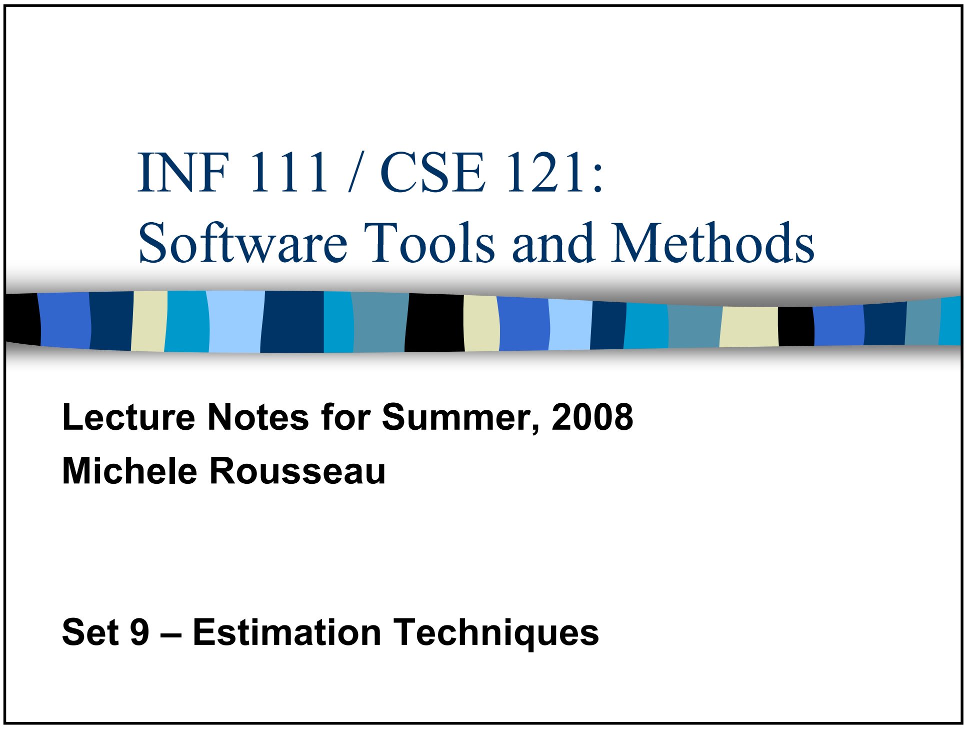
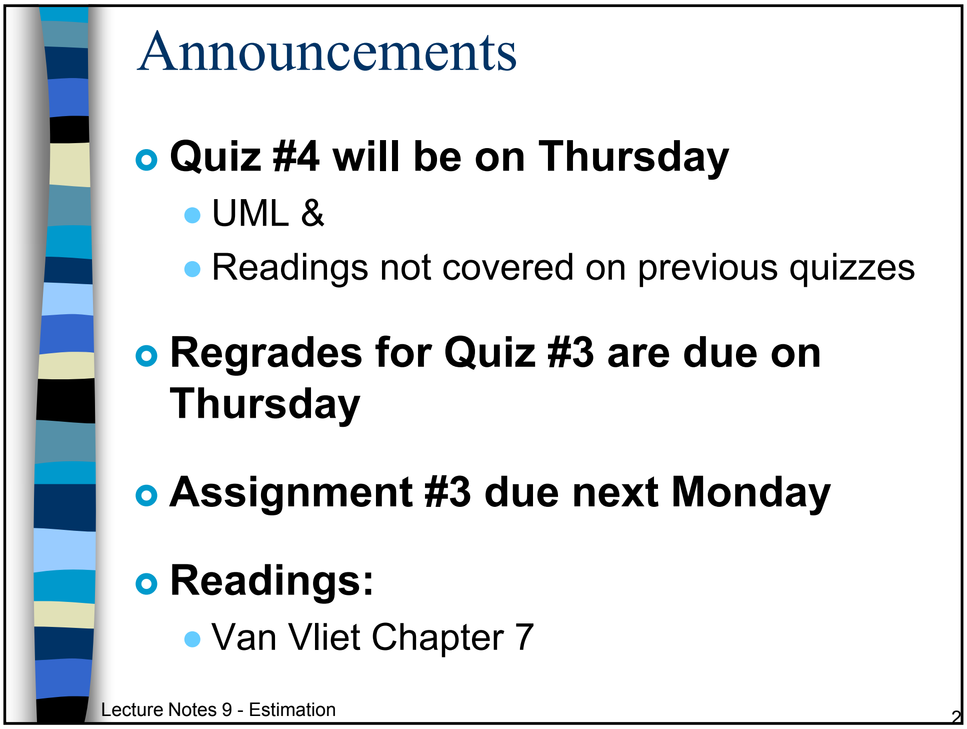


INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Summer, 2008

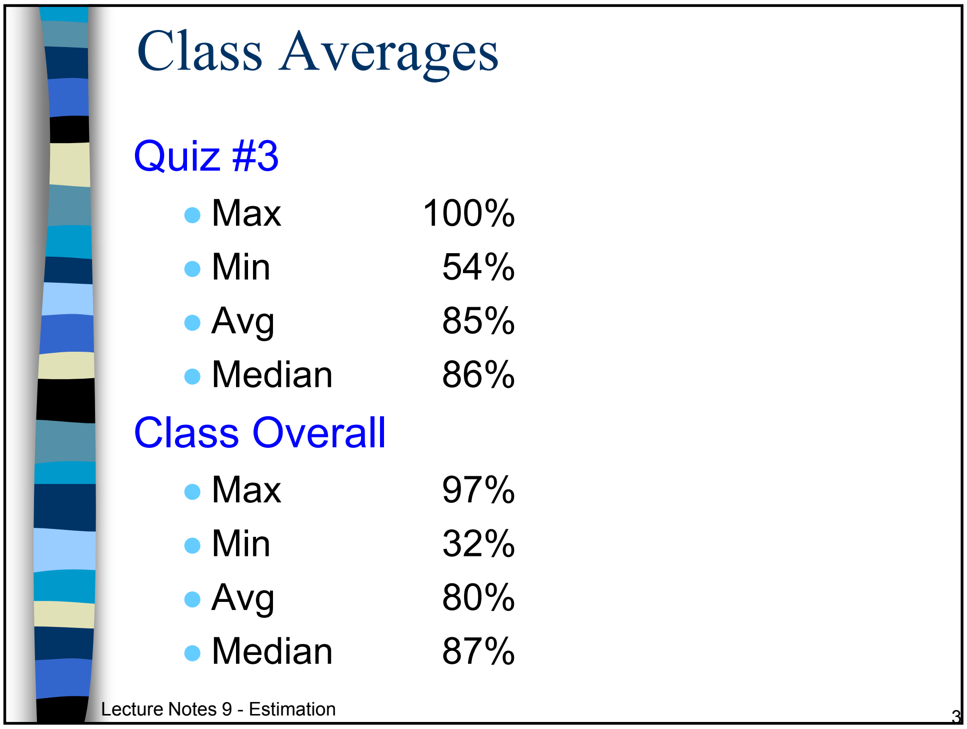
Michele Rousseau

Set 9 – Estimation Techniques



Announcements

- **Quiz #4 will be on Thursday**
  - UML &
  - Readings not covered on previous quizzes
- **Regrades for Quiz #3 are due on Thursday**
- **Assignment #3 due next Monday**
- **Readings:**
  - Van Vliet Chapter 7



