# Software Project Management

Session 4: WBS, Estimation & Scheduling

## Estimation

- “Predictions are hard, especially about the future”, Yogi Berra
- 2 Types: Lucky or Lousy?

## Planning, Estimating, Scheduling

- What’s the difference?
- Plan: Identify activities/work packages

  - first with No specific start and end dates
- Estimating: Determining the effort/size & duration of activities.
- Schedule: Adds specific start and end dates, relationships, and resources.

## Project Planning: A 12 Step Program

1) Set goal and scope
2) Select lifecycle
3) Set org./team form
4) Start team selection
5) Determine risks
6) Create WBS
7) Identify tasks
8) Estimate size
9) Estimate effort
10) Identify task dependencies
11) Assign resources
12) Schedule work

## How To Schedule

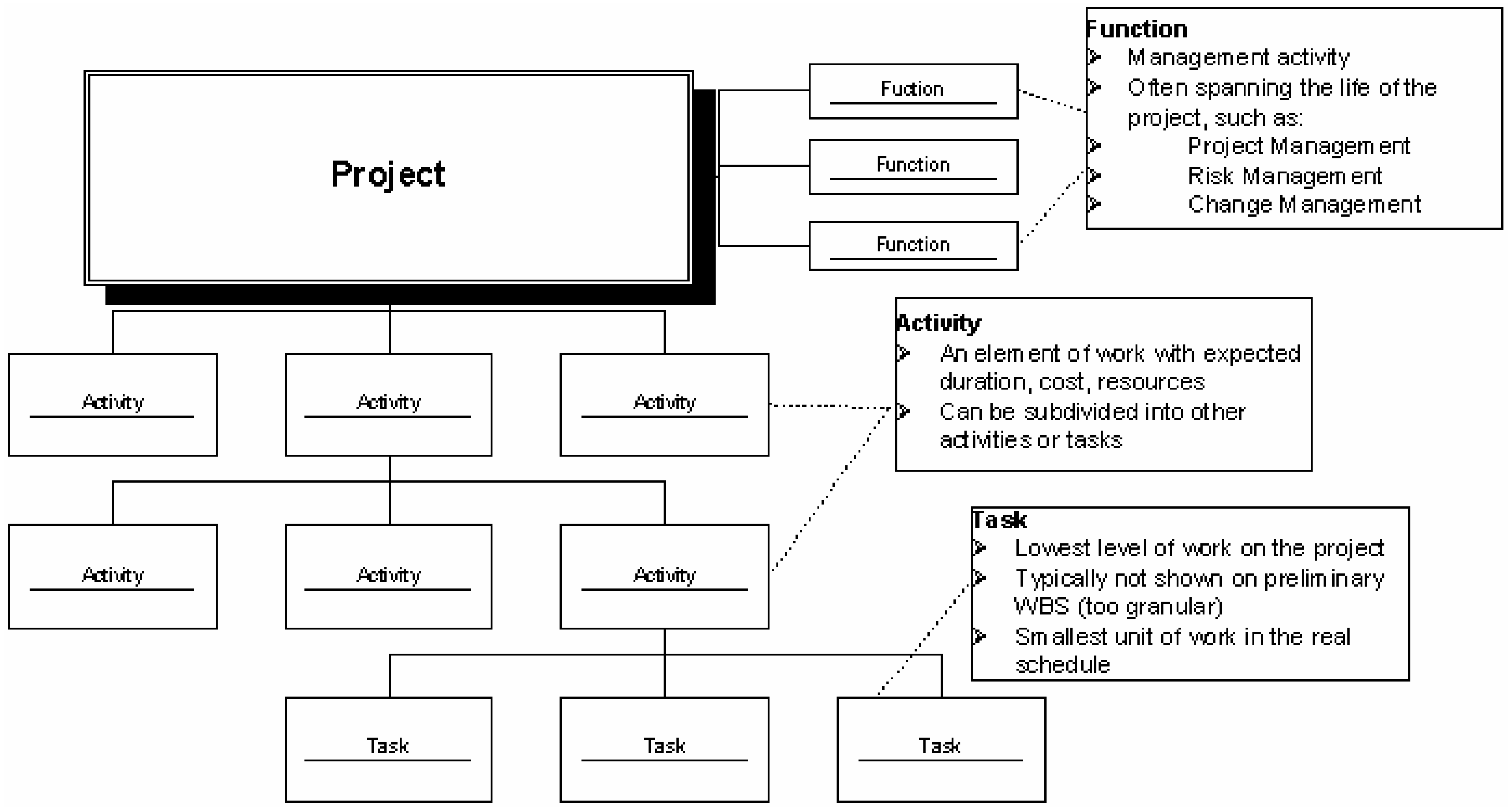
- 1. Identify “what” needs to be done
  - Work Breakdown Structure (WBS)
- 2. Identify “how much” (the size)
  - Size estimation techniques
- 3. Identify the dependency between tasks
  - Dependency graph, network diagram
- 4. Estimate total duration of the work to be done
  - The actual schedule

## Partitioning Your Project

- You need to decompose your project into manageable chunks
- ALL projects need this step
- Divide & Conquer
- Two main causes of project failure
  - Forgetting something critical
  - Ballpark estimates become targets
- How does partitioning help this?

