

▶ POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Session 4

WBS, Estimation & Scheduling

Emanuele Della Valle

<http://home.dei.polimi.it/dellavalle>

- This slides are largely based on Prof. John Musser class notes on “Principles of Software Project Management”
- Original slides are available at <http://www.projectreference.com/>
- Reuse and republish permission was granted

Today & Next Week

3

- Finish & Review Session 3
- Work Breakdown Structures (WBS)
- Estimation
- Network Fundamentals
- PERT & CPM Techniques
- Gantt Charts
- This slide set will likely roll-over to next week

- Full Lifecycle
 - Know your pure waterfall, 7 phase model
 - Understand the steps in each phase
 - Know the primary issues and goals of each
 - An example at a little consulting corporation
 - <http://www.creativedd.com/projlifecycle.html>
- Methodologies
 - Know a representative sample
 - Waterfall, an variation, 1-2 iterative ones
 - Be familiar with spectrum from XP->RUP
- Planning, we'll cover that now...

- Web-based customized reporting project
- “How to Fail with the Rational Unified Process or 7 Steps to Pain and Suffering”
 - <http://www.ceng.metu.edu.tr/~gtumuklu/web/SE548/Reading%20Material/HowToFailWRational.pdf>

- Planning
 - http://www.emanueledellavalle.org/slides/P&MSP2009_04_Planning.pdf

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

- What's the difference?
- Plan: Identify activities. No specific start and end dates.
- Estimating: Determining the size & duration of activities.
- Schedule: Adds specific start and end dates, relationships, and resources.

Project Planning: A 12 Step Program

9

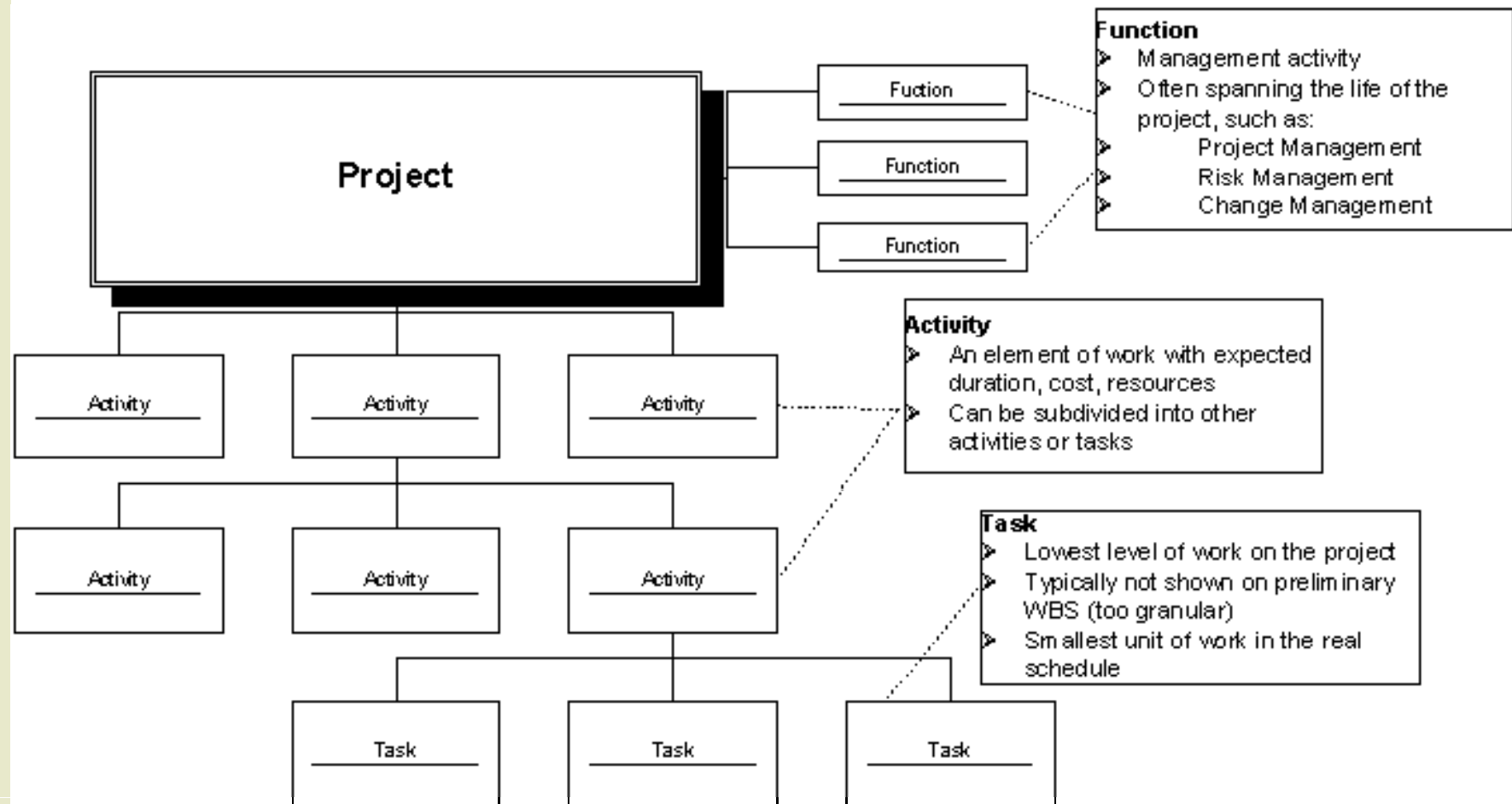
- Set goal and scope
- Select lifecycle
- Set org./team form
- Start team selection
- Determine risks
- Create WBS
- Identify tasks
- Estimate size
- Estimate effort
- Identify task dependencies
- Assign resources
- Schedule work

- 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

- How did you feel when I asked
 - “How long will your project take?”
- Not an easy answer to give right?
- At least not if I were a real customer on a real project
- How can you manage that issue?