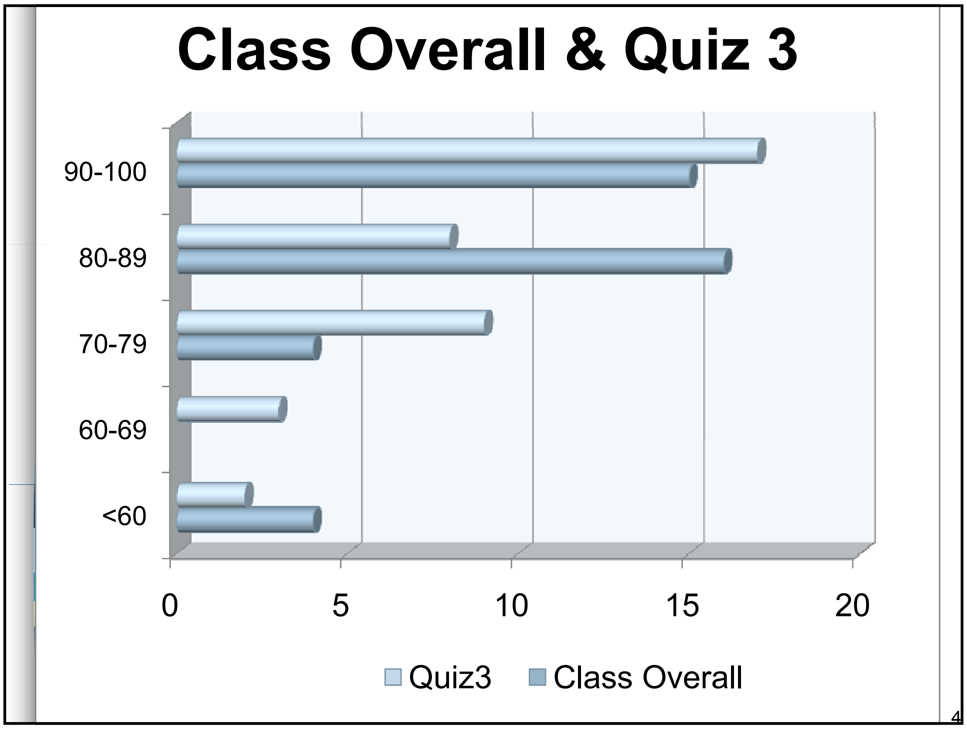
Class Averages

Quiz #3

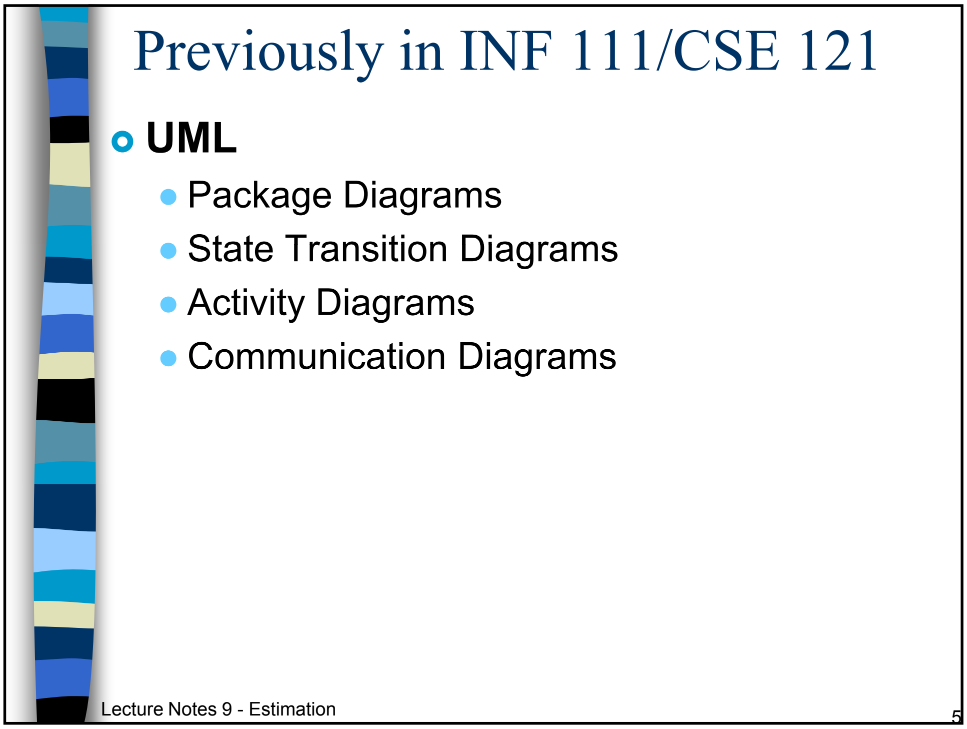
Max	100%
Min	54%
Avg	85%
Median	86%

Class Overall

Max	97%
Min	32%
Avg	80%
Median	87%

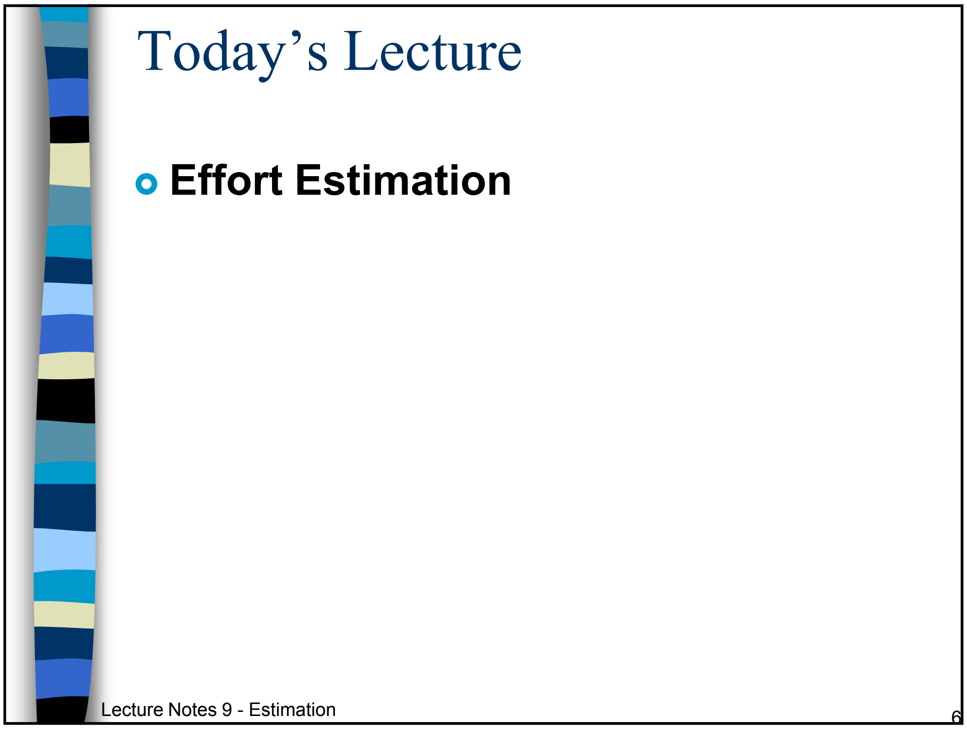


Class Overall & Quiz 3



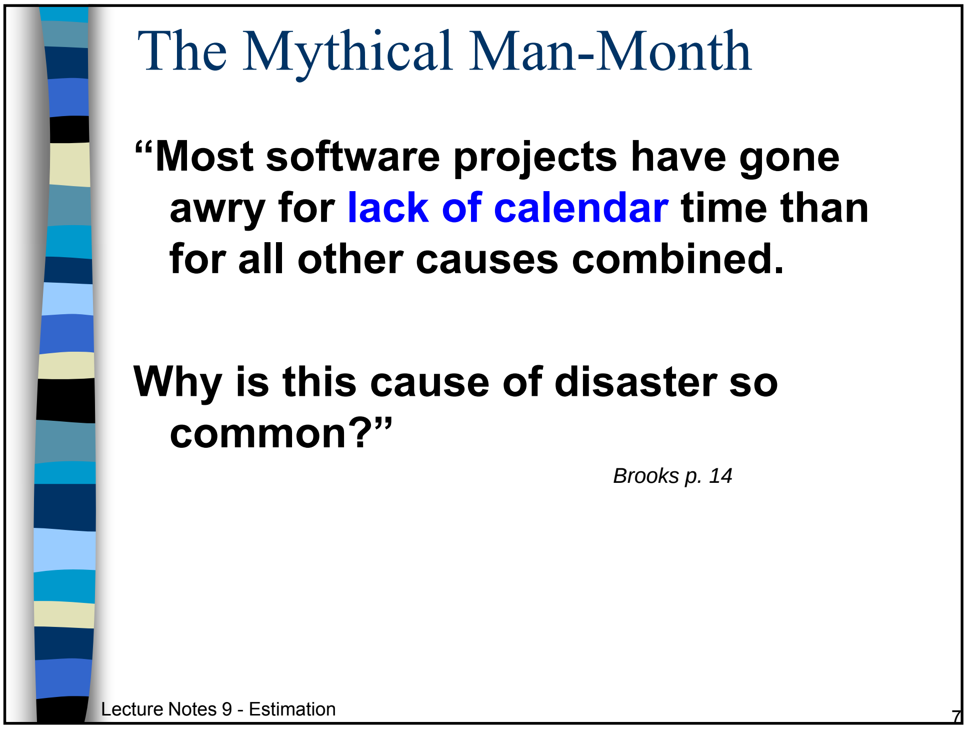
Previously in INF 111/CSE 121

- **UML**
  - Package Diagrams
  - State Transition Diagrams
  - Activity Diagrams
  - Communication Diagrams



Today’s Lecture

- **Effort Estimation**

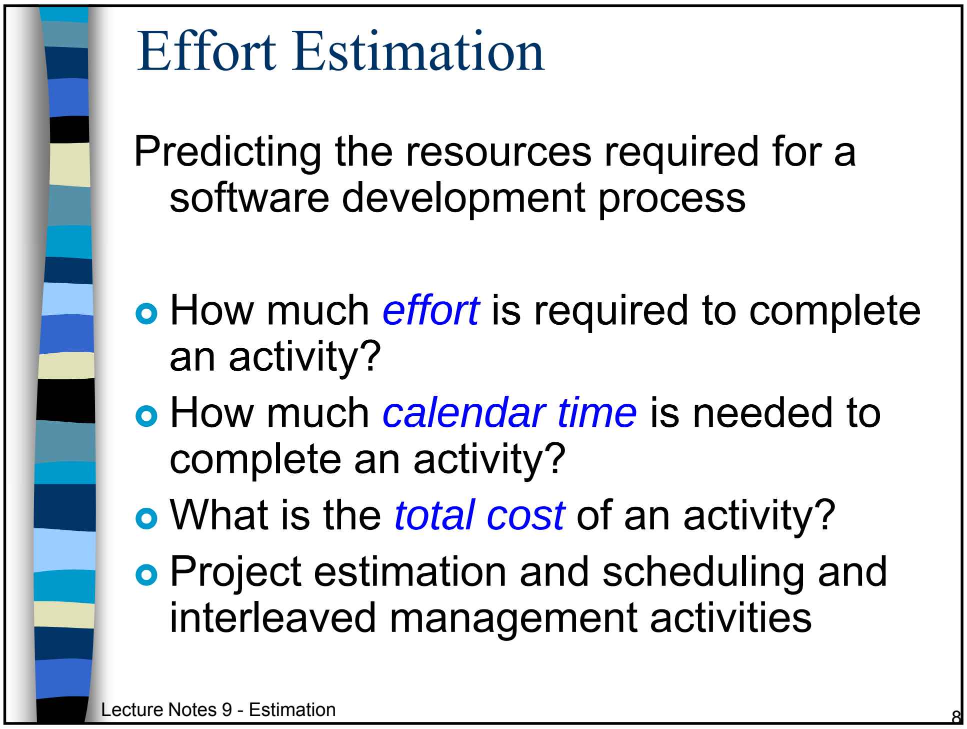


The Mythical Man-Month

“Most software projects have gone awry for lack of calendar time than for all other causes combined.

Why is this cause of disaster so common?”