# Project Elements

- A Project: functions/Processes, activities, tasks



**Function**
- Management activity
- Often spanning the life of the project, such as:
- Project Management
- Risk Management
- Change Management

**Activity**
- An element of work with expected duration, cost, resources
- Can be subdivided into other activities or tasks

**Task**
- Lowest level of work on the project
- Typically not shown on preliminary WBS (too granular)
- Smallest unit of work in the real schedule

# Work Breakdown Structure: WBS

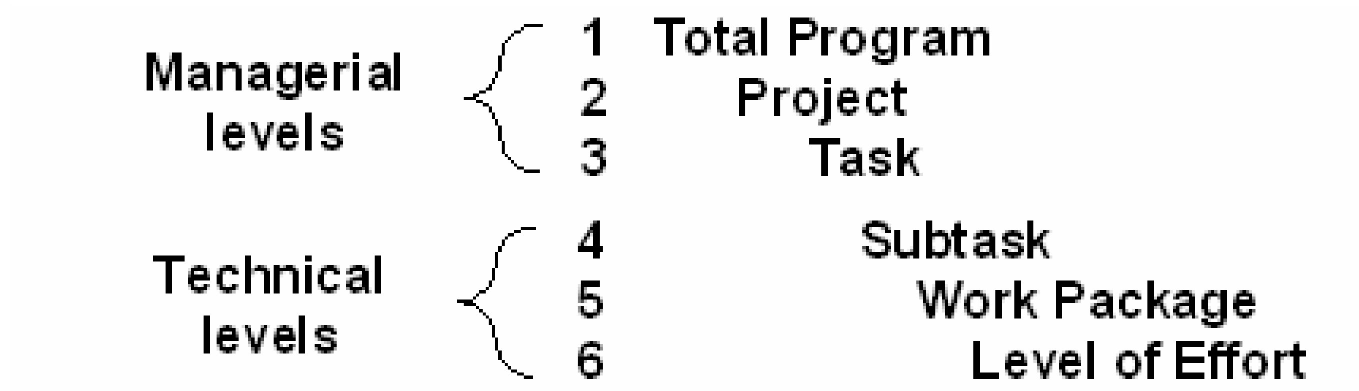
- Hierarchical list of project’s work activities
- 2 Formats
  - Outline (indented format)
  - Graphical Tree (Organizational Chart)
- Uses a decimal numbering system
  - Ex: 3.1.5
  - 0 is typically top level
- Includes
  - Development, Mgmt., and project support tasks
- Shows “is contained in” relationships
- Does not show dependencies or durations

# WBS

- Contract WBS (CWBS)
  - First 2 or 3 levels
  - High-level tracking
- Project WBS (PWBS)
  - Defined by PM and team members
  - Tasks tied to deliverables
  - Lowest level tracking

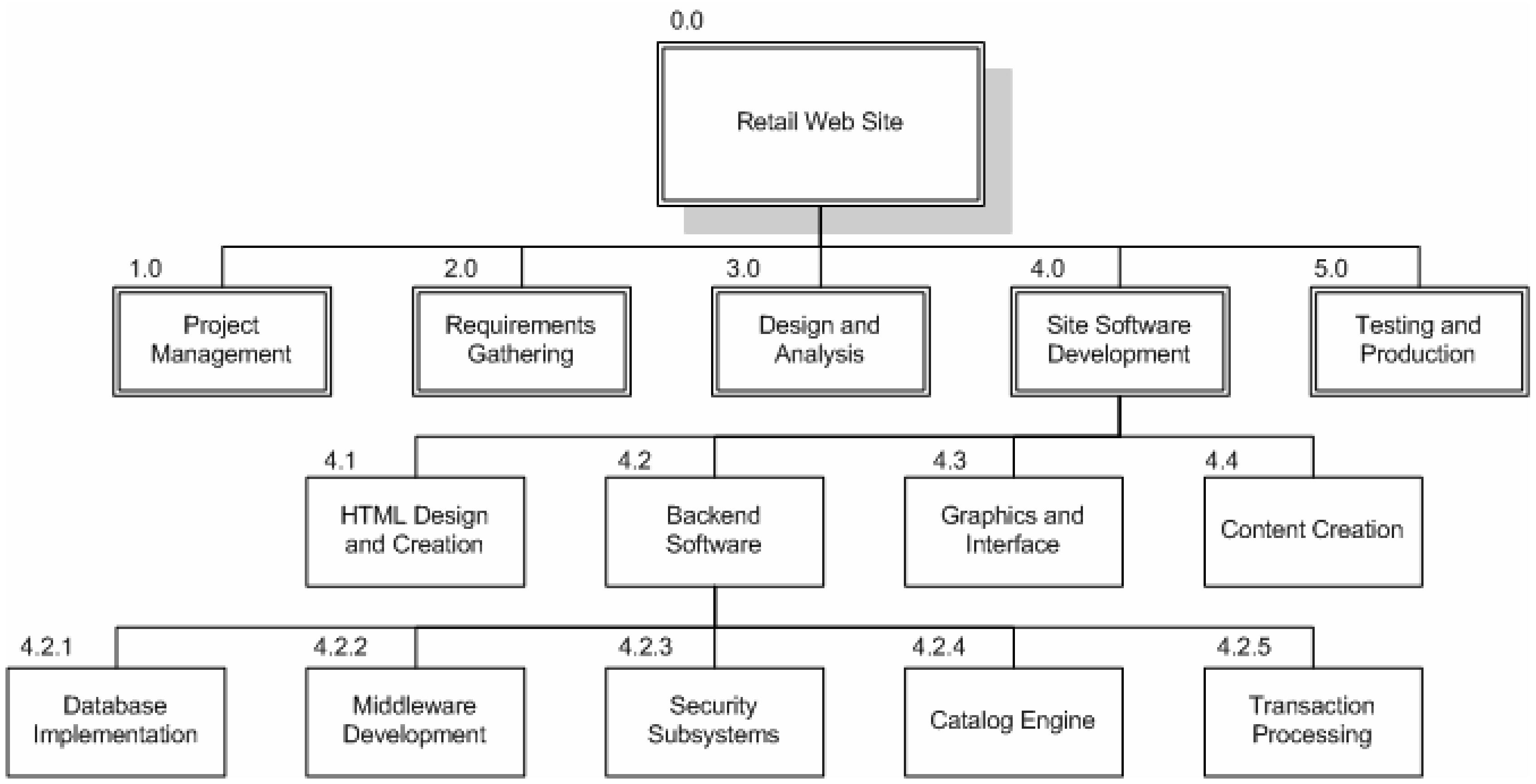
# A Full WBS Structure

- Up to six levels (3-6 usually) such as



- Upper 3 can be used by customer for reporting (if part of RFP/RFQ)
- Different level can be applied to different uses
  - Ex: Level 1: authorizations; 2: budgets; 3: schedules