- 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?

- A Project: functions, activities, tasks

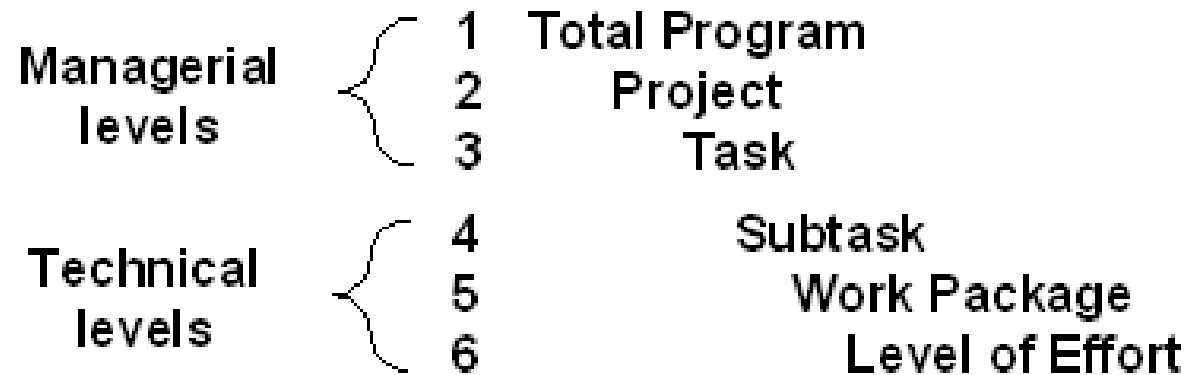


- Hierarchical list of project's work activities
- 2 Formats
 - Outline (indented format)
 - Graphical Tree (Organizational Chart)
- Uses a decimal numbering system
 - Ex: 3.1.5
 - 0 is typically top level
- Includes
 - Development, Mgmt., and project support tasks
- Shows "is contained in" relationships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

- 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

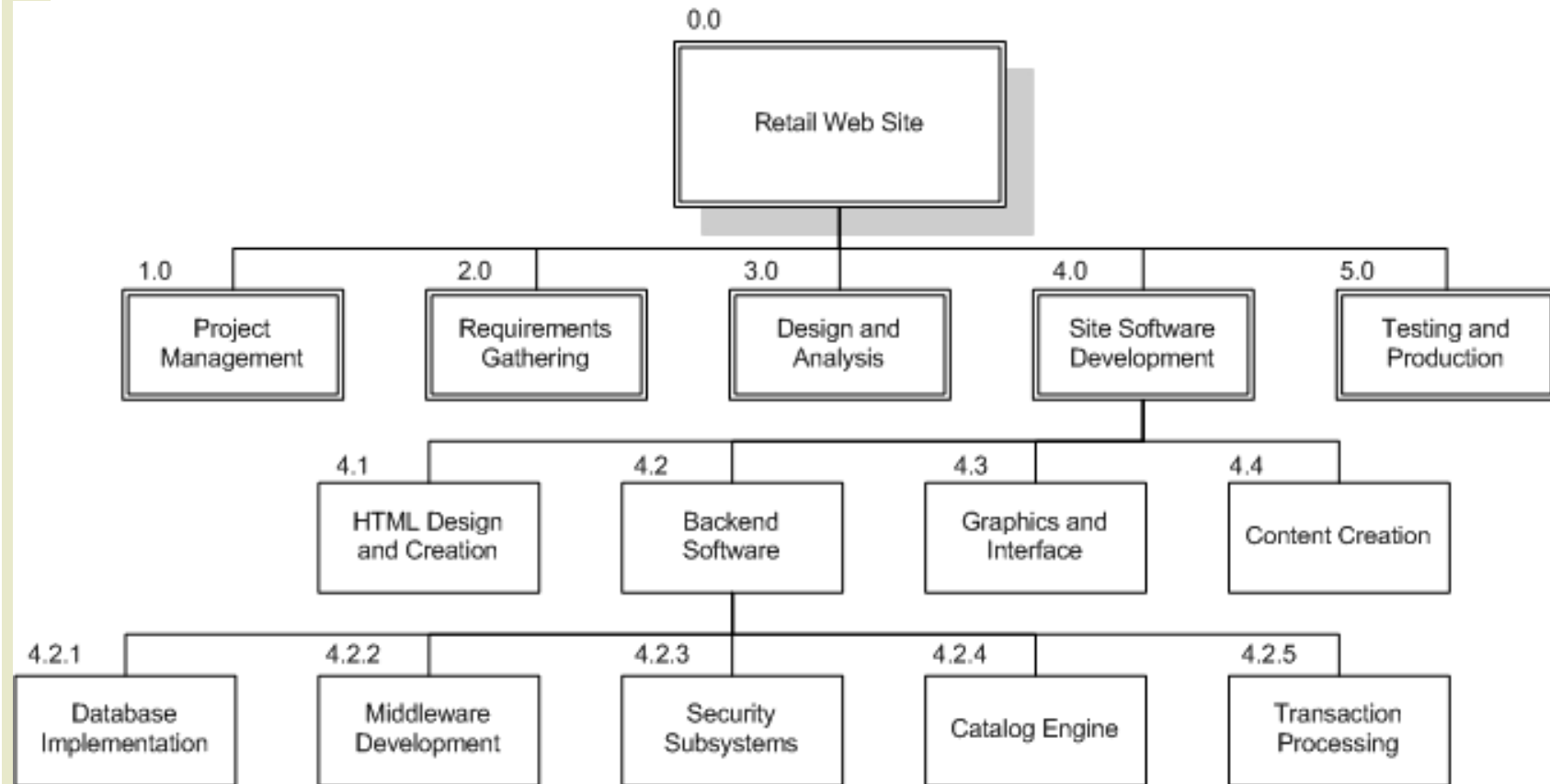
- Up to six levels (3-6 usually) such as



