INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Fall Quarter, 2007
Michele Rousseau
Set 21

Announcements

- **Quiz #4 will be on Wednesday after the holidays 11/28**
  - It will cover Lectures from 11/14-11/26 (sets 17-21)
  - Readings not covered on previous quizzes
- **Readings:**
  - Van Vliet Chapter 7

Previously in INF 111…

- **UML**
  - State Transition Diagrams
  - Activity Diagrams

Review

Review

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 we will 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.

Topic 21 10

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

Topic 21 11

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!!**

Topic 21 12

Problems with Communication

- **Adding new people requires training them**
- **Productive people are taking 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**
- 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

- 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

Better Estimation Techniques

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

1. Experts independently create estimates.
2. Moderator collects written estimates from individuals.
3. Estimates are distributed to group.
   - **No names**
4. Experts deliver new estimates based on new information from moderator.
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

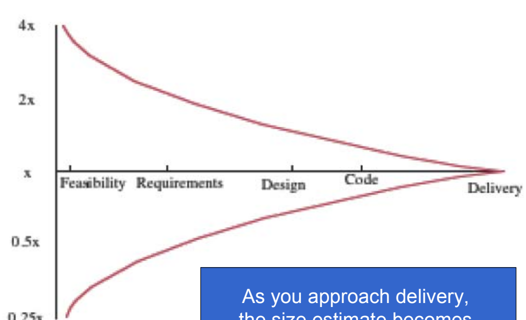
$$E = (a + S^{C})m(X)$$

- S is estimated size of the systems (LOC)
- a, c, m, are constants
  - **a is an organization-dependent constant**
  - **c reflects the disproportionate effort for large projects**
  - **m is a multiplier reflecting product, process and people attributes**
- X is a vector of cost factors, x1…xn
  - **Complexity of product, risk, resources, process, people attributes, methods**
- Most commonly used product attribute for cost estimation is code size
- Most models are basically similar but with different values for A, B and 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**
- Recall Brooks Chapter 2
  - **Effort ≠ progress**
  - **The B 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

COCOMO

- **COCOMO is one of the most widely used software estimation models in the world**
- **It was developed by Barry Boehm in 1981**
- **COCOMO predicts the effort and schedule for a software product development based on inputs relating to the** size **of the software and a number of** cost drivers **that affect productivity**

Boehm: COCOMO

Constructive Cost Model (COCOMO)

- COCOMO is one of the most widely used software estimation models in the world
- An empirical model based on project experience
- Well-documented, 'independent' model which is not tied to a specific software vendor
- Long history from initial version published in 1981 (COCOMO-81)
- COCOMO II takes into account different approaches to software development, reuse, etc.
- predicts the effort and schedule for a software product development based on inputs relating to the **size** of the software and a number of **cost drivers** that affect productivity

COCOMO: Three Models

- **3 Models reflect the** complexity**:**
  - the Basic Model
  - the Intermediate Model
  - and the Detailed Model

The Development Modes: Project Characteristics

- Organic Mode
  - developed in a **familiar**, stable environment,
  - **similar** to the previously developed projects
  - relatively **small** and requires little innovation
  - Eg. Payroll system
- Semidetached Mode
  - **intermediate** between Organic and Embedded
  - Eg. Banking System
- Embedded Mode
  - tight, **inflexible** constraints and interface requirements
  - The product requires **great innovation**
  - Eg. Nuclear power plant system

COCOMO: Some Assumptions

- Primary cost driver is the number of Delivered Source Instructions (DSI) developed by the project
- COCOMO estimates assume that the project will enjoy good management by both the developer and the customer
- Assumes the requirements specification is not substantially changed after the plans and requirements phase

Basic COCOMO Model: When Should You Use It

- Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs

Basic COCOMO Model: Limitations

- Its accuracy is necessarily limited because of its lack of factors which have a significant influence on software costs
- The Basic COCOMO estimates are within a factor of 1.3 only 29% of the time, and within a factor of 2 only 60% of the time

Basic COCOMO Model: An Example

- We have determined our project fits the characteristics of Semi-Detached mode
- We estimate our project will have 32,000 Delivered Source Instructions. Using the formulas, we can estimate:
- Effort = $3.0*(32)^{1.12}$ = 146 man-months
- Schedule = $2.5*(146)^{0.35}$ = 14 months
- Productivity = 32,000 DSI / 146 MM = 219 DSI/MM
- Average Staffing = 146 MM /14 months = 10 FSP

Data Collection

- Regardless of the method or model used, data is needed for calibration
- Programmers need to know their own “constant adjustment factors”
  - **Goal of Personal Software Process to establish such a database**

Comparison of Basic Formula

	Halstead	Boehm	Walston-Felix
KLOC	$E=0.7\ KLOC^{1.50}$	$E=2.4\ KLOC^{1.05}$	$E=5.2\ KLOC^{0.91}$
1	0.7	2.4	5.2
10	22.1	26.9	42.3
50	247.5	145.9	182.8
100	700.0	302.1	343.6
1000	22135.9	3390.1	2792.6

- Coefficients derived using actual project data
  - **Variability in project characteristics**
- At best, yield estimates that are at most 25% off, 75% of the time, *for projects used to derive the model*.

So, what can you do?

- You
  - **Don't have a historical database**
  - **Are not an expert**
- Generate estimates using multiple models and compare based on your guesses or assumptions
  - **Similar to using the models as your personal experts in Delphi method**
  - **Candidate models:**
    - **Walston and Felix (simple and easy to use)**
    - **COCOMO 2 (complicated and detailed)**
    - **DeMarco (based on UI requirements)**
- Brooks, p. 20
  - **1/3 planning, 1/6 coding, 1/4 component tests and early system test, 1/4 system test**