# WBS Chart Example



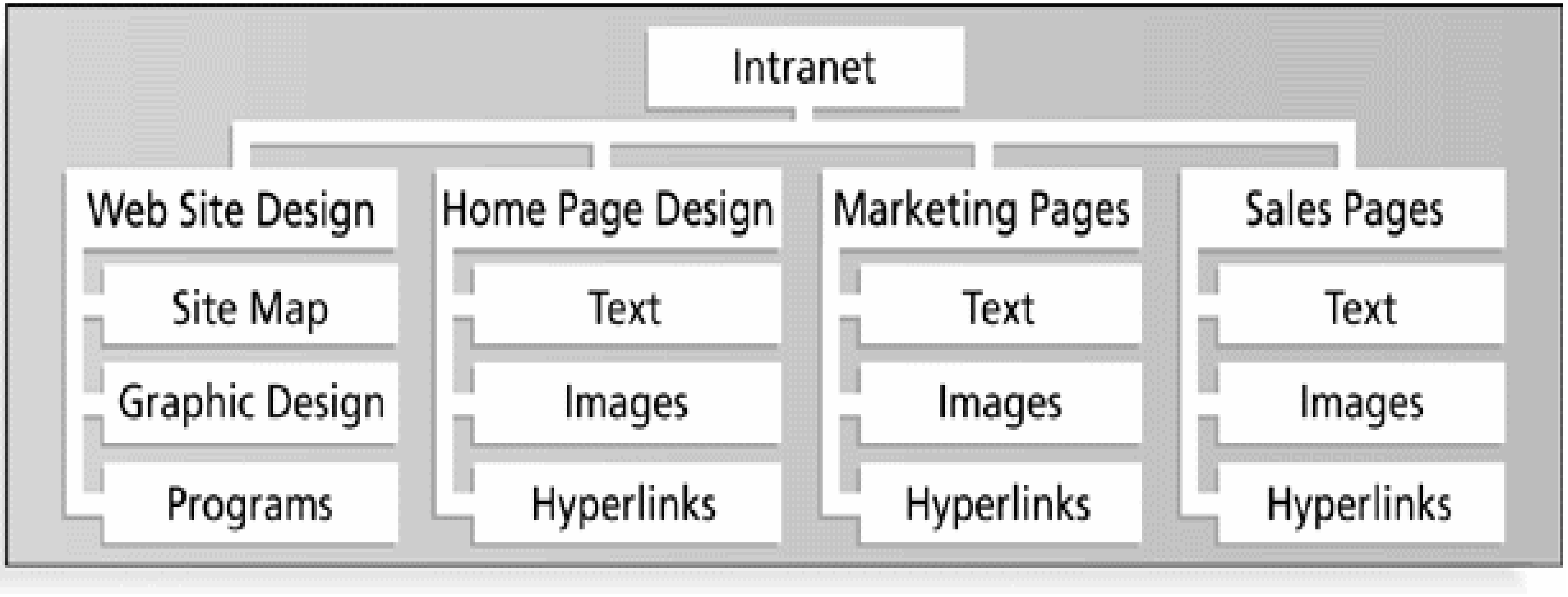
# WBS Outline Example

- 0.0 Retail Web Site
- 1.0 Project Management
- 2.0 Requirements Gathering
- 3.0 Analysis & Design
- 4.0 Site Software Development
  - 4.1 HTML Design and Creation
  - 4.2 Backend Software
    - 4.2.1 Database Implementation
    - 4.2.2 Middleware Development
    - 4.2.3 Security Subsystems
    - 4.2.4 Catalog Engine
    - 4.2.5 Transaction Processing
  - 4.3 Graphics and Interface
  - 4.4 Content Creation
- 5.0 Testing and Production