- 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

17

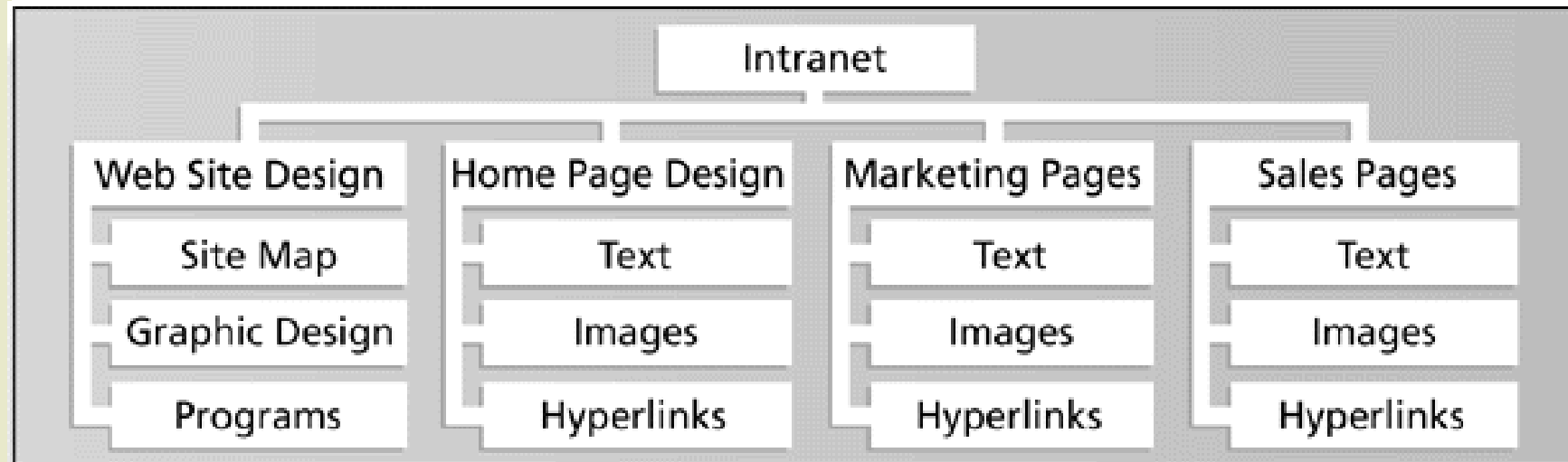


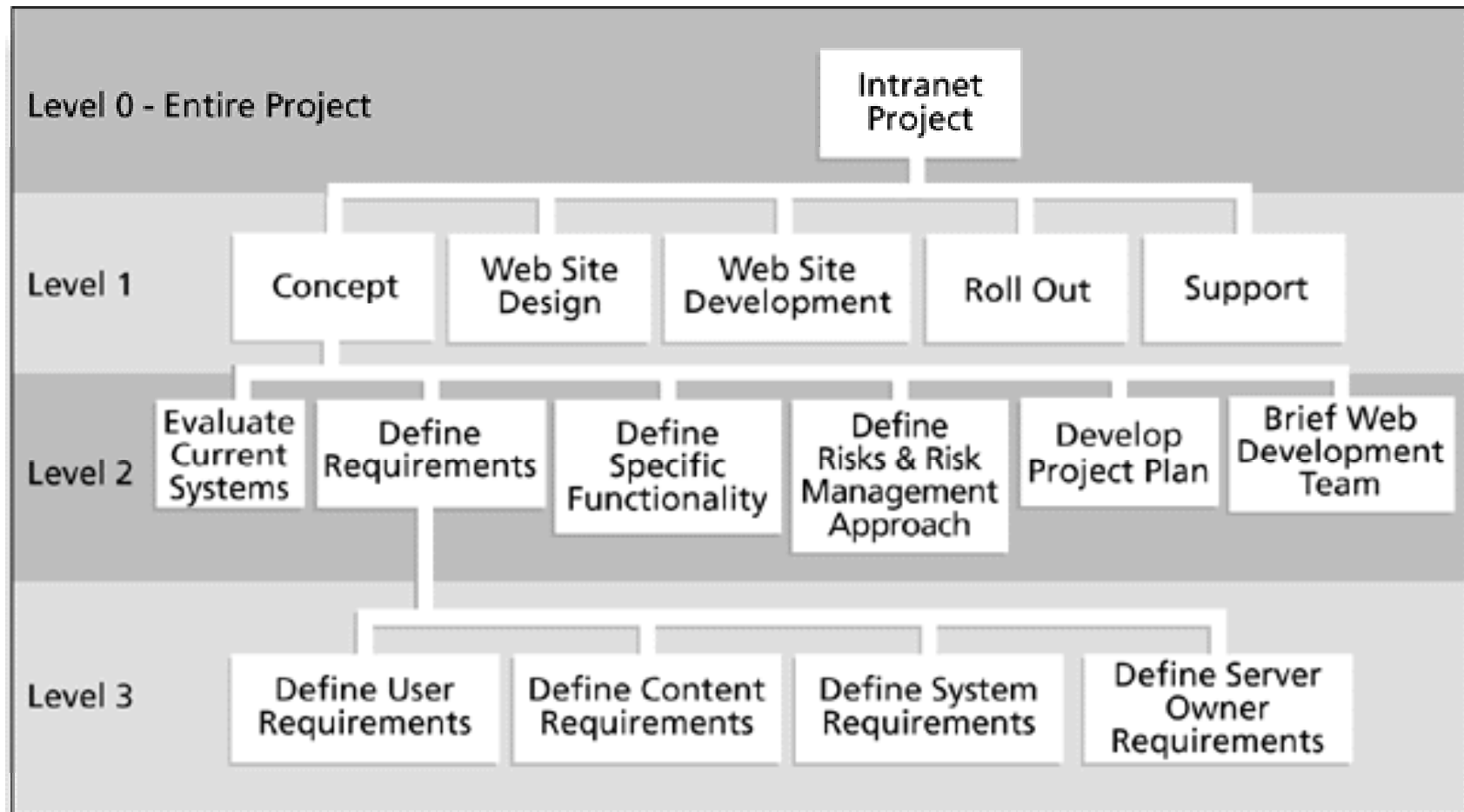
- 0.0 Retail Web Site
- 1.0 Project Management
- 2.0 Requirements Gathering
- 3.0 Analysis & Design
- 4.0 Site Software Development
 - 4.1 HTML Design and Creation
 - 4.2 Backend Software
 - 4.2.1 Database Implementation
 - 4.2.2 Middleware Development
 - 4.2.3 Security Subsystems
 - 4.2.4 Catalog Engine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

