Software Project Management

CS6704: Class 7

Instructor: Shawn A. Bohner
Voice: (703) 538-8374
Email: sbohner@vt.edu

Teaching Assistant: Yunxian Zhou
Email: yxzhou@nvc.cs.vt.edu
Voice: (703) 538-8381

Agenda

- Review Last Week’s Material
  - Turn in Homework
  - Reading Discussion
  - Review last week’s class
- Review of Class for Midterm Preparation
  - Break
- More Review of Class
- Homework Assignment
Fall Semester Timeline

Class Begins
PM Basics
Managing with Metrics
Emerging PM Paradigms

Mid-Term Exam
Program Management
Final Exam

Aug − Sept − Oct − Nov − DEC

Software Project Planning
Software Estimation
Risk Management
Project Portfolio Management

Software Process and Life Cycle
Human Side of PM

Reading Discussions...

“Using Metrics to Manage Software Projects” by Edward Weller
- What do you think of Weller’s Levels 1-3
- Size, Defects, Effort, Schedule – What else?
- Planned Estimates vs. Tracked Actuals – Why?
- What goals do you think this focuses on?

“What are Function Points” by Capers Jones
- Would you use FPs on your project? Why?
- When are FP estimates the most effective?
- If LOC in not important, why is language an issue with FPs?
Reading Discussions...

- “A Practical View of SW Measurement…” by M. Daskalantonakis
  - Why use a goal-oriented metrics approach?
  - What are some example goals used?
  - Before a metrics program can be effective, what systems must be in place?
  - What’s a MUG? What other infrastructure parts?

- “Evolutionary Project Management” by Stuart Woodward
  - What key metrics did OMAR and ProjectX use?
  - What does EPM stress? Why?

- “Software Management Metrics” by Herman Schultz
  - Old paper -- are the project metrics still relevant?
  - What does software volatility track?

Why Measure Projects?

- You can’t ??? what you can’t measure
- Measurement can be top-down (intrusive or excessive) management ???, or it can be a bottom-up engineer process with lots of precision but no ???...

*Measurement is about ???: Project Management measures must be balanced for best ??? of the right decisions*
Measurement-in-the-Small: How are Metrics Used?

- ??? — Which choice or improvement should be made?
  - Benchmarking for performance improvement

- ??? — What situations need to be addressed?
  - Dashboard of indicators

- ??? — How well is it (or IT) doing?
  - Scorecard on goals

Decision?

Goal

Question

Metric

Measurement Myths

- Myth 1: Measurement is ???.
- Myth 2: There is a “???” metric, and it will solve all measurement problems.
- Myth 3: The most critical success factor for a measurement system is choosing the right metric.
- Myth 4: Metrics are ???.
- Myth 5: IT departments should leave the business of ??? ?? to the business.

Source: “Dispensing the Measurement Myths”
Howard Rubin, ComputerWorld 8/15/1994
Measuring Quality

◆ ??? — the degree to which a program operates according to specification
◆ ???—the degree to which a program is amenable to change
◆ Integrity—the degree to which a program is impervious to outside attack
◆ Usability—the degree to which a program is easy to use

What is Software Complexity?

◆ Complexity - degree to which a system or component is difficult to ???, ???, or explain. Key factors include:
  • the # and intricacy of interfaces and branches
  • the degree of nesting
  • the types of data structures
◆ Why is it important?
  • Provides a means of relating the complexity of modules produced by the same process, in the same language, or within some other aspect of interest
  • It provides a systematic way of allocating sparse test or inspection ??? to the modules which show the greatest need
  • Furnishes information on ??? the complexity in the code lies: data structures, flow control, interfaces...
Software Reliability

- **Software reliability** is the probability that a software system will perform its ??? without failure for a specified ??? in a specified ???

![Diagram of software reliability concepts](image)

Source: George Stark (IBM)

---

**Performance-in-the-Large through Balanced Scorecards**

<table>
<thead>
<tr>
<th>Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>???</td>
</tr>
<tr>
<td>Cost Performance</td>
</tr>
<tr>
<td>Risk Consumption</td>
</tr>
<tr>
<td>Shareholder Value</td>
</tr>
<tr>
<td>Financial Breakeven</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>???</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfaction</td>
</tr>
<tr>
<td>Quality</td>
</tr>
<tr>
<td>Service Level</td>
</tr>
<tr>
<td>Price/Performance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>??? &amp; ???</th>
</tr>
</thead>
<tbody>
<tr>
<td>Innovation</td>
</tr>
<tr>
<td>Improvement</td>
</tr>
<tr>
<td>Morale</td>
</tr>
<tr>
<td>Rate of Change</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>???</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle Time</td>
</tr>
<tr>
<td>Quality</td>
</tr>
<tr>
<td>Productivity</td>
</tr>
<tr>
<td>Alignment</td>
</tr>
</tbody>
</table>

Source: Norton and Kaplan, HBR
Let’s Review What We’ve Covered

- **1980’s and Prior: Programming in the Small**
  - Relatively small, simple systems
  - Programmer oriented (or small teams)
  - Programs hacked or crafted
  - Unsophisticated customer

- **1990’s: Software Engineering in the Large**
  - Increasingly large, complex systems
  - Software engineering process
  - Varied hardware/software platforms
  - Teams of developers
    - Many people, roles, and risks

---

**e-Engineering in the Global?**

- **Large, complex, distributed systems**
  - CPU speed, memory, bandwidth
  - Things we’re taught to ignore
- **Heterogeneous platform architectures**
  - Mainframe, desktop, laptop, palmtop ...
- **Mobility** (Several 2000/1 IEEE Computer Issues)
  - One application runs on many devices across varied communications
  - Each device has varied resource profiles
- **Modeling, analysis, simulation...**
  - Collaborative design on partial information
- **Development teams across the globe**
  - Around the clock development
  - Decentralized ownership and control
Why Are Projects Late?