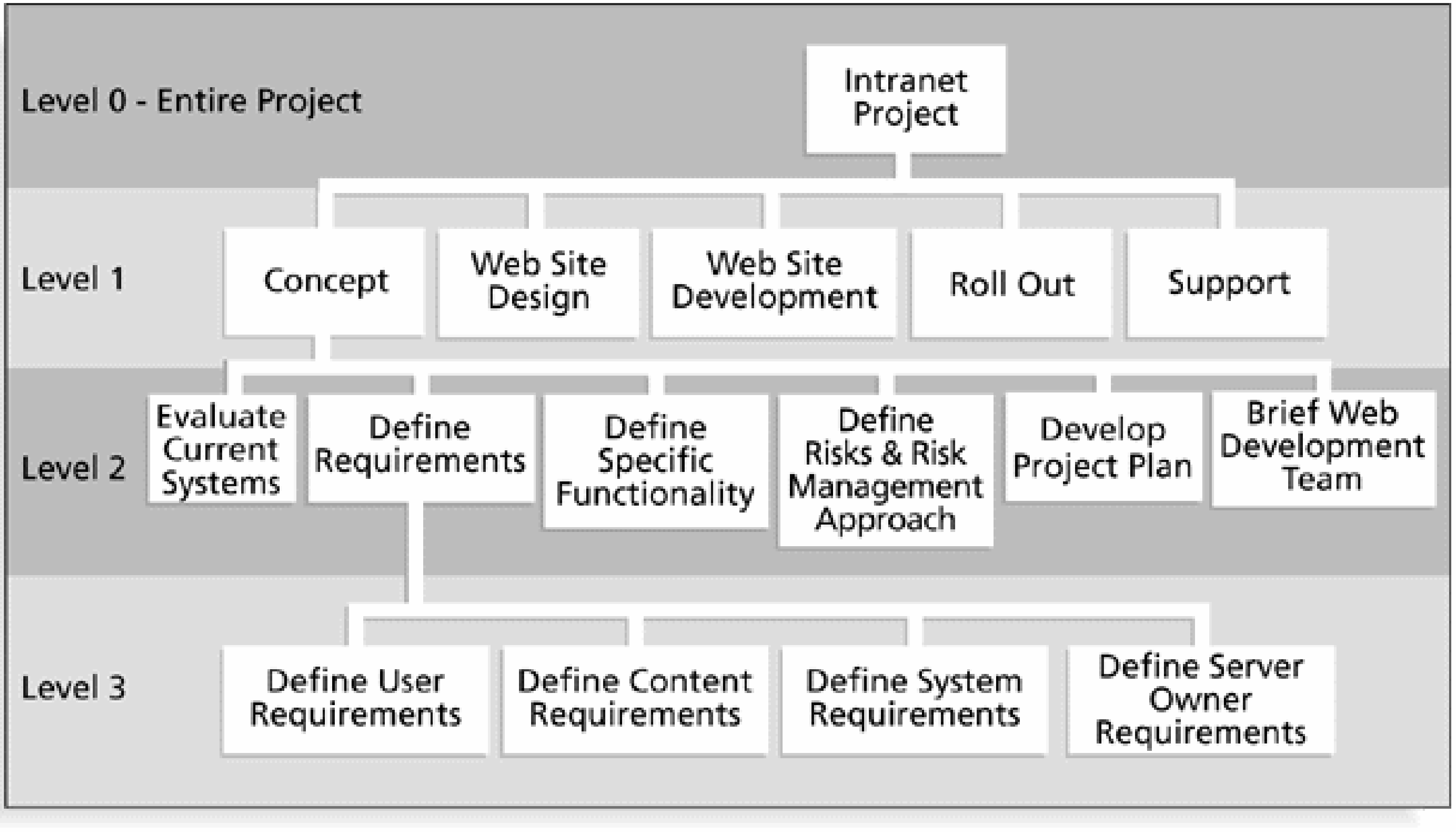
## WBS Types

- Process WBS
  - a.k.a Activity-oriented
  - Ex: Requirements, Analysis, Design, Testing
  - Typically used by PM
- Product WBS
  - a.k.a. Entity-oriented
  - Ex: Financial engine, Interface system, DB
  - Typically used by engineering manager
- Hybrid WBS: both above
  - This is not unusual
  - Ex: Lifecycle phases at high level with component or feature-specifics within phases
  - Rationale: processes produce products

# Product WBS

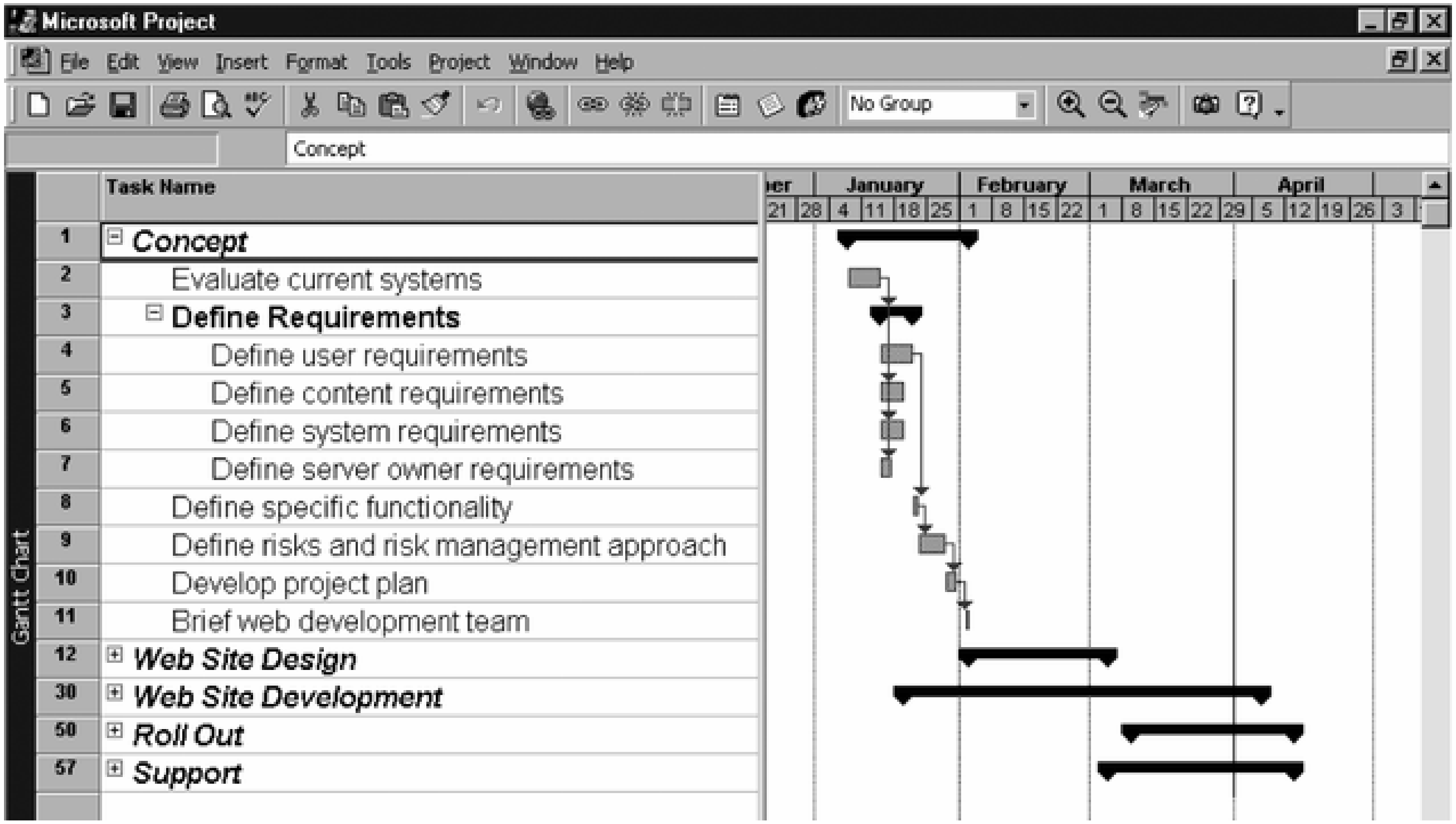

Intranet
Web Site Design
Site Map
Graphic Design
Programs
Home Page Design
Text
Images
Hyperlinks
Marketing Pages
Text
Images
Hyperlinks
Sales Pages
Text
Images
Hyperlinks