- 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

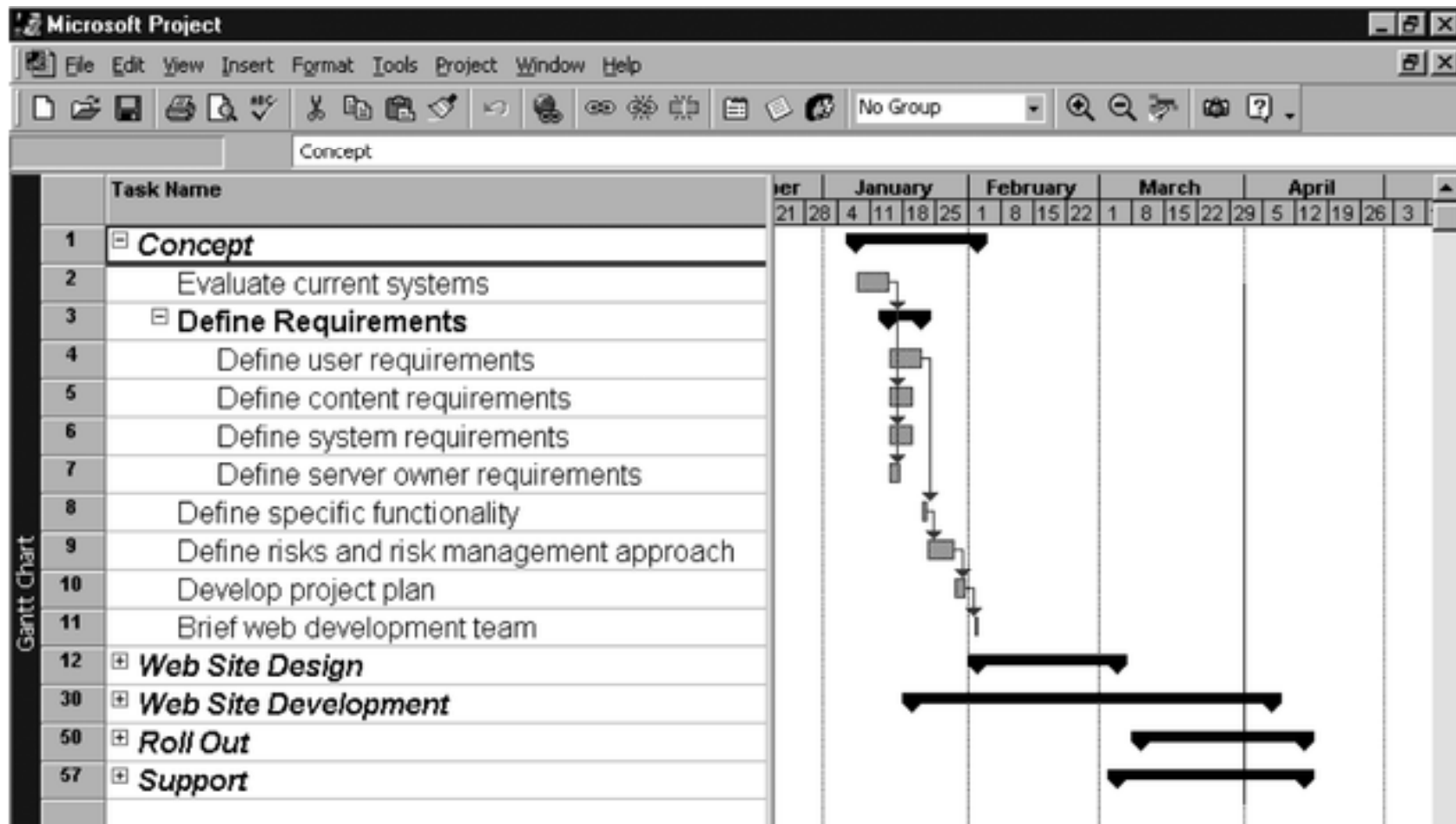
- Hybrid WBS: both above
 - This is not unusual
 - Ex: Lifecycle phases at high level with component or feature-specifics within phases
 - Rationale: processes produce products