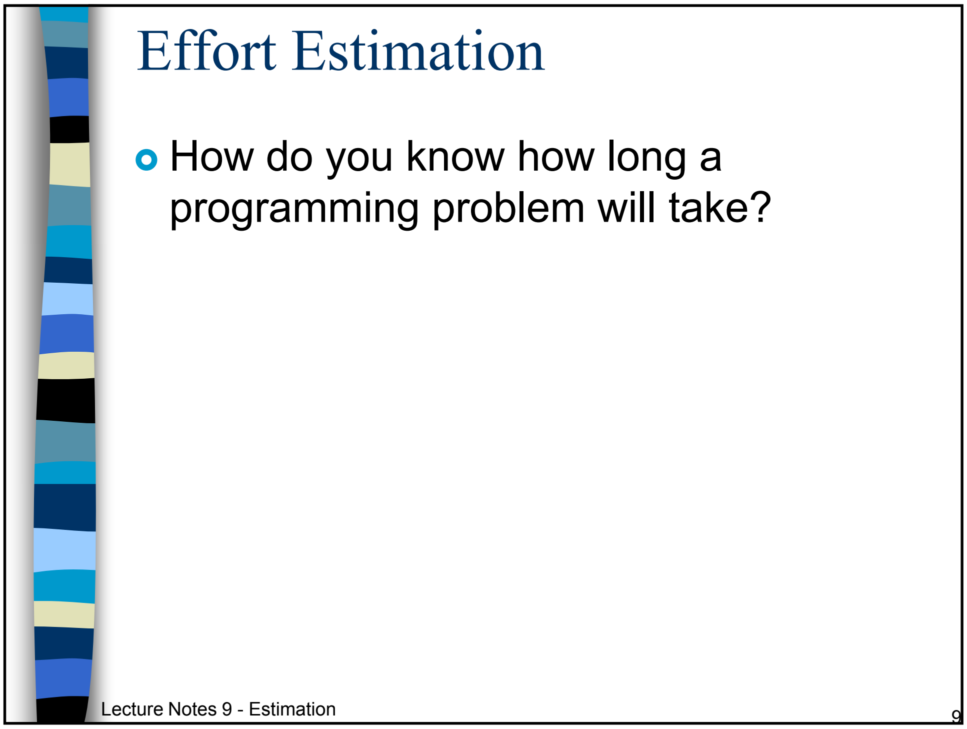
Brooks p. 14



Effort Estimation

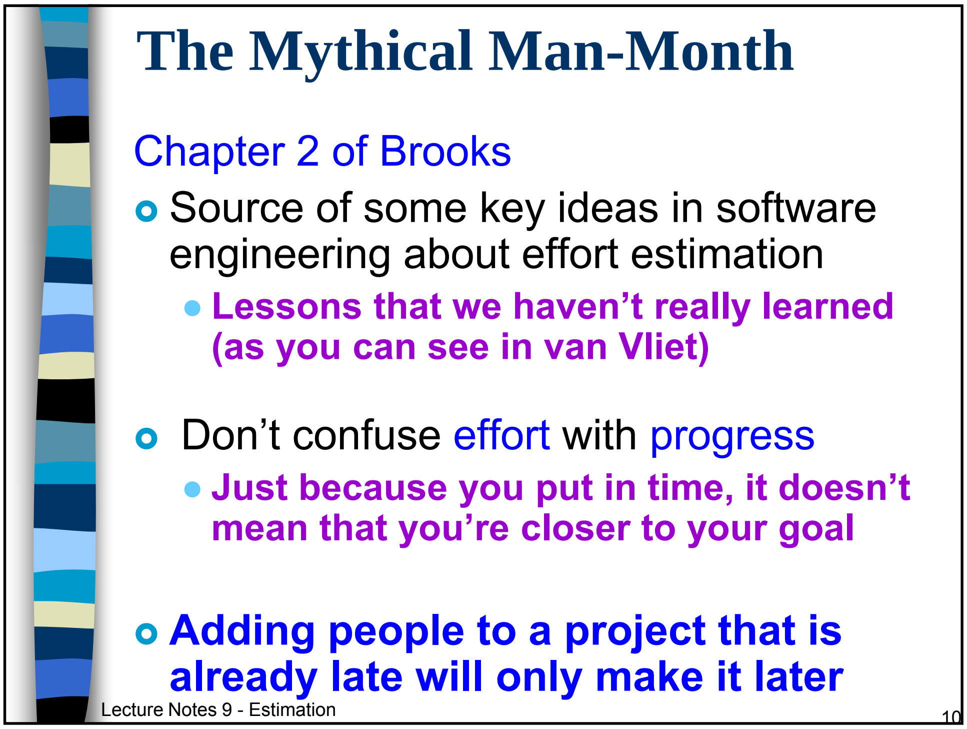
Predicting the resources required for a software development process

- How much *effort* is required to complete an activity?
- How much *calendar time* is needed to complete an activity?
- What is the *total cost* of an activity?
- Project estimation and scheduling and interleaved management activities



Effort Estimation

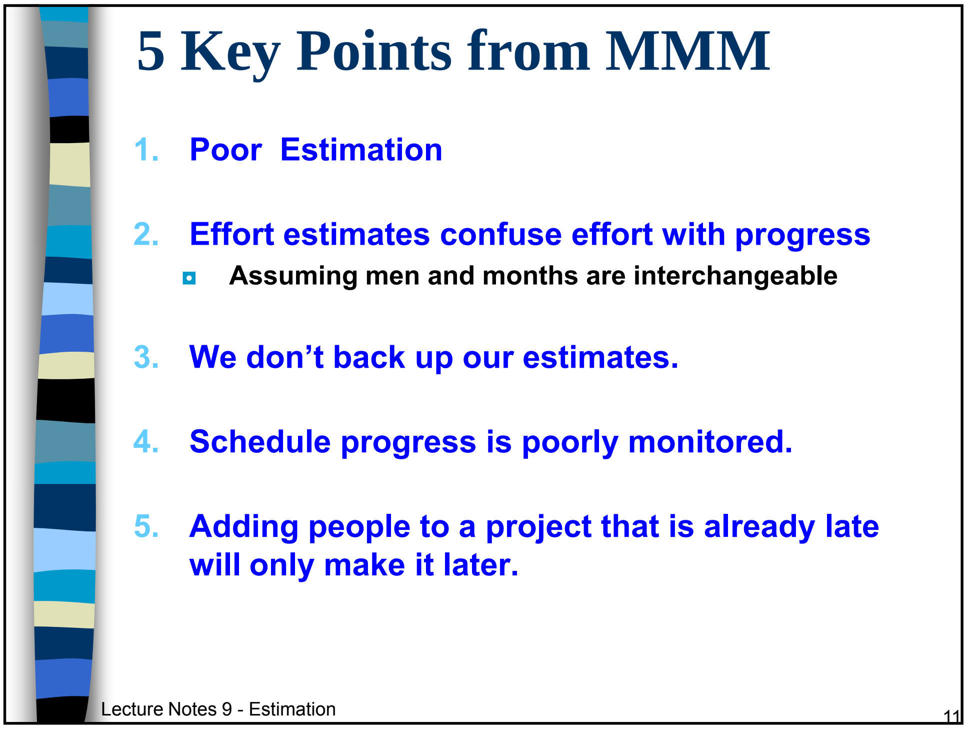
- How do you know how long a programming problem will take?



The Mythical Man-Month

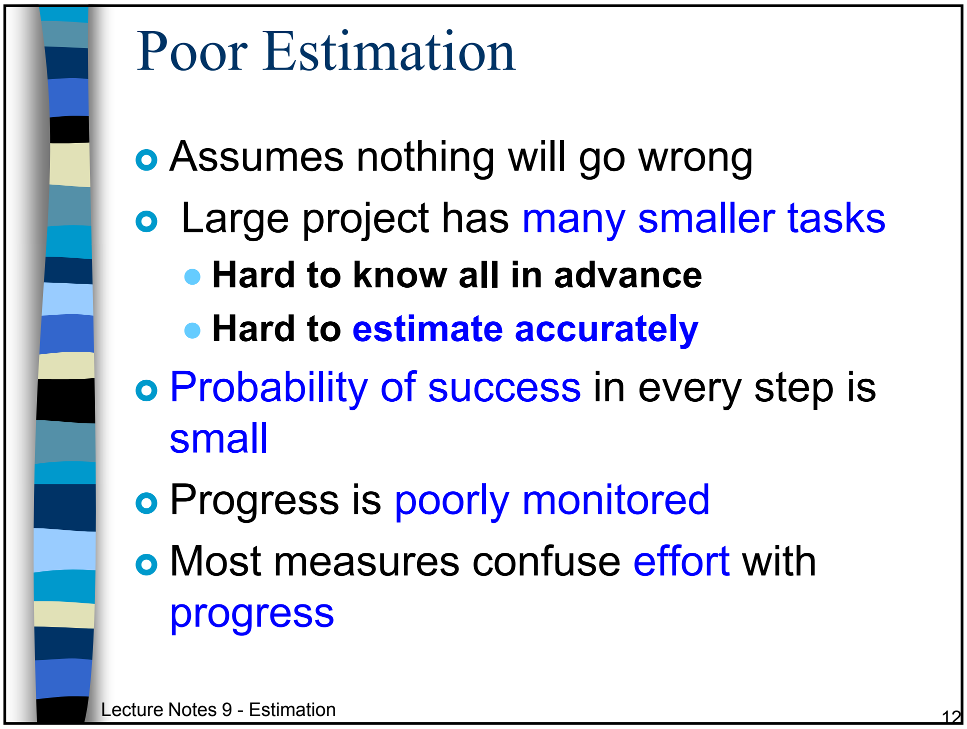
Chapter 2 of Brooks

- Source of some key ideas in software engineering about effort estimation
  - **Lessons that we haven't really learned (as you can see in van Vliet)**
- Don't confuse effort with progress
  - **Just because you put in time, it doesn't mean that you're closer to your goal**
- **Adding people to a project that is already late will only make it later**



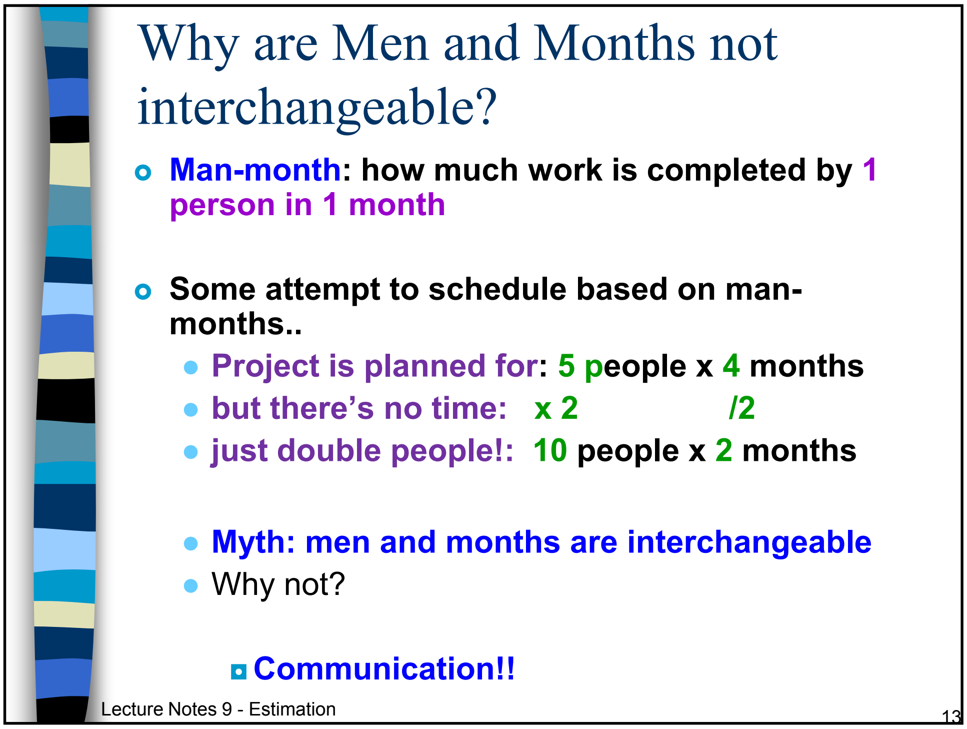
5 Key Points from MMM

1. Poor Estimation
2. Effort estimates confuse effort with progress
   - Assuming men and months are interchangeable
3. We don’t back up our estimates.
4. Schedule progress is poorly monitored.
5. Adding people to a project that is already late will only make it later.



Poor Estimation

- Assumes nothing will go wrong
- Large project has many smaller tasks
  - **Hard to know all in advance**
  - **Hard to estimate accurately**
- Probability of success in every step is small
- Progress is poorly monitored
- Most measures confuse effort with progress

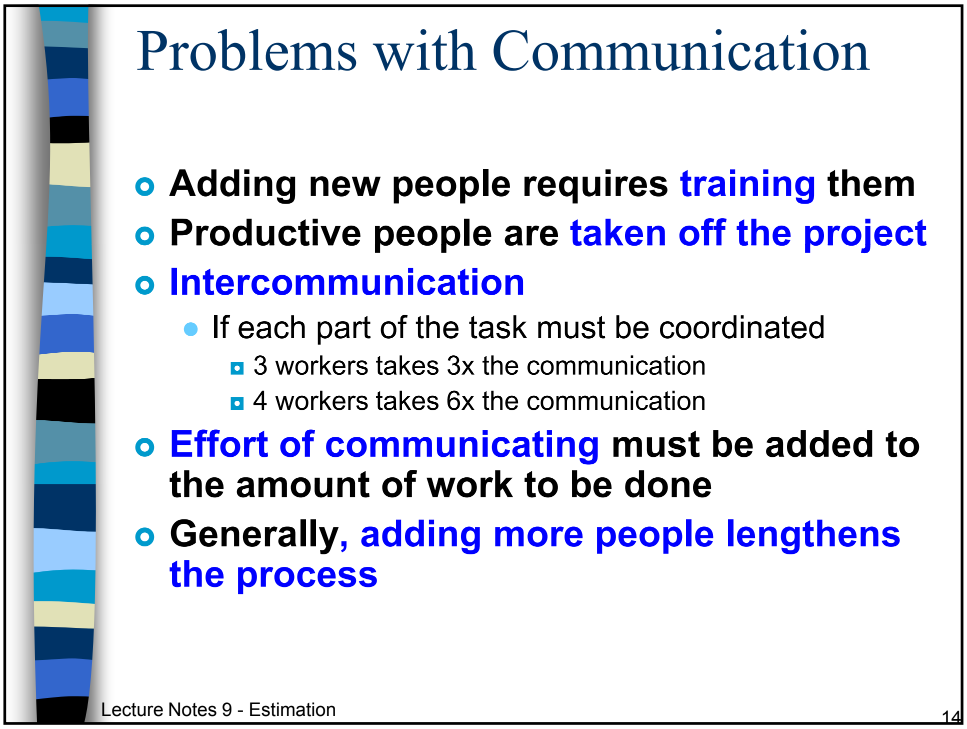


Why are Men and Months not interchangeable?