- Unrealistic deadline established by someone outside the software development group
- Changing customer requirements that are not reflected in schedule changes
- Honest underestimate of the amount of effort and/or the number of resources that will be required to do the job
- Predictable and/or unpredictable risks that were not considered when the project commenced
- Technical difficulties that could not have been foreseen in advance
- Human difficulties that could not have been foreseen in advance
- Miscommunication among project staff that results in delays
- Failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem

From Reactive to Anticipatory Management

**Common Practice**

- Lagging Performance
- Focusing on projects like winning a battle
  - Meeting “crunch mode” deliverables
- Planning/Tracking for Project Accountability
- Excusing Failure or Rationalizing Success after Delivery
- Sacrificing quality at the altar of schedule

**Emerging Practice**

- Focus on Value in Business Context
  - Economies of Scale
  - Manifold Risk of Multiple Initiatives
- Navigation through leading indicators
  - Scorecards and Dashboards
- Active Project/Program Management

**Performance Managed**
Software Projects: Part of Larger System

- Frequently cog in the wheel of a larger systems effort
  - Albeit an increasingly important and sophisticated cog!
- System engineering projects/programs entail managing multi-disciplined projects
  - Hardware, Software, Facilities, Human factors, scientists, business people, ...

Semi-Formal Definition

- Management – The activities undertaken by 1 or more persons to plan and control activities of others to achieve an objective.

- Project Management – “… a system of procedures, practices, technologies, and know-how that provide the planning, organizing, staffing, directing, and controlling necessary to successfully manage an engineering project.” [Thayer 1987]
Classic Management Activities

- Planning
- Directing
- Staffing
- Organizing
- Controlling

Software Project

Planning

- Predetermining a course of action for accomplishing organizational objectives.
  - Set goals and objectives
  - Develop strategies
  - Determine courses of action
  - Make decisions
  - Set procedures & rules
  - Develop programs
  - Forecast future situations
  - Prepare budgets
  - Document project plans
Directing

- Creating an atmosphere that will assist and motivate people to achieve desired results.
  - Provide leadership
  - Supervise personnel
  - Delegate authority
  - Motivate personnel
  - Coordinate activities
  - Facilitate communications
  - Resolve conflicts
  - Manage changes
  - Document directing decisions

Organizing

- Arranging and relating work for accomplishment of objectives and the granting of responsibility and authority to obtain those objectives.
  - Identify and group required tasks
  - Select and establish organizational structures
  - Create organizational positions
  - Define responsibilities and authority
  - Establish position qualifications
  - Document organizational structures
Controlling (Active Navigation)

- Measuring and correcting performance of activities toward objectives according to plan.
  - Develop standards of performance
  - Establish monitoring and reporting systems
  - Measure results
  - Initiate corrective actions
  - Reward and discipline
  - Document controlling methods

Staffing

- Selecting and training people for positions in the organization.
  - Fill organizational positions
  - Assimilate newly assigned personnel
  - Educate or train personnel
  - Provide for general development
  - Evaluate and appraise personnel
  - Compensate
  - Terminate assignments
  - Document staffing decisions
**Project Management Truths**

- You can get someone to commit to an unreasonable deadline, but you can’t bully him into meeting it
  - Corollary: The more outrageous the deadline, the more it costs to attempt to meet it
- Experienced PM’s Motto:
  Projects fail one day at a time
  - Corollary: It is infinitely harder to get out of trouble than it is to get into or stay out of trouble
- The Project is the only IT value delivery mechanism
  - Corollary: Value is seldom delivered before a project is complete, it takes time for that value to be realized
- Projects progress to 90% completion very rapidly...
  - Corollary: They remain there while great sums of money and time are spent to keep them there

---

**Project Management Truths**

- You can freeze specifications, but you can’t freeze expectations
  - Corollary: The conditions attached to a promise are forgotten, but the promise is remembered
- When quoting “off the shelf”, remember there is no shelf
- Nine women cannot have a baby in one month
- It is more important to know where you are going than how fast you are going
- For a project to succeed, you must have good planning and control...however, good planning and control will not ensure the success of the project
- If project content is allowed to change freely, the rate of change will soon exceed the rate of progress
What makes a good Software Manager?  
People Perspective

◆ Rob Thomasett – “most projects fail because of people and PM concern rather than technical issues.”
◆ ~80% of software project managers come from technical ranks
  • Some with natural abilities, others must learn
  • Often with predispositions about managers and how projects should be run
  • Most are surprised by their first project
    – Budgets, resources, customers, estimates, teams, meetings, decisions, and risks
◆ Beware of the little hairy boss syndrome...

Some Sage Advice…

◆ Be flexible.
  • Let your people perform according to their capabilities. Stretch goals are good but they have to have enough room to stretch.
◆ Have compassion.
  • Deal compassionately with difficult people (often they do not know they are difficult and may be difficult because they do not understand)
◆ Know when to lead and when to manage.
  • Lead people... manage process and product (by example) (systematically)
◆ Accept the role of meetings.
  • Communication, not bureaucracy
  • Prepare and manage them
Software Teams

Consider the following factors when selecting a software project team structure ...

- Difficulty of the problem to be solved
- Size of the resultant program(s) in lines of code or function points
- Time that the team will stay together
- Degree to which the problem can be modularized
- Required quality and reliability of the system to be built
- Rigidity of the delivery date
- Degree of sociability (communication) required for the project – Relationship management

Business Perspective