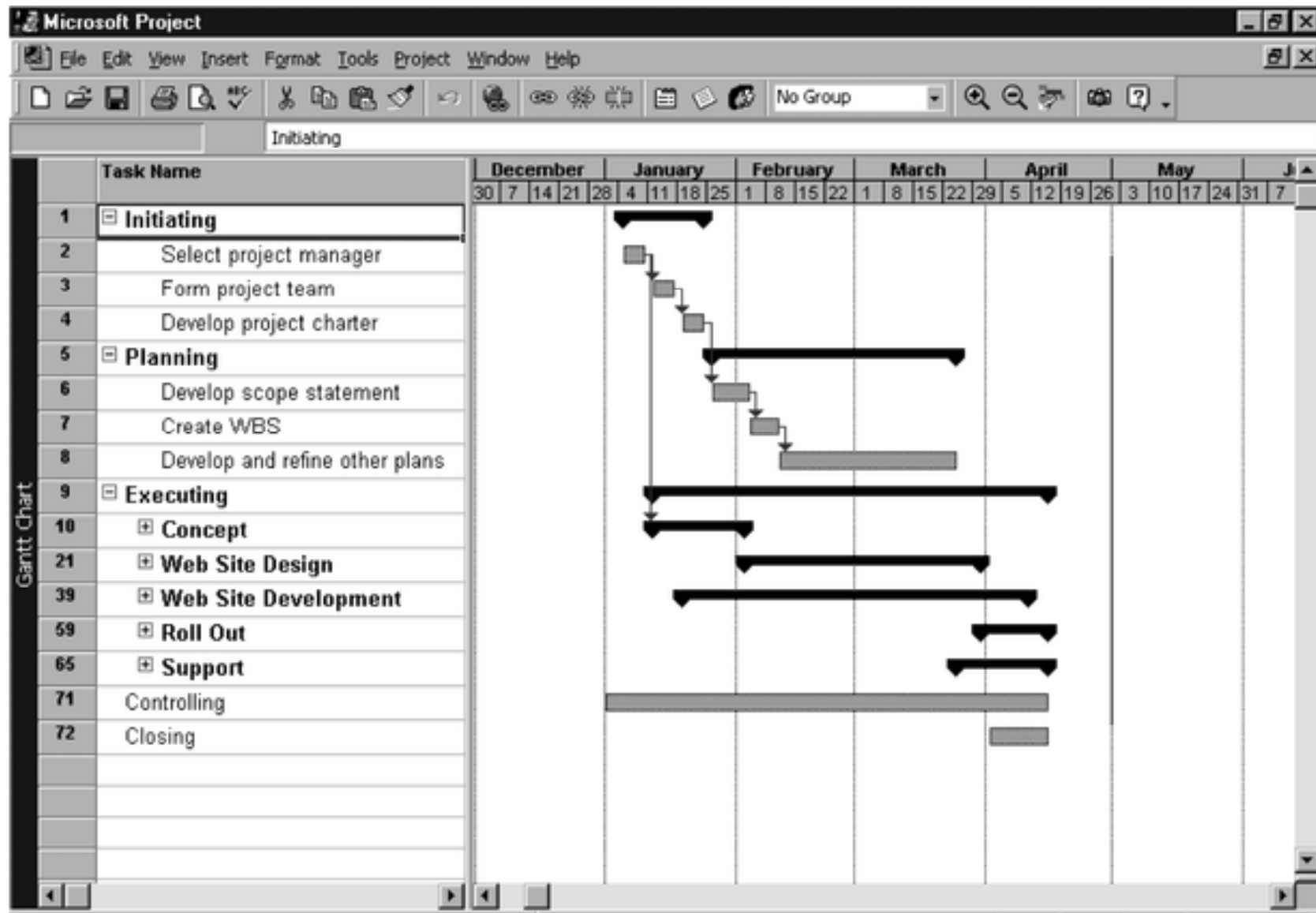
Outline WBS w/Gantt

22



WBS by PMI Process Groups

23



- Less frequently used alternatives
 - Organizational WBS
 - Research, Product Design, Engineering, Operations
 - Can be useful for highly cross-functional projects
 - Geographical WBS
 - Can be useful with distributed teams
 - NYC team, San Jose team, Off-shore team

- 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 2 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

- List of Activities, not Things
- List of items can come from many sources
 - SOW, Proposal, brainstorming, stakeholders, team
- Describe activities using “bullet language”
 - Meaningful but terse labels
- All WBS paths do not have to go to the same level
- Do not plan more detail than you can manage

- PM must map activities to chosen lifecycle
- Each lifecycle has different sets of activities
- Integral process activities occur for all
 - Planning, configuration, testing
- Operations and maintenance phases are not normally in plan (considered post-project)
- Some models are “straightened” for WBS
 - Spiral and other iterative models
 - Linear sequence several times
- Deliverables of tasks vary by methodology