# Process WBS



Level 0 - Entire Project
Intranet Project
Level 1
Concept
Web Site Design
Web Site Development
Roll Out
Support
Level 2
Evaluate Current Systems
Define Requirements
Define Specific Functionality
Define Risks & Risk Management Approach
Develop Project Plan
Brief Web Development Team
Level 3
Define User Requirements
Define Content Requirements
Define System Requirements
Define Server Owner Requirements

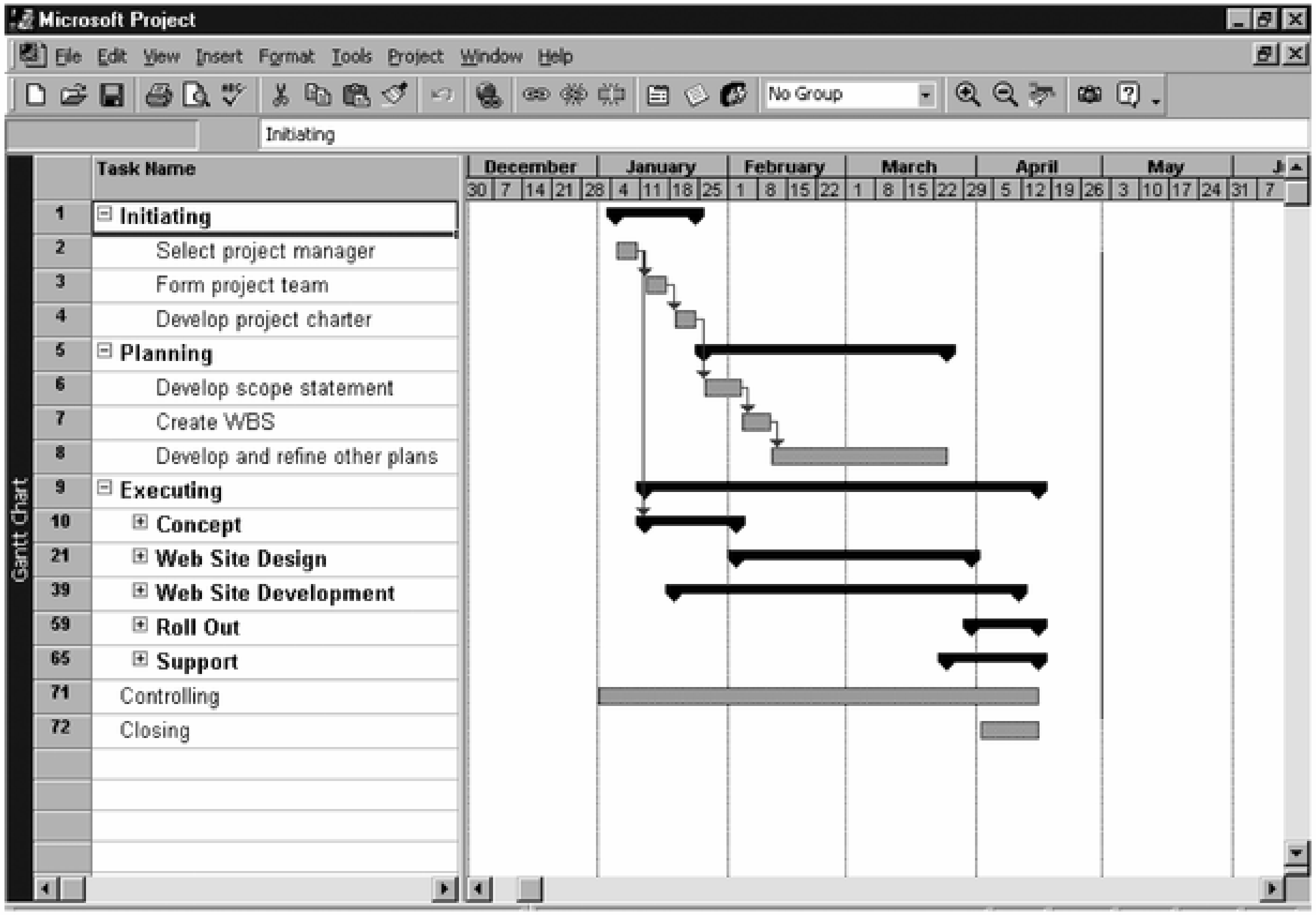
# Outline WBS w/Gantt


Microsoft Project
File Edit View Insert Format Tools Project Window Help
No Group
Gantt Chart
January February March April

Concept

| | Task Name |
|---|---|
| 1 | ⊟ *Concept* |
| 2 | Evaluate current systems |
| 3 | ⊟ **Define Requirements** |
| 4 | Define user requirements |
| 5 | Define content requirements |
| 6 | Define system requirements |
| 7 | Define server owner requirements |
| 8 | Define specific functionality |
| 9 | Define risks and risk management approach |
| 10 | Develop project plan |
| 11 | Brief web development team |
| 12 | ⊞ *Web Site Design* |
| 30 | ⊞ *Web Site Development* |
| 50 | ⊞ *Roll Out* |
| 57 | ⊞ *Support* |