- **Man-month: how much work is completed by 1 person in 1 month**

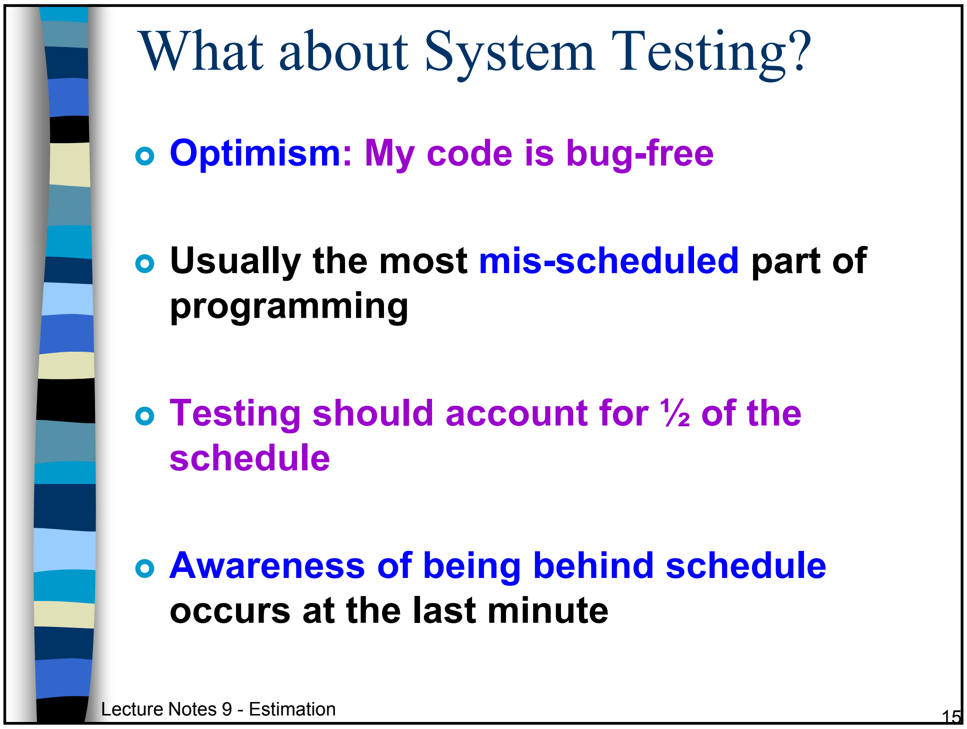
- **Some attempt to schedule based on man-months..**
  - **Project is planned for: 5 people x 4 months**
  - **but there's no time: x 2   /2**
  - **just double people!: 10 people x 2 months**

  - **Myth: men and months are interchangeable**
  - Why not?
    - **Communication!!**



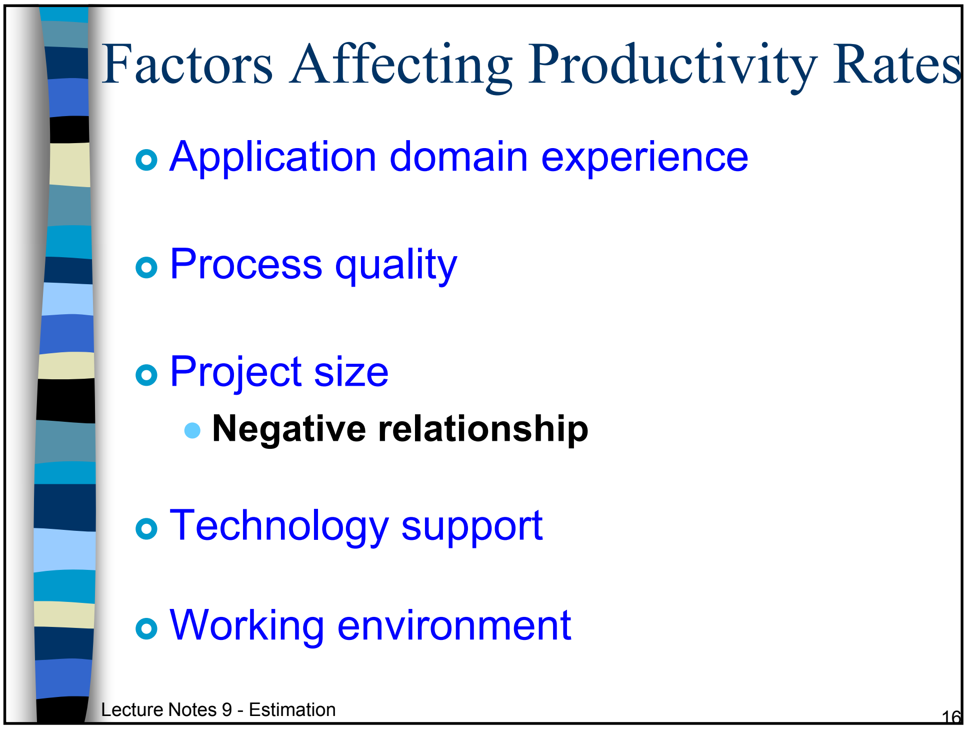
Problems with Communication

- **Adding new people requires training them**
- **Productive people are taken off the project**
- **Intercommunication**
  - If each part of the task must be coordinated
    - 3 workers takes 3x the communication
    - 4 workers takes 6x the communication
- **Effort of communicating must be added to the amount of work to be done**
- **Generally, adding more people lengthens the process**



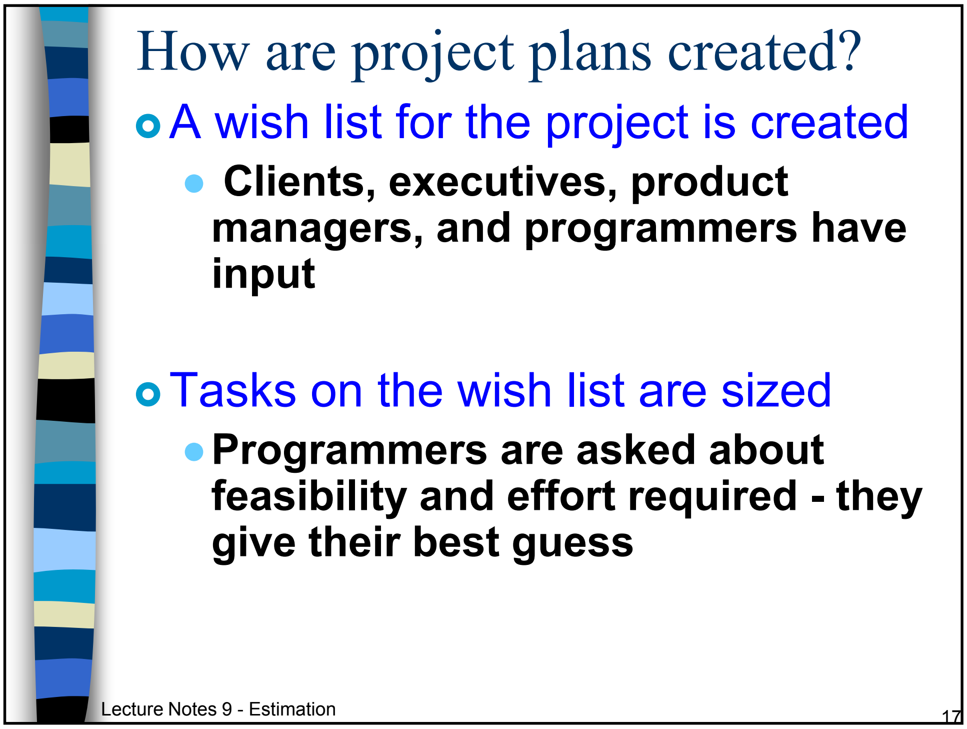
What about System Testing?

- **Optimism: My code is bug-free**
- **Usually the most mis-scheduled part of programming**
- **Testing should account for ½ of the schedule**
- **Awareness of being behind schedule occurs at the last minute**



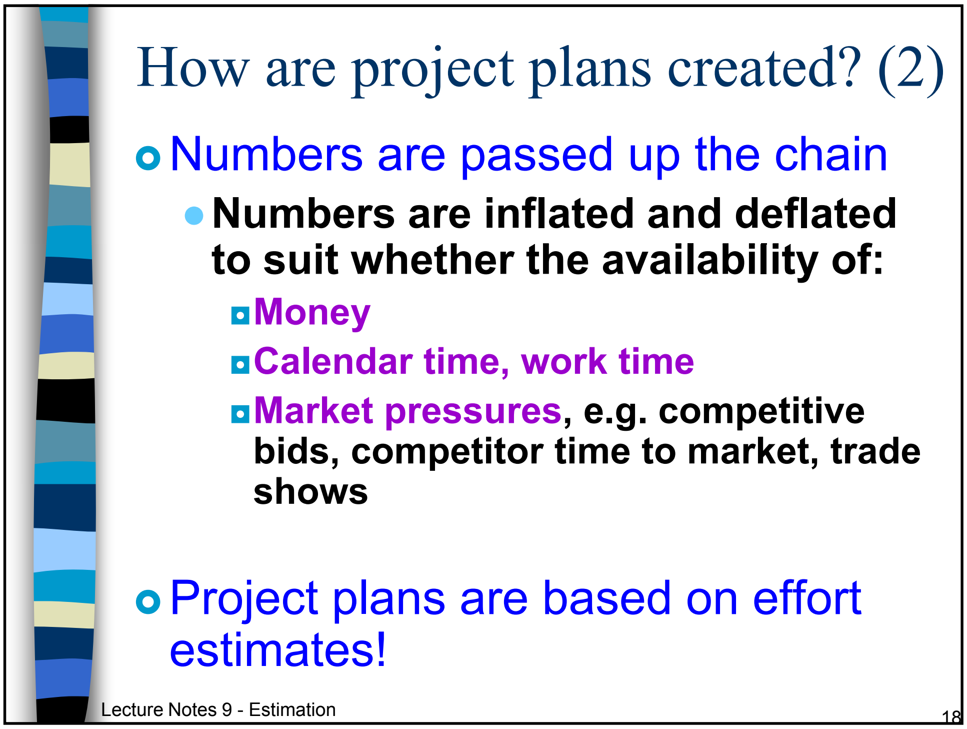
Factors Affecting Productivity Rates

- Application domain experience
- Process quality
- Project size
  - **Negative relationship**
- Technology support
- Working environment