- Top-Down
- Bottom-Up
- Analogy
- Rolling Wave
 - 1st pass: go 1-3 levels deep
 - Gather more requirements or data
 - Add more detail later
- Post-its on a wall

- Top-down
 - Start at highest level
 - Systematically develop increasing level of detail
 - Best if
 - The problem is well understood
 - Technology and methodology are not new
 - This is similar to an earlier project or problem
 - But is also applied in majority of situations

- Bottom-up
 - Start at lowest level tasks
 - Aggregate into summaries and higher levels
 - Cons
 - Time consuming
 - Needs more requirements complete
 - Pros
 - Detailed

- Analogy
 - Base WBS upon that of a “similar” project
 - Use a template
 - Analogy also can be estimation basis
 - Pros
 - Based on past actual experience
 - Cons
 - Needs comparable project

- Brainstorming
 - Generate all activities you can think of that need to be done
 - Group them into categories
- Both Top-down and Brainstorming can be used on the same WBS
- Remember to get the people who will be doing the work involved (buy-in matters!)

WBS – Basis of Many Things

33

- Network scheduling
- Costing
- Risk analysis
- Organizational structure
- Control
- Measurement

- Should be easy to understand
- Some companies have corporate standards for these schemes
- Some top-level items, like Project Mgmt. are in WBS for each project
 - Others vary by project
- What often hurts most is what's missing
- Break down until you can generate accurate time & cost estimates
- Ensure each element corresponds to a deliverable

- How detailed should it be?
 - Not as detailed as the final MS-Project plan
 - Each level should have no more than 7 items
 - It can evolve over time
- What tool should you use?
 - Excel, Word, Project
 - Org chart diagramming tool (Visio, etc)
 - Specialized commercial apps
- Re-use a “template” if you have one

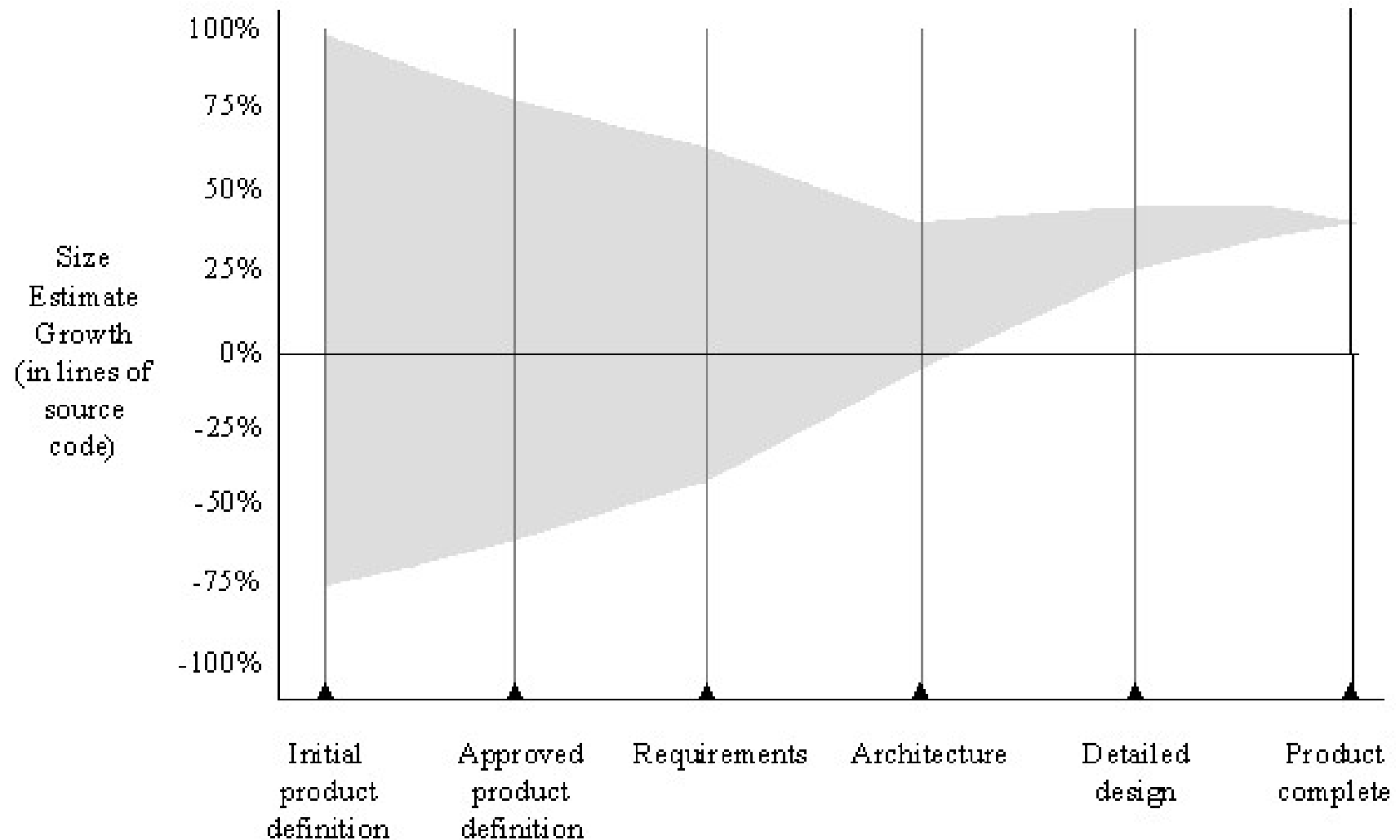
- Very difficult to do, but needed often
- Created, used or refined during
 - Strategic planning
 - Feasibility study and/or SOW
 - Proposals
 - Vendor and sub-contractor evaluation
 - Project planning (iteratively)
- Basic process
 - Estimate the size of the product
 - Estimate the effort (man-months)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

- 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%

- 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

39



Copyright 1998 Steven C. McConnell. Reprinted with permission from *Software Project Survival Guide* (Microsoft Press, 1998).