# WBS by PMI Process Groups

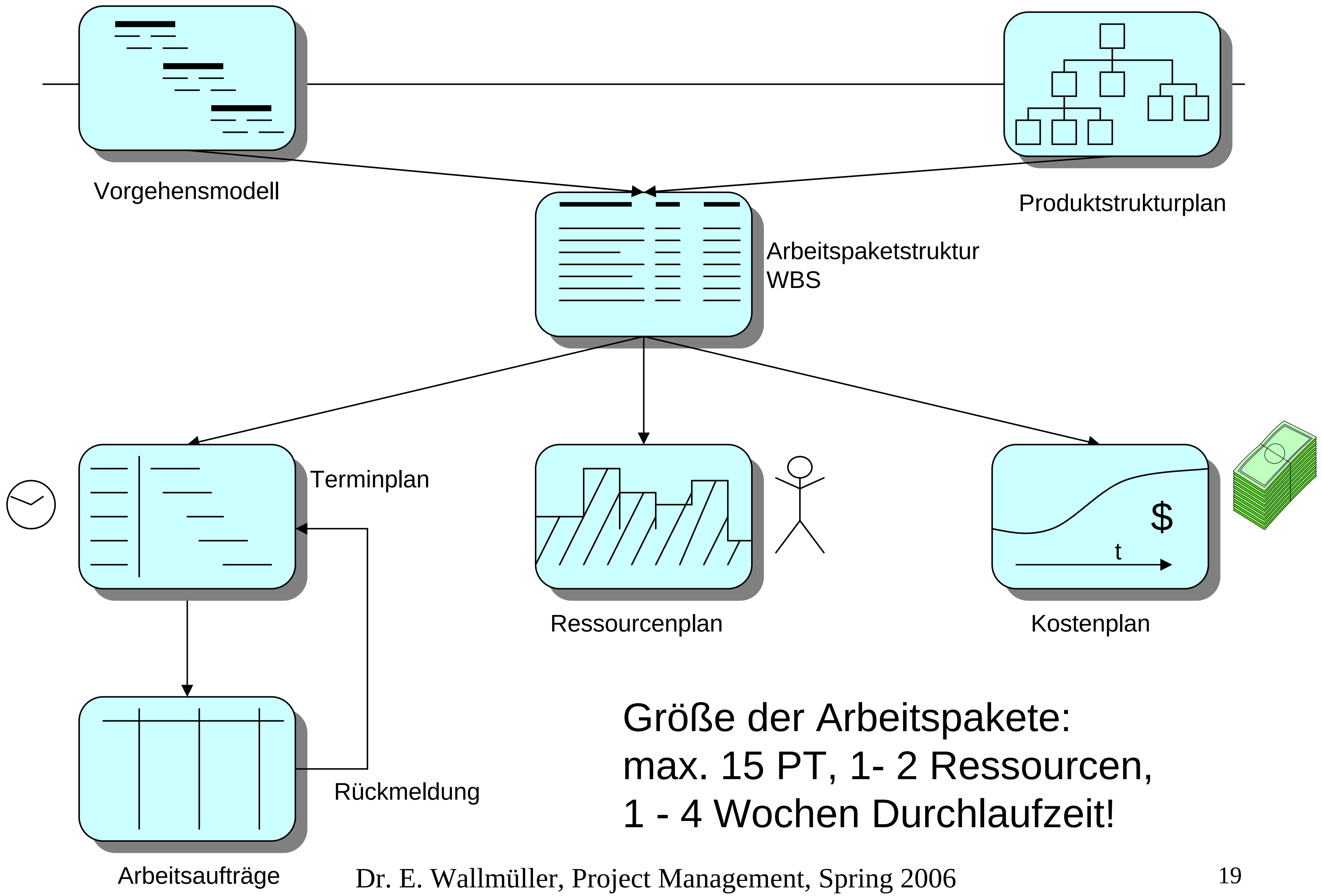


| | Task Name |
|---|---|
| 1 | Initiating |
| 2 | Select project manager |
| 3 | Form project team |
| 4 | Develop project charter |
| 5 | Planning |
| 6 | Develop scope statement |
| 7 | Create WBS |
| 8 | Develop and refine other plans |
| 9 | Executing |
| 10 | Concept |
| 21 | Web Site Design |
| 39 | Web Site Development |
| 59 | Roll Out |
| 65 | Support |
| 71 | Controlling |
| 72 | Closing |

# Work Packages

- Generic term for discrete **tasks** with definable end results
- Typically the “leaves” on the tree
- The “one-to-two” rule
    - Often at: 1 or 2 persons for 1 or 4 weeks
- Basis for monitoring and reporting progress
    - Can be tied to budget items (charge numbers)
    - Resources (personnel) assigned
- Ideally shorter rather than longer
    - Longer makes in-progress estimates needed
    - These are more subjective than “done”
    - 2-3 weeks maximum for software projects
    - 1 day minimum (occasionally a half day)
    - Not so small as to micro-manage

# Planungsschritte



Größe der Arbeitspakete:
max. 15 PT, 1- 2 Ressourcen,
1 - 4 Wochen Durchlaufzeit!

# Estimations

- Very difficult to do, but needed often
- Created, used or refined during
  - Strategic planning
  - Feasibility study and/or SOW
  - Proposals
  - Vendor and sub-contractor evaluation
  - Project planning (iteratively)
- Basic process
  1) Estimate the **size** of the product
  2) Estimate the **effort** (man-months)
  3) Estimate the **schedule**
  - NOTE: Not all of these steps are always explicitly performed