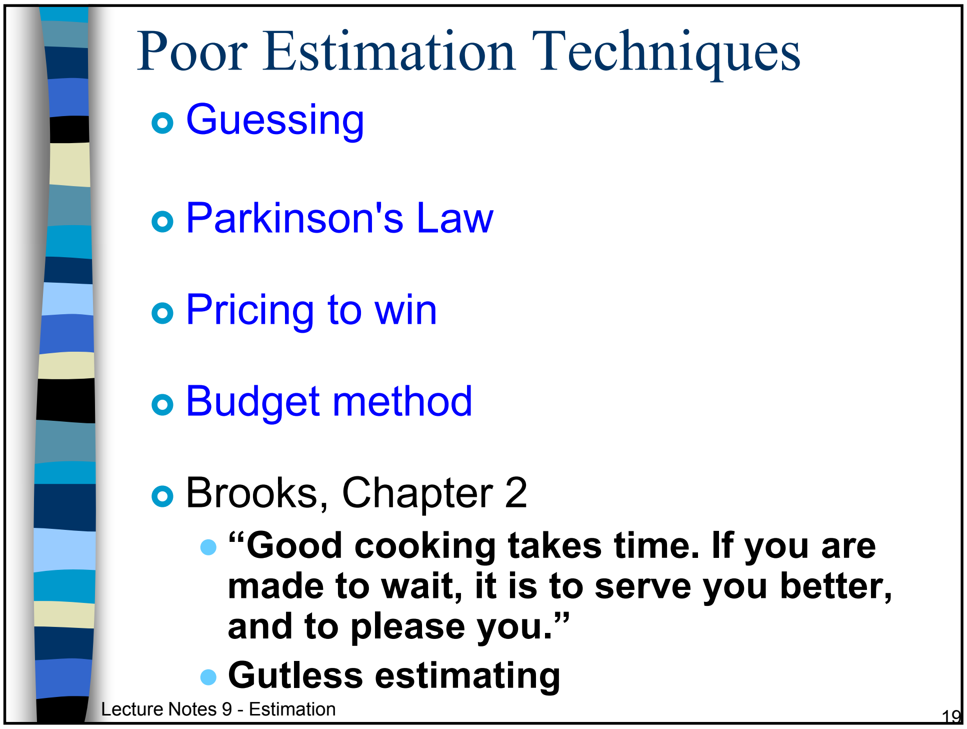
How are project plans created?

- A wish list for the project is created
  - Clients, executives, product managers, and programmers have input
- Tasks on the wish list are sized
  - Programmers are asked about feasibility and effort required - they give their best guess



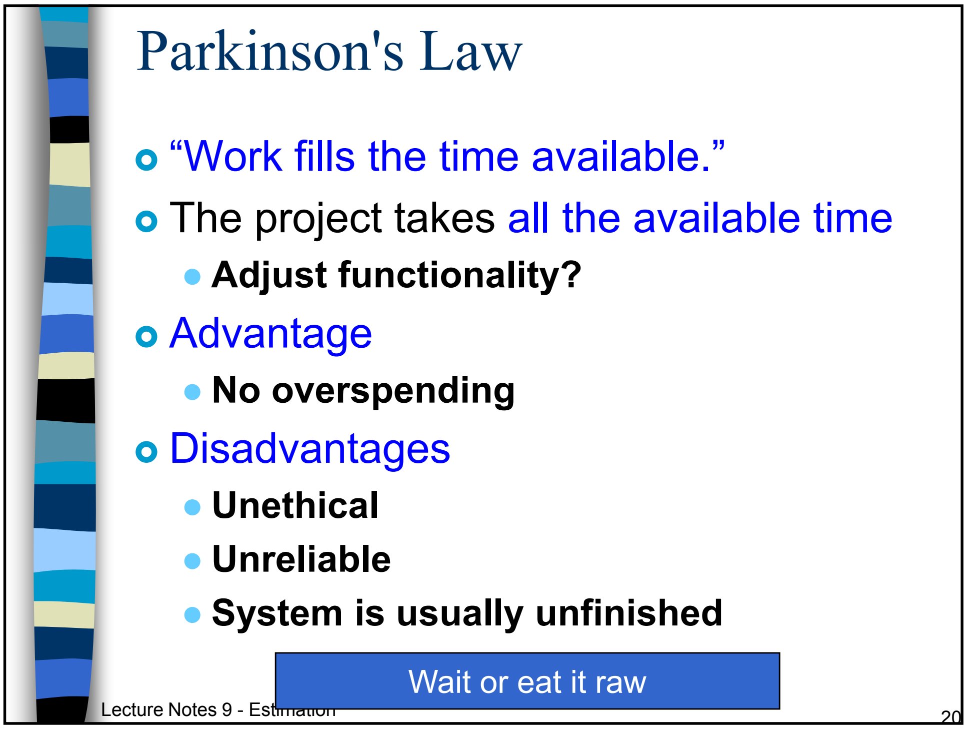
How are project plans created? (2)

- Numbers are passed up the chain
  - Numbers are inflated and deflated to suit whether the availability of:
    - Money
    - Calendar time, work time
    - Market pressures, e.g. competitive bids, competitor time to market, trade shows
- Project plans are based on effort estimates!



Poor Estimation Techniques

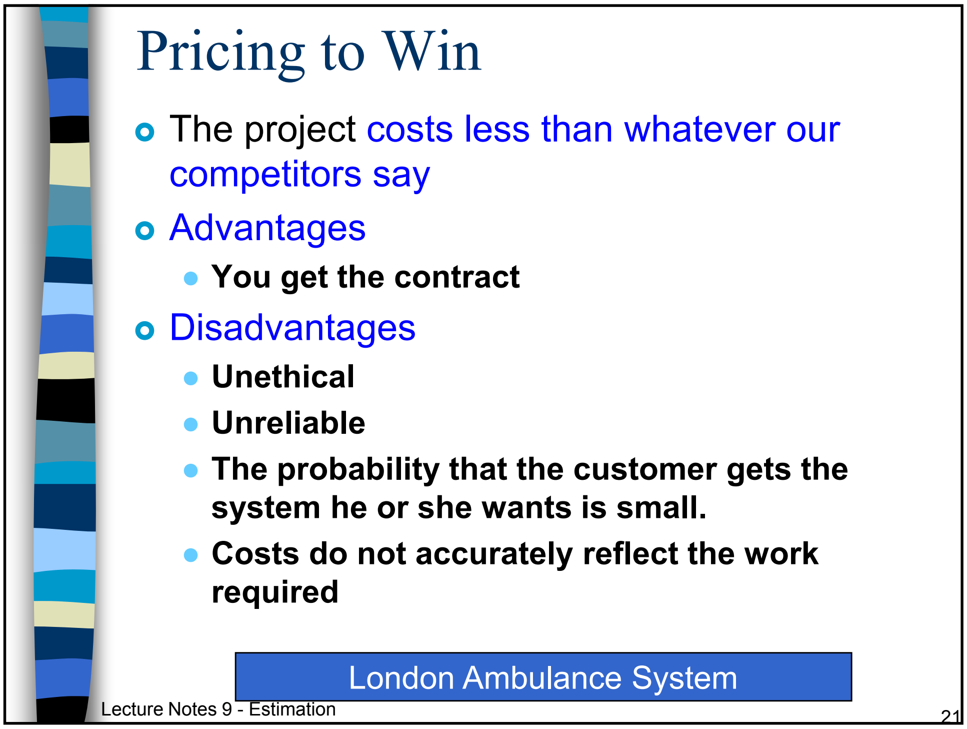
- Guessing
- Parkinson's Law
- Pricing to win
- Budget method
- Brooks, Chapter 2
  - **“Good cooking takes time. If you are made to wait, it is to serve you better, and to please you.”**
  - **Gutless estimating**



Parkinson's Law

- “Work fills the time available.”
- The project takes all the available time
  - **Adjust functionality?**
- Advantage
  - **No overspending**
- Disadvantages
  - **Unethical**
  - **Unreliable**
  - **System is usually unfinished**