- 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

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

- 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

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

- 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

- 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

- Just follow other estimates
- Save on doing full estimate
- Needs information on other estimates (or prices)
- Purchaser must closely watch trade-offs
- Priced to lose?

- 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

- 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

- 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)

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

- 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

- 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

- 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
- Catch-22: how to get started
 - See [http://en.wikipedia.org/wiki/Catch-22_\(logic\)](http://en.wikipedia.org/wiki/Catch-22_(logic))

- Does not come for free
- Code types: New, Modified, Reused
- If code is more than 50% modified, it's "new"
- Reuse factors have wide range
 - Reused code takes 30% effort of new
 - Modified is 60% of new
- Integration effort with reused code almost as expensive as with new code

- 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)

- 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)

- 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

- 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 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

- 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

- 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)

- 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.

- Account for resource experience or skill
 - Up to a point
 - Often needed more on the “low” end, such as for a new or junior person
- Allow for “non-project” time & common tasks
 - Meetings, phone calls, web surfing, sick days
- There are commercial ‘estimation tools’ available
 - They typically require configuration based on past data

- Remember: “manage expectations”
- Parkinson’s Law
 - “Work expands to fill the time available”
- The Student Syndrome
 - Procrastination until the last minute (cram)

- 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

Net Present Value Analysis: NPV

66

- NPV: a method of calculating the expected net monetary gain or loss from a project by discounting all expected future cash inflows and outflows to the present point in time
- Projects with a positive NPV should be considered if financial value is a key criterion
- The higher the NPV, the better

NPV Example

	A	B	C	D	E	F	G	H	I	J	
2											
3	AN. INT. RATE -->	10%									
4											
5	PROJECT 1	YEAR 1	YEAR 2	YEAR 3	YEAR 4	YEAR 5	TOTAL				
6	REVENUES	\$0	\$2,000	\$3,000	\$4,000	\$5,000	\$14,000				
7	COSTS	\$5,000	\$1,000	\$1,000	\$1,000	\$1,000	\$9,000				
8	CASH FLOW	(\$5,000)	\$1,000	\$2,000	\$3,000	\$4,000	\$5,000				
9	NPV	\$2,316									
10		Formula =npv(b3,b8:f8)									
11											
12	PROJECT 2	YEAR 1	YEAR 2	YEAR 3	YEAR 4	YEAR 5	TOTAL				
13	REVENUES	\$1,000	\$2,000	\$4,000	\$4,000	\$4,000	\$15,000				
14	COSTS	\$2,000	\$2,000	\$2,000	\$2,000	\$2,000	\$10,000				
15	CASH FLOW	(\$1,000)	\$0	\$2,000	\$2,000	\$2,000	\$5,000				
16	NPV	\$3,201									
17		Formula =npv(b3,b15:f15)									
18											
19	RECOMMEND PROJECT 2 BECAUSE IT HAS THE HIGHER NPV.										
20											
21	IF STATEMENT -->	=IF(B9>B16,A5,A12)									
22	RESULT -->	PROJECT 2									

Notice that cash flow totals are the same, but NPVs are different.

Return on Investment (ROI)

68

- ROI: income divided by investment
 - $ROI = (\text{total discounted benefits} - \text{total discounted costs}) / \text{discounted costs}$
- The higher the ROI, the better
- Many organizations have a required rate of return or minimum acceptable rate of return on investment for projects

- Another important financial consideration is payback analysis
- The “payback period” is the amount of time it will take to recoup, in the form of net cash inflows, the net dollars invested in a project
- Payback occurs when the cumulative discounted benefits and costs are greater than zero
- Many organizations want IT projects to have a fairly short payback period

NPV, ROI, Payback Period: Ex 1

70

	A	B	C	D	E	F	G	H
1								
2	DISCOUNT RATE →	10%		Years				
3		1	2	3	4	5	TOTAL	
4	COSTS	(\$5,000)	(\$1,000)	(\$1,000)	(\$1,000)	(\$1,000)	-9,000	
5	DISCOUNT FACTOR	0.91	0.83	0.75	0.68	0.62		
6	DISCOUNTED COSTS	-4,545	-826	-751	-683	-621	-7,427	
7								
8	BENEFITS	\$0	\$2,000	\$3,000	\$4,000	\$5,000	14,000	
9	DISCOUNT FACTOR	0.91	0.83	0.75	0.68	0.62		
10	DISCOUNTED BENEFITS	0	1,653	2,254	2,732	3,105	9,743	
11								
12	DISCOUNTED BENEFITS + COSTS	-4,545	826	1,503	2,049	2,484	2,316 ← NPV	
13	CUMULATIVE BENEFITS + COSTS	-4,545	-3,719	-2,216	-167	2,316	4,633	
14						↑		
15	ROI	31%				Payback in this year		