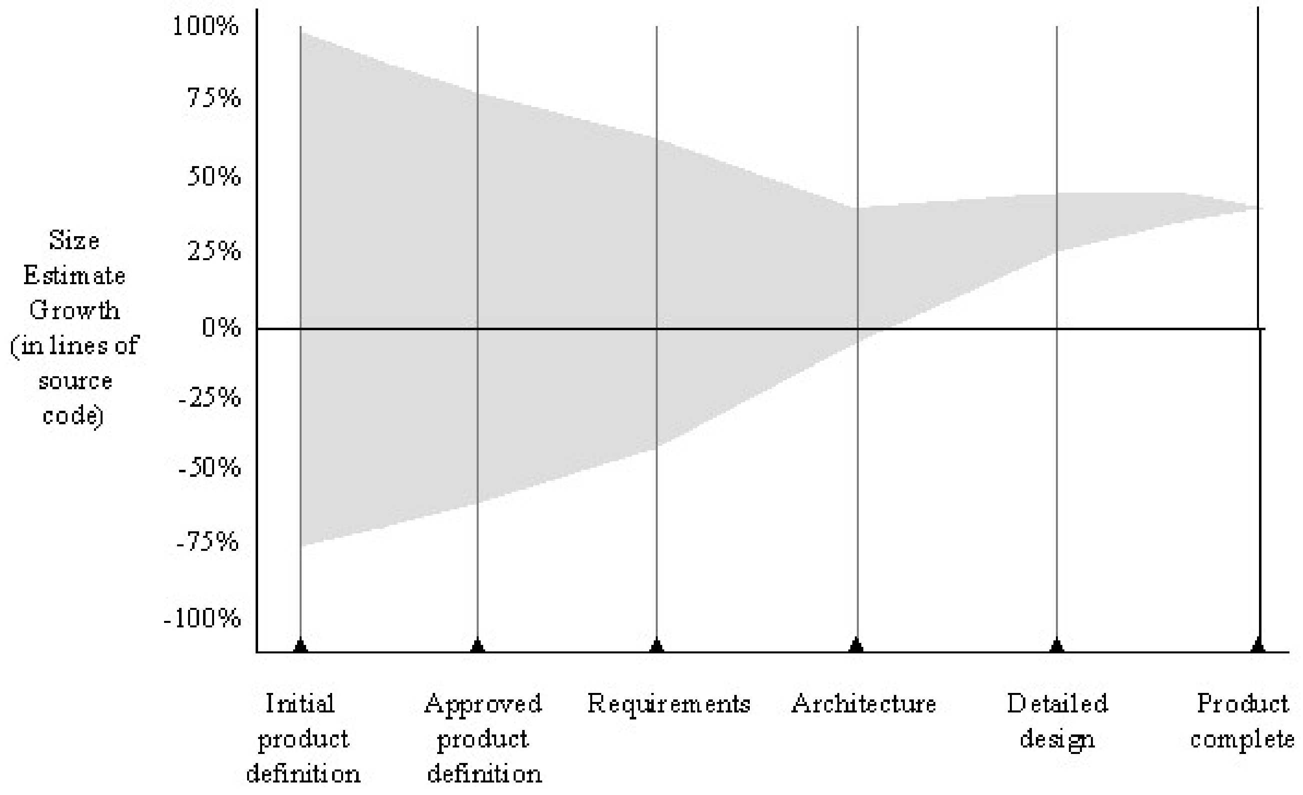
# Estimations

- Remember, an “exact estimate” is an oxymoron
- Estimate how long will it take you to get home from class tonight
  - On what basis did you do that?
  - Experience right?
  - Likely as an “average” probability
  - For most software projects there is no such ‘average’
- Most software estimations are off by 25-100%

## Estimation

- Target vs. Committed Dates
  - Target: Proposed by business or marketing
  - Do not commit to this too soon!
  - Committed: Team agrees to this
  - After you’ve developed a schedule

# Cone of Uncertainty



# Estimation

- Size:
  - Small projects (10-99 FPs), variance of 7% from post-requirements estimates
  - Medium (100-999 FPs), 22% variance
  - Large (1000-9999 FPs) 38% variance
  - Very large (> 10K FPs) 51% variance

## Estimation Methodologies

- Top-down
- Bottom-up
- Analogy
- Expert Judgment
- Priced to Win
- Parametric or Algorithmic Method
  - Using formulas and equations

# Top-down Estimation

- Based on overall characteristics of project
  - Some of the others can be “types” of top-down (Analogy, Expert Judgment, and Algorithmic methods)
- Advantages
  - Easy to calculate
  - Effective early on (like initial cost estimates)
- Disadvantages
  - Some models are questionable or may not fit
  - Less accurate because it doesn’t look at details

## Bottom-up Estimation

- Create WBS
- Add from the bottom-up
- Advantages
  - Works well if activities well understood
- Disadvantages
  - Specific activities not always known
  - More time consuming

## Expert Judgment

- Use somebody who has recent experience on a similar project
- You get a “guesstimate”
- Accuracy depends on their ‘real’ expertise
- Comparable application(s) must be accurately chosen
  - Systematic
- Can use a weighted-average of opinions

## Estimation by Analogy

- Use past project
  - Must be sufficiently similar (technology, type, organization)
  - Find comparable attributes (ex: # of inputs/outputs)
  - Can create a function
- Advantages
  - Based on actual historical data
- Disadvantages
  - Difficulty ‘matching’ project types
  - Prior data may have been mis-measured
  - How to measure differences – no two exactly same

## Algorithmic Measures

- Lines of Code (LOC)
- Function points
- Feature points or object points
- Other possible
  - Number of bubbles on a DFD
  - Number of of ERD entities
  - Number of processes on a structure chart
- LOC and function points most common
  - (of the algorithmic approaches)
- Majority of projects use none of the above

## Code-based Estimates

- LOC Advantages
  - Commonly understood metric
  - Permits specific comparison
  - Actuals easily measured
- LOC Disadvantages
  - Difficult to estimate early in cycle
  - Counts vary by language
  - Many costs not considered (ex: requirements)
  - Programmers may be rewarded based on this
    - Can use: # defects/# LOC
  - Code generators produce excess code

## LOC Estimate Issues

- How do you know how many in advance?
- What about different languages?
- What about programmer style?
- Stat: avg. programmer productivity: 3,000 LOC/yr
- Most algorithmic approaches are more effective after requirements (or have to be after)