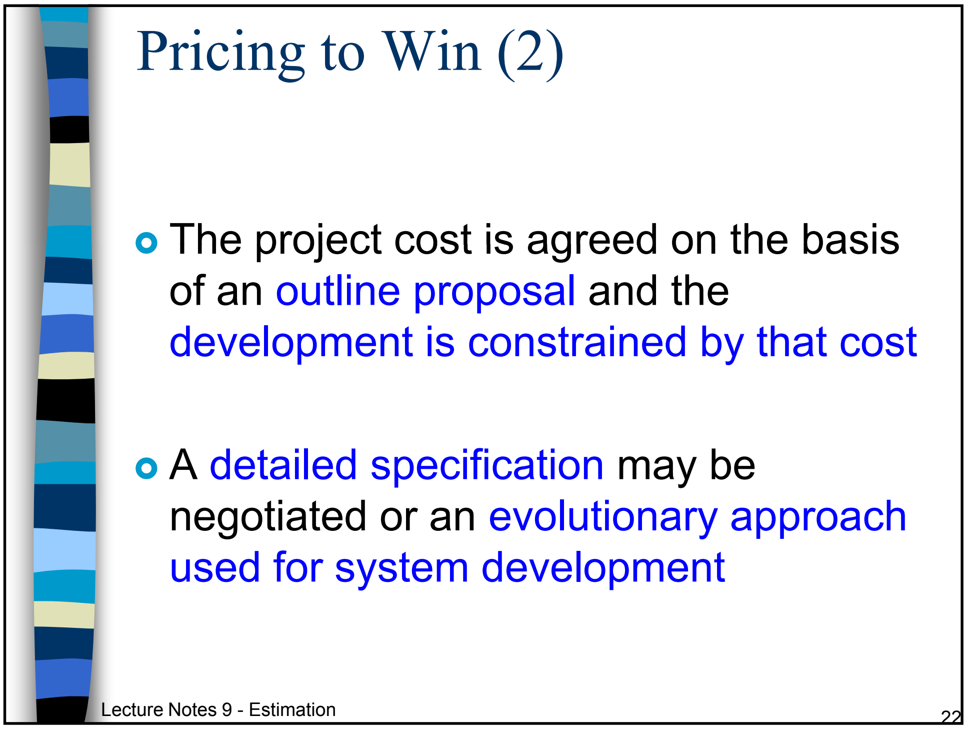
Wait or eat it raw



Pricing to Win

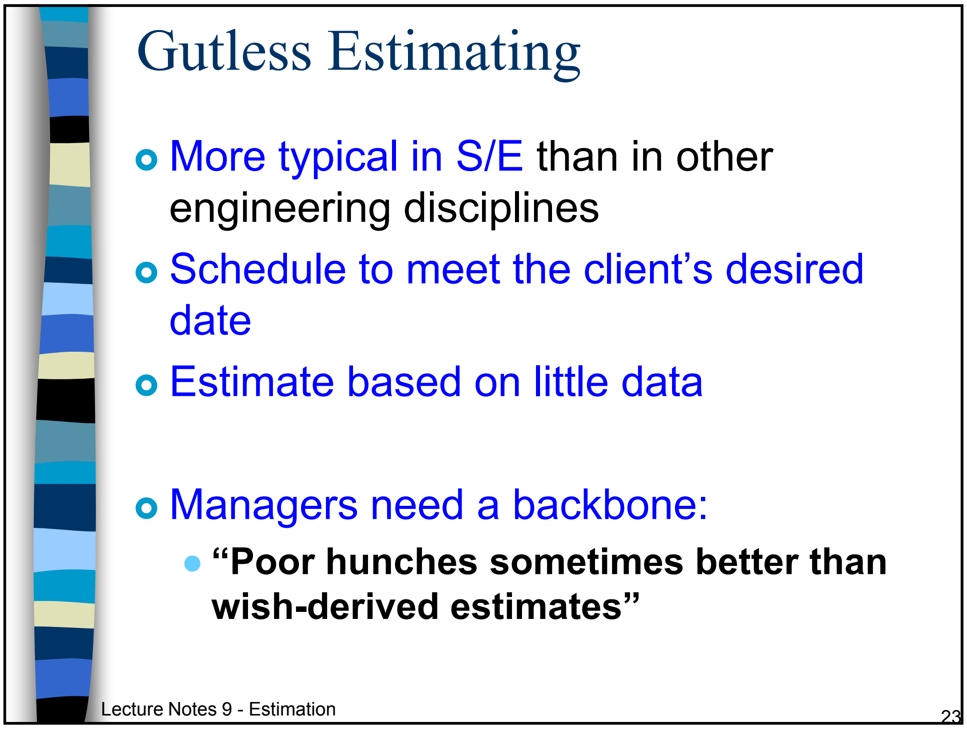
- The project costs less than whatever our competitors say
- Advantages
  - You get the contract
- Disadvantages
  - Unethical
  - Unreliable
  - The probability that the customer gets the system he or she wants is small.
  - Costs do not accurately reflect the work required

London Ambulance System



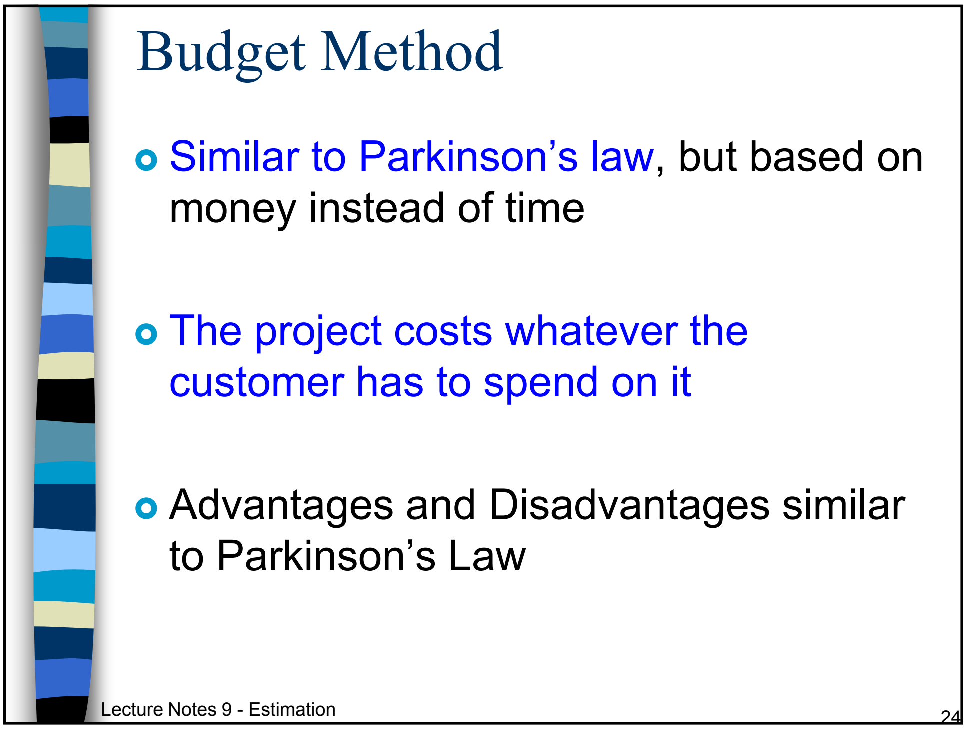
Pricing to Win (2)

- The project cost is agreed on the basis of an outline proposal and the development is constrained by that cost
- A detailed specification may be negotiated or an evolutionary approach used for system development



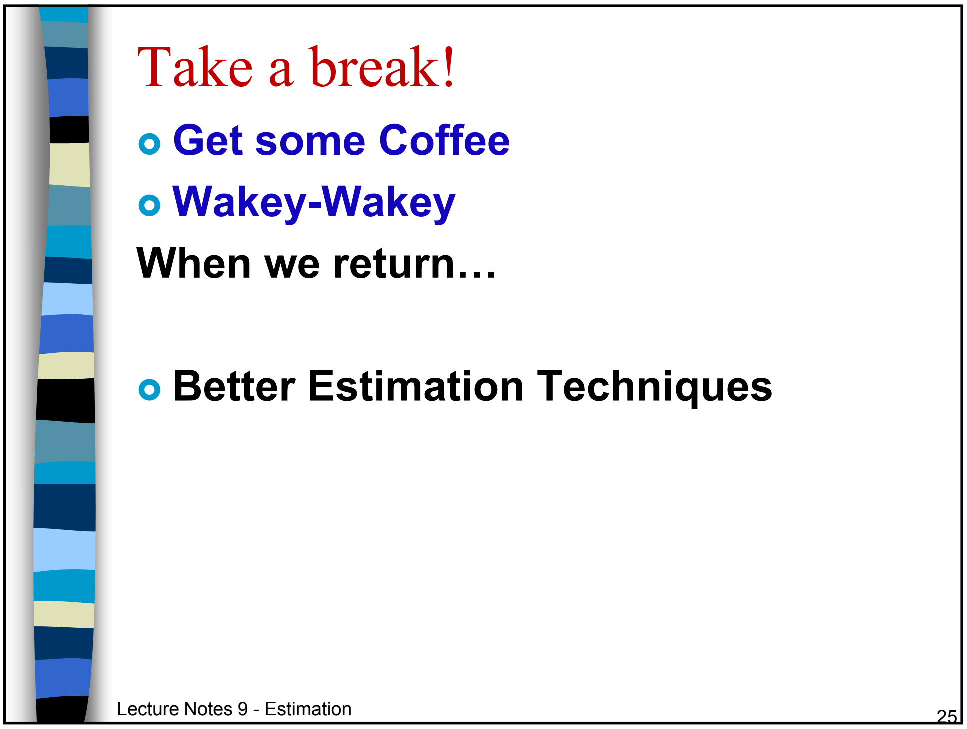
Gutless Estimating

- More typical in S/E than in other engineering disciplines
- Schedule to meet the client’s desired date
- Estimate based on little data
- Managers need a backbone:
  - **“Poor hunches sometimes better than wish-derived estimates”**



Budget Method

- Similar to Parkinson’s law, but based on money instead of time
- The project costs whatever the customer has to spend on it
- Advantages and Disadvantages similar to Parkinson’s Law

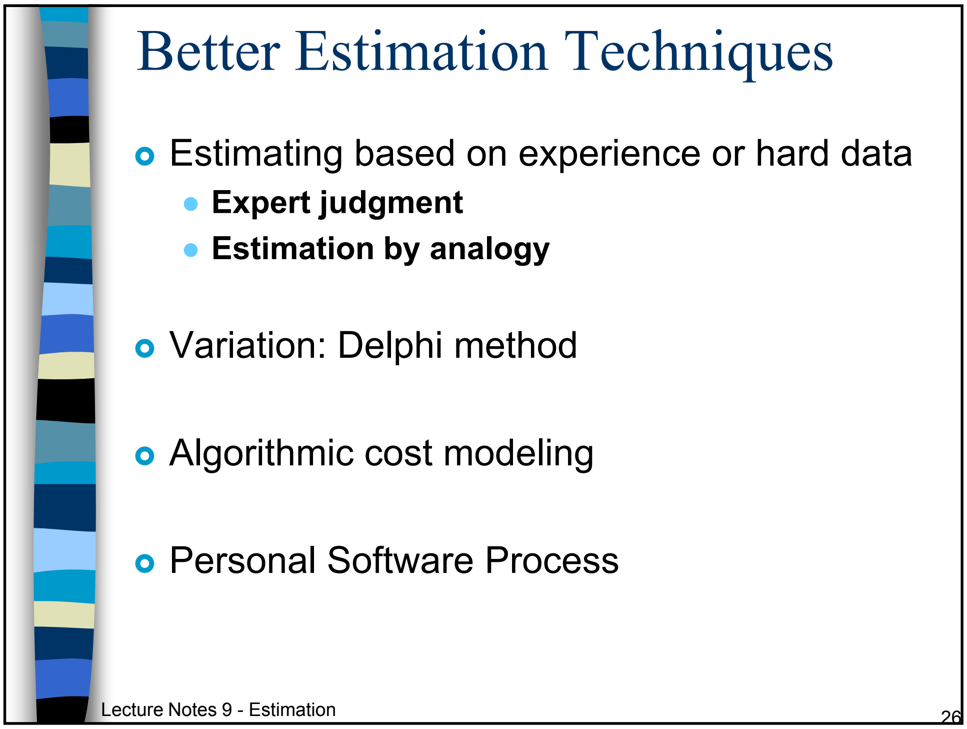


Take a break!