NPV, ROI, Payback Period: Ex 2

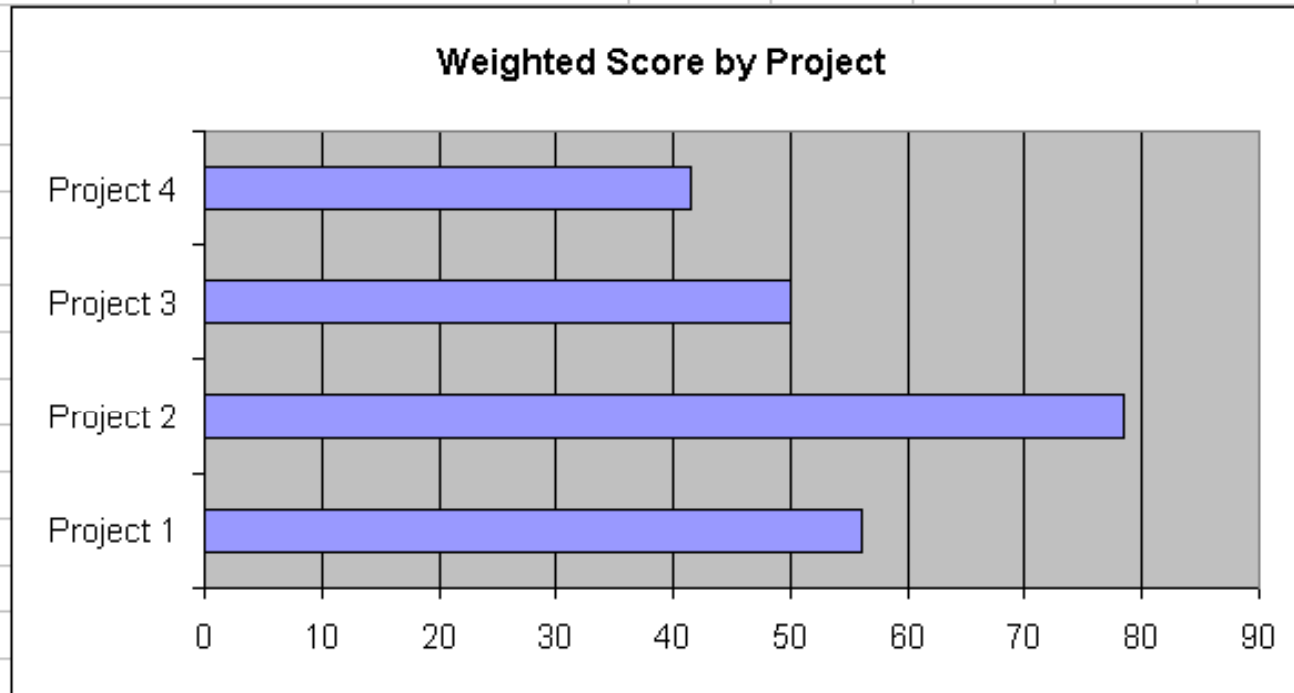
71

	A	B	C	D	E	F	G	H
1								
2	DISCOUNT RATE →	10%		Years				
3		1	2	3	4	5	TOTAL	
4	COSTS	(\$2,000)	(\$2,000)	(\$2,000)	(\$2,000)	(\$2,000)	-10,000	
5	DISCOUNT FACTOR	0.91	0.83	0.75	0.68	0.62		
6	DISCOUNTED COSTS	-1,818	-1,653	-1,503	-1,366	-1,242	-7,582	
7								
8	BENEFITS	\$1,000	\$2,000	\$4,000	\$4,000	\$4,000	15,000	
9	DISCOUNT FACTOR	0.91	0.83	0.75	0.68	0.62		
10	DISCOUNTED BENEFITS	909	1,653	3,005	2,732	2,484	10,783	
11								
12	DISCOUNTED BENEFITS + COSTS	-909	0	1,503	1,366	1,242	3,201 ← NPV	
13	CUMULATIVE BENEFITS + COSTS	-909	-909	594	1,960	3,201	6,403	
14				↑				
15	ROI	42%		Payback in this year				

- A weighted scoring model is a tool that provides a systematic process for selecting projects based on many criteria
 - First identify criteria important to the project selection process
 - Then assign weights (percentages) to each criterion so they add up to 100%
 - Then assign scores to each criterion for each project
 - Multiply scores * weights = total weighted scores
- The higher the weighted score, the better

Sample Weighted Scoring Model

	A	B	C	D	E	F
1	Criteria	Weight	Project 1	Project 2	Project 3	Project 4
2	Supports key business objectives	25%	90	90	50	20
3	Has strong internal sponsor	15%	70	90	50	20
4	Has strong customer support	15%	50	90	50	20
5	Realistic level of technology	10%	25	90	50	70
6	Can be implemented in one year or less	5%	20	20	50	90
7	Provides positive NPV	20%	50	70	50	50
8	Has low risk in meeting scope, time, and cost goals	10%	20	50	50	90
9	Weighted Project Scores	100%	56	78.5	50	41.5
10						



- McConnell: 9, "Scheduling"
- Schwalbe: 5, "Project Time Management"
- URLs: See projectreference.com on "PERT/CPM"
 - <http://www.projectreference.com/#PERT>

- Create a WBS for your project
- The fine print
 - You only need to do this down to level 2 (or 3 if your so inclined or that helps clarify some higher-level activities). Going to level 2 means means start with level 0 (a single node – the name of your project), level 1, the highest level breakdown (4-7 nodes), and one level below that. You can go deeper at your discretion.
 - As we covered in class, you can use either a process, product or hybrid approach. For most of your projects I suspect the process approach would work best at the highest level.
 - You can use either Excel, Word, or Project. Follow the standard hierarchical numbering scheme for WBS structures.
 - Please think this through. You're the PM now.

Questions?

76