- **User problem: Can't backtrack to fix previous inputs**



# Conclusions

- **Theory W addresses success criteria reasonably well**
- **Simple**
  - Expressible in 4 words, 8 steps
  - Detailed guidelines derivable from steps
- **General**
  - Applies to all classes of projects
  - Strongest on people issues, but also addresses procedural, technical, economic issues
- **Specific**
  - Provides specific solutions for both strategic and tactical management issues
  - Provides criterion for testing management solutions



# Theory W Management and Trust

- **Effective management is built on a bedrock of trust**
- **Practicing Theory W generates trust**
  - People see that you're looking out for their win conditions
- **Theory W is self-reinforcing**
  - If people know that as a manager you're going to consider other people's win conditions - they'll start thinking about them too