## Function Points

- Software size s/b measured by number & complexity of functions it performs
- More methodical than LOC counts
- House analogy
  - House’s Square Feet ~= Software LOC
  - # Bedrooms & Baths ~= Function points
  - Former is size only, latter is size & function
- Six basic steps

# Function Point Process

- 1. Count # of biz functions per category
  - Categories: outputs, inputs, db inquiries, files or data structures, and interfaces
- 2. Establish Complexity Factor for each and apply
  - Simple, Average, Complex
  - Set a weighting multiplier for each (0->15)
  - This results in the “unadjusted function-point total”
- 3. Compute an “influence multiplier” and apply
  - It ranges from 0.65 to 1.35; is based on 14 factors
- 4. Results in “function point total”
  - This can be used in comparative estimates

# Wideband Delphi

- Group consensus approach
- Rand corp. used orig. Delphi approach to predict future technologies
- Present experts with a problem and response form
- Conduct group discussion, collect anonymous opinions, then feedback
- Conduct another discussion & iterate until consensus
- Advantages
  - Easy, inexpensive, utilizes expertise of several people
  - Does not require historical data
- Disadvantages
  - Difficult to repeat
  - May fail to reach consensus, reach wrong one, or all may have same bias

## Wideband Delphi Technique

| STEP | ACTION |
|---|---|
| 1. | Coordinator presents each expert with the project's specification and an estimation form. |
| 2. | Coordinator calls a group meeting in which the experts discuss product issues related to size. |
| 3. | Each expert fills out the form anonymously. |
| 4. | The coordinator prepares a summary of the estimates on an Iteration Form and returns them to the experts. |
| 5. | The coordinator calls a group meeting, primarily to discuss the most widely-varied estimates. |
| 6. | The experts review the summary and submit another anonymous estimate on the Iteration Form. |
| 7. | Steps 4 through 6 are repeated until a consensus of the lowest and highest possible estimates are reached. |

# Wideband Delphi Technique: Iterationform

DELPHI SIZE ESTIMATE ITERATION FORM

PROJECT *Fusion - TM&C CSCI* DATE *7/22/91*

ESTIMATOR *J. Smith* ROUND # *2*

```
________|_____| (x1) | (x2) |_______________________________________________
0       5       10      15      20      25      30      35      40      45      50
```

Size Estimate in KSLOC

X - Estimates, X(1) - Your Estimate, X(2) - Median Expert

Please enter your estimate for the next round *10K* SLOC

Please explain any rationale for your estimate.

*experience with a similar project*

## Parametric Method Issues

- Remember: most projects you’ll run into don’t use these
- Which is ‘normal’, so don’t be surprised
  - Or come-in to new job and say “Hey, let’s use COCOMO”
- These are more effective on large projects
  - Where a past historical base exists
- Primary issue for most projects are
  - Lack of similar projects
    - Thus lack of comparable data

## Effort Estimation

- Now that you know the “size”, determine the “effort” needed to build it
- Various models: empirical, mathematical, subjective
- Expressed in units of duration
  - Man-months (or ‘staff-months’ now)

# Effort Estimation

- McConnell shows schedule tables for conversion of size to effort
- As with parametric size estimation, these techniques perform better with historical data
- Again, not seen in ‘average’ projects
- Often the size and effort estimation steps are combined (not that this is recommended, but is what often is done)
- “Commitment-Based” Scheduling is what is often done
  - Ask developer to ‘commit’ to an estimate (his or her own)

# COCOMO

- COnstructive COst MOdel
- Allows for the type of application, size, and "Cost Drivers"
- Outputs in Person Months
- Cost drivers using High/Med/Low & include
  - Motivation
  - Ability of team
  - Application experience
- Biggest weakness?
  - Requires input of a product size estimate in LOC

## Home Work

- The Essence of COCOMO II
- Only 7 slides

# Estimation Issues

- Quality estimations needed early but information is limited
- Precise estimation data available at end but not needed
  - Or is it? What about the next project?
- Best estimates are based on past experience
- Politics of estimation:
  - You may anticipate a "cut" by upper management
- For many software projects there is little or none
  - Technologies change
  - Historical data unavailable
  - Wide variance in project experiences/types
  - Subjective nature of software estimation

# Over and Under Estimation

- Over estimation issues
  - The project will not be funded
    - Conservative estimates guaranteeing 100% success may mean funding probability of zero.
  - Parkinson’s Law: Work expands to take the time allowed
  - Danger of feature and scope creep
  - Be aware of “double-padding”: team member + manager
- Under estimation issues
  - Quality issues (short changing key phases like testing)
  - Inability to meet deadlines
  - Morale and other team motivation issues

## Estimation Guidelines

- Estimate iteratively!
  - Process of gradual refinement
  - Make your best estimates at each planning stage
  - Refine estimates and adjust plans iteratively
  - Plans and decisions can be refined in response
  - Balance: too many revisions vs. too few

## Know Your Deadlines

- Are they ‘Real Deadlines’?
  - Tied to an external event
  - Have to be met for project to be a success
  - Ex: end of financial year, contractual deadline, Y2K
- Or ‘Artificial Deadlines’?
  - Set by arbitrary authority
  - May have some flexibility (if pushed)

# Estimation “Presentation”

- How you present the estimation can have **huge** impact
- Techniques
    - Plus-or-minus qualifiers
        - 6 months +/-1 month
    - Ranges
        - 6-8 months
    - Risk Quantification
        - +/- with added information
        - +1 month of new tools not working as expected
        - -2 weeks for less delay in hiring new developers
    - Cases
        - Best / Planned / Current / Worst cases
    - Coarse Dates
        - Q3 02
    - Confidence Factors
        - April 1 – 10% probability, July 1 – 50%, etc.

## Financial Analysis of Projects

- Financial considerations are often an important consideration in selecting projects
- Three primary methods for determining the projected financial value of projects:
  - Net present value (NPV) analysis
  - Return on investment (ROI)
  - Payback analysis

## Questions?