- Get some Coffee
- Wakey-Wakey

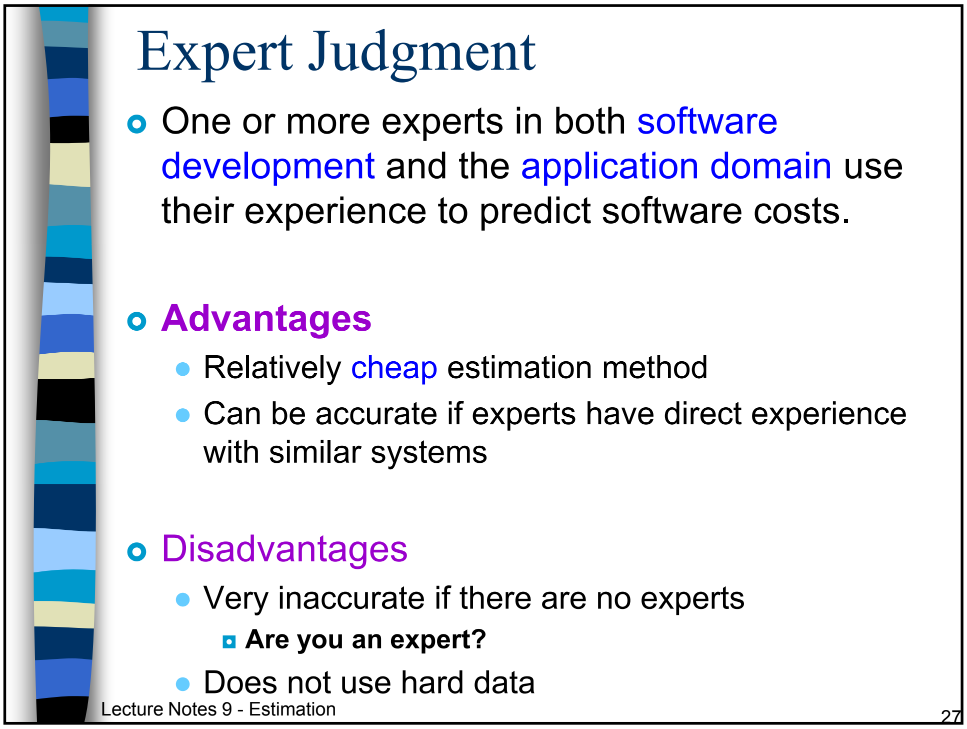
When we return…

- **Better Estimation Techniques**



Better Estimation Techniques

- Estimating based on experience or hard data
  - **Expert judgment**
  - **Estimation by analogy**
- Variation: Delphi method
- Algorithmic cost modeling
- Personal Software Process

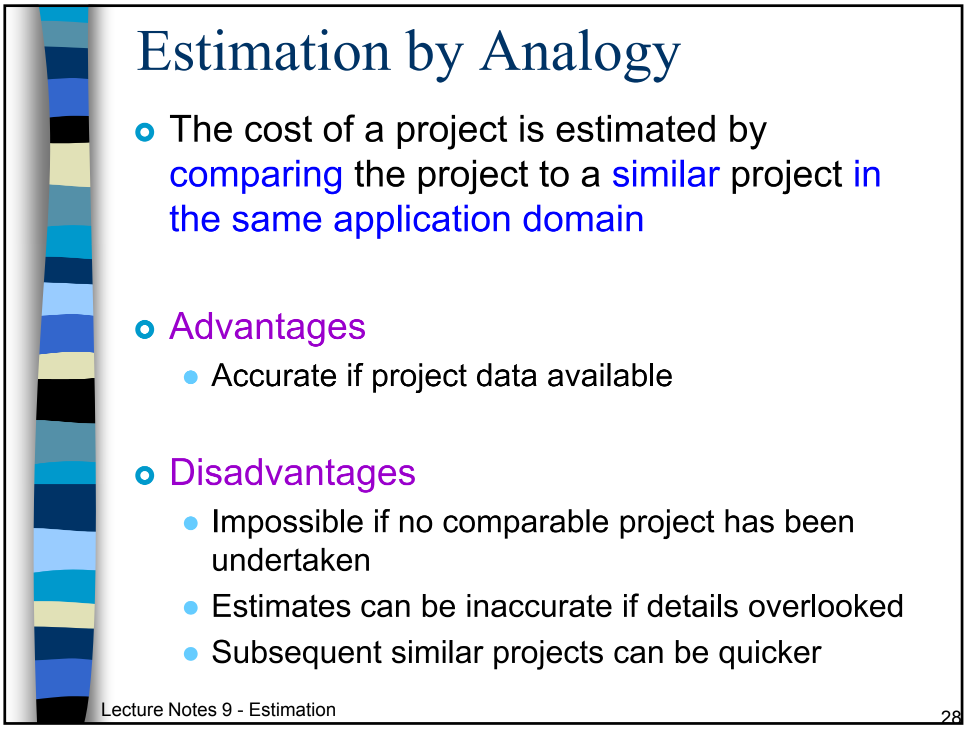


Expert Judgment

- One or more experts in both software development and the application domain use their experience to predict software costs.

- **Advantages**
  - Relatively cheap estimation method
  - Can be accurate if experts have direct experience with similar systems

- Disadvantages
  - Very inaccurate if there are no experts
    - **Are you an expert?**
  - Does not use hard data

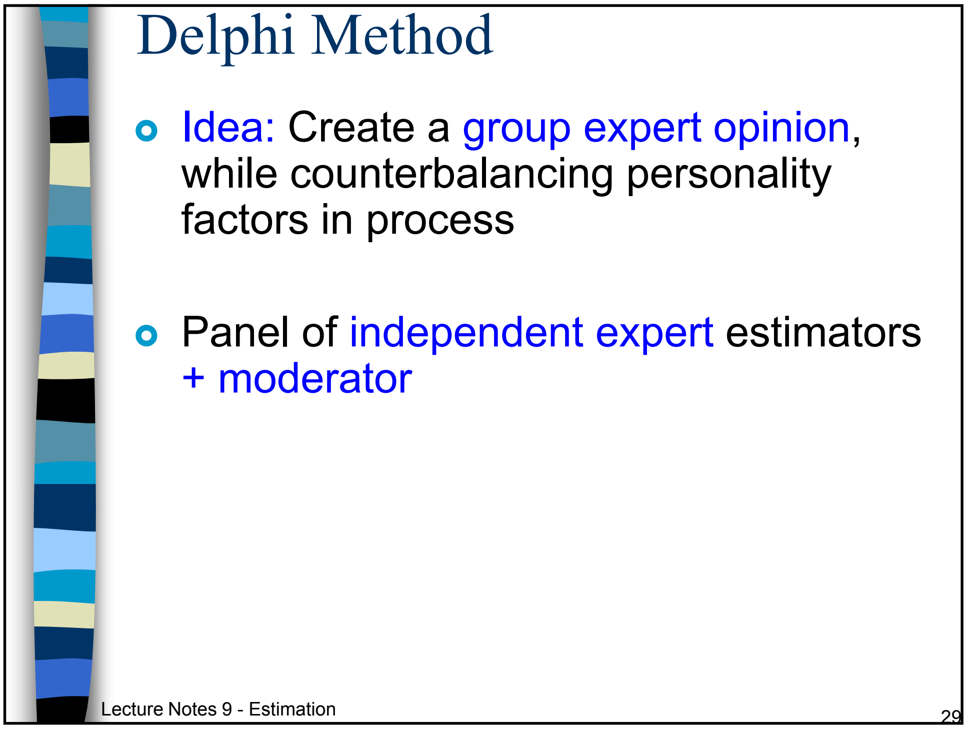


Estimation by Analogy

- The cost of a project is estimated by comparing the project to a similar project in the same application domain

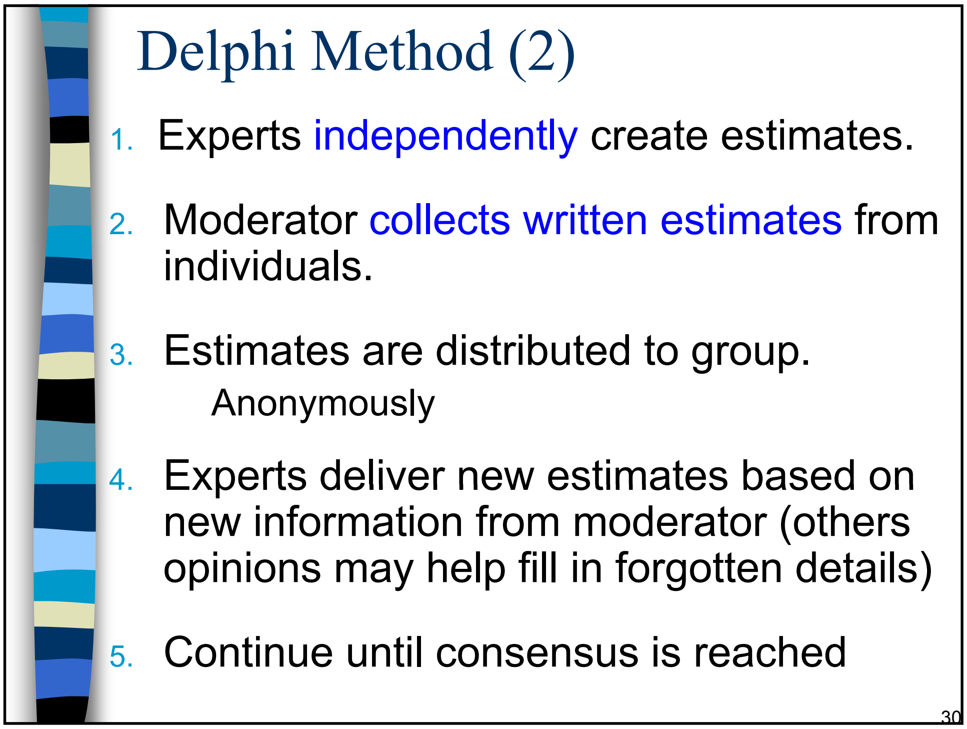
- Advantages
  - Accurate if project data available

- Disadvantages
  - Impossible if no comparable project has been undertaken
  - Estimates can be inaccurate if details overlooked
  - Subsequent similar projects can be quicker



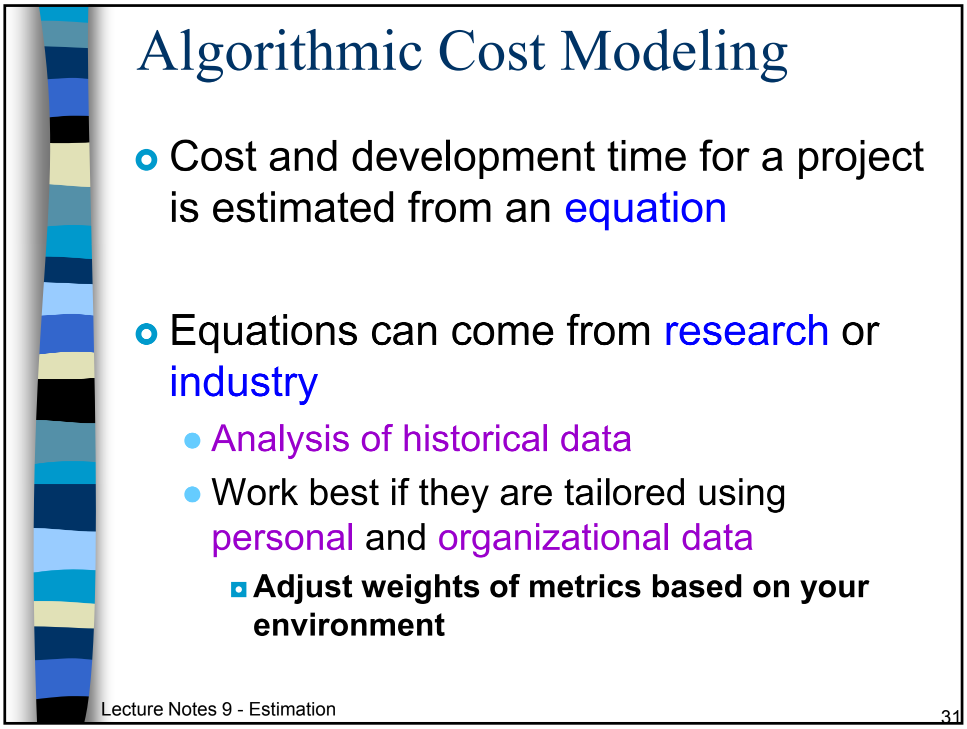
Delphi Method

- Idea: Create a group expert opinion, while counterbalancing personality factors in process
- Panel of independent expert estimators + moderator



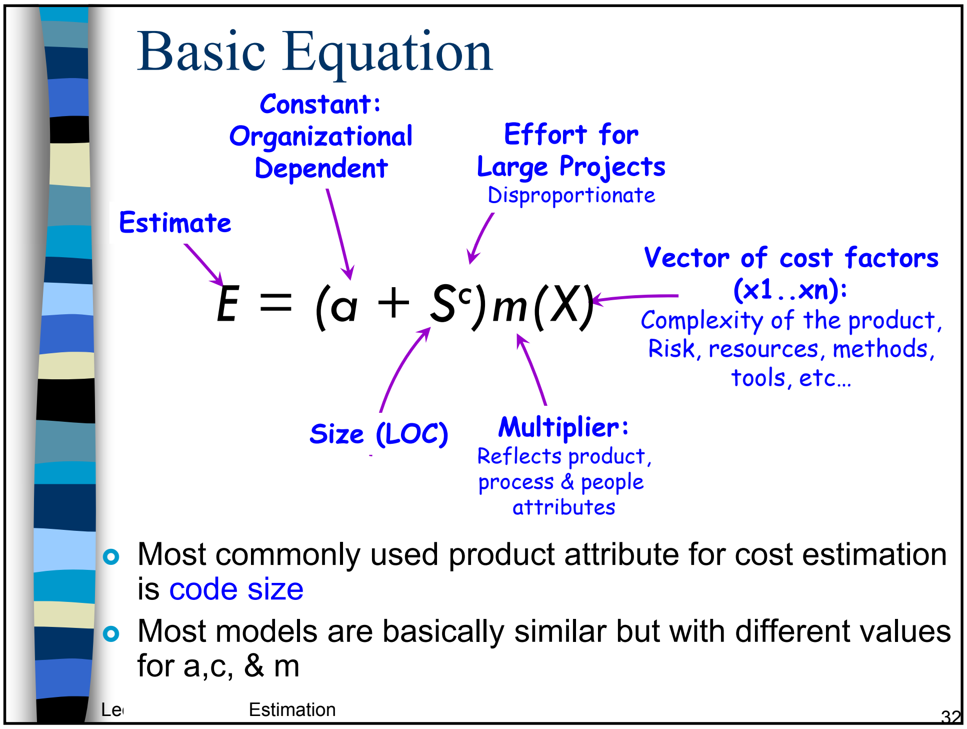
Delphi Method (2)

1. Experts independently create estimates.
2. Moderator collects written estimates from individuals.
3. Estimates are distributed to group.
   - Anonymously
4. Experts deliver new estimates based on new information from moderator (others opinions may help fill in forgotten details)
5. Continue until consensus is reached



Algorithmic Cost Modeling

- Cost and development time for a project is estimated from an equation
- Equations can come from research or industry
  - Analysis of historical data
  - Work best if they are tailored using personal and organizational data
    - **Adjust weights of metrics based on your environment**



Basic Equation

$$E = (a + S^c)m(X)$$

Constant: Organizational Dependent

Effort for Large Projects
Disproportionate

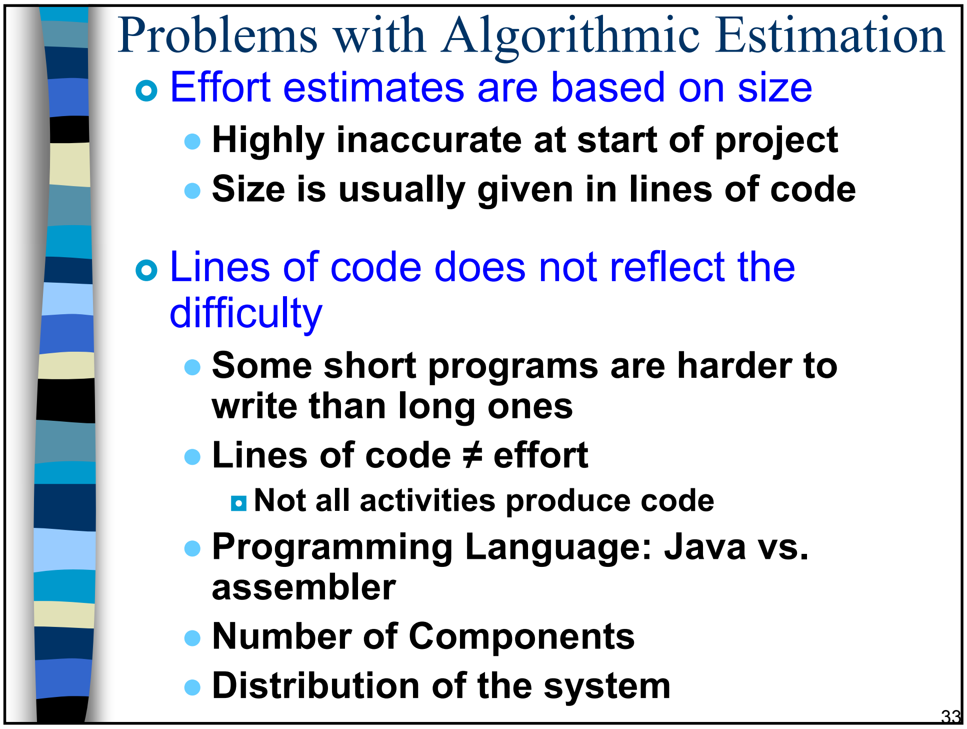
Estimate

Vector of cost factors (x1..xn):
Complexity of the product, Risk, resources, methods, tools, etc…

Size (LOC)

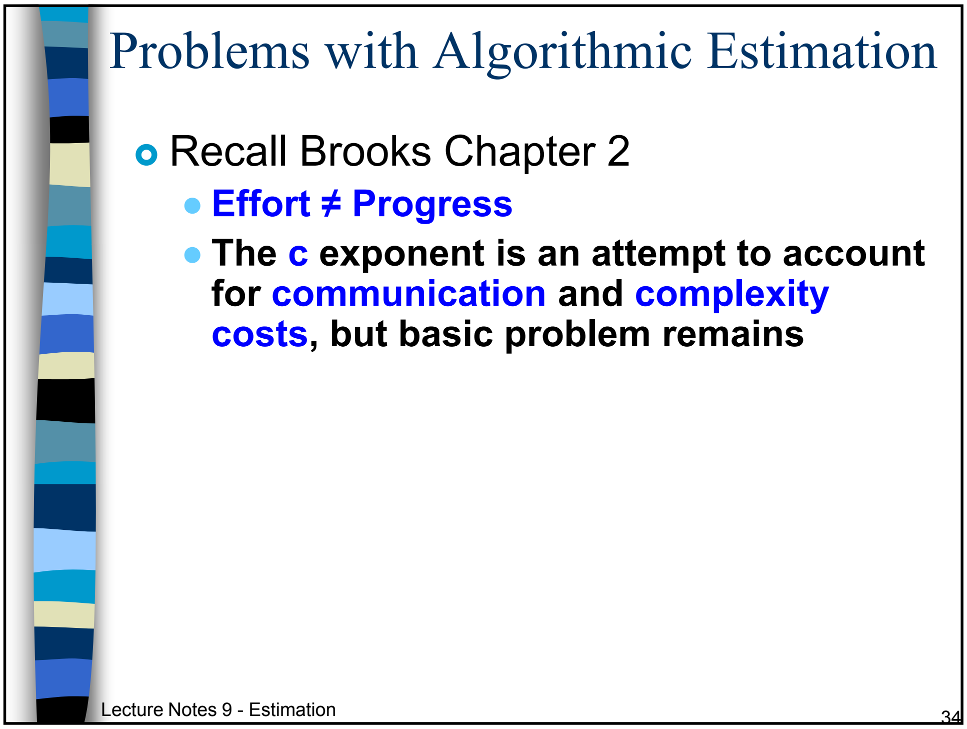
Multiplier:
Reflects product, process & people attributes

- Most commonly used product attribute for cost estimation is code size
- Most models are basically similar but with different values for a,c, & m



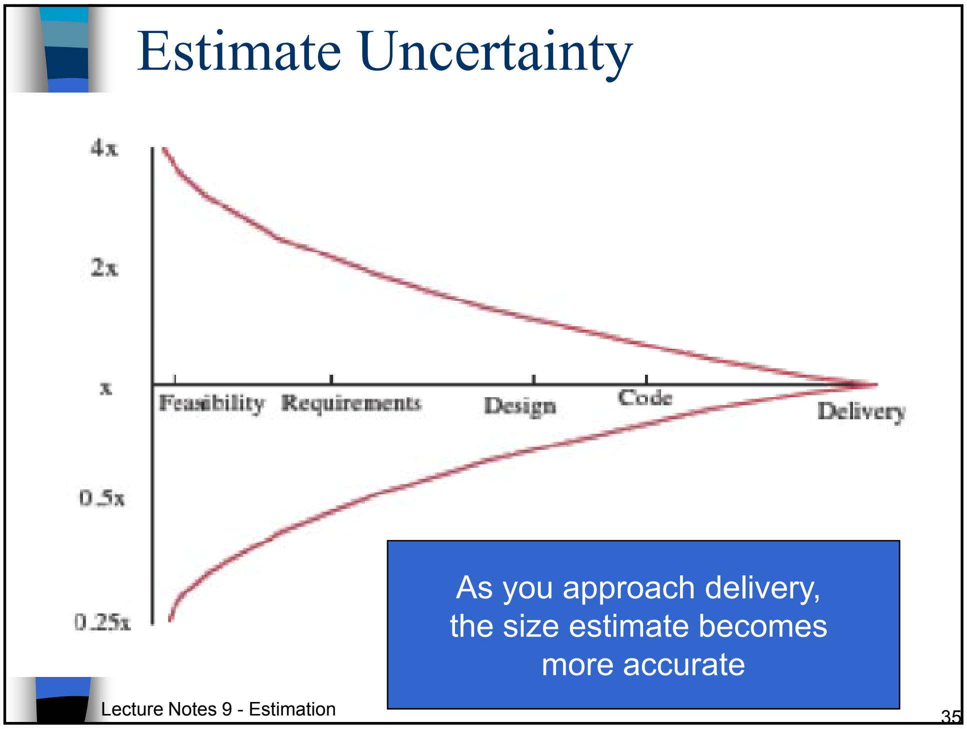
Problems with Algorithmic Estimation

- Effort estimates are based on size
  - **Highly inaccurate at start of project**
  - **Size is usually given in lines of code**
- Lines of code does not reflect the difficulty
  - **Some short programs are harder to write than long ones**
  - **Lines of code ≠ effort**
    - **Not all activities produce code**
  - **Programming Language: Java vs. assembler**
  - **Number of Components**
  - **Distribution of the system**



Problems with Algorithmic Estimation

- Recall Brooks Chapter 2
  - **Effort ≠ Progress**
  - **The c exponent is an attempt to account for communication and complexity costs, but basic problem remains**



Estimate Uncertainty

As you approach delivery, the size estimate becomes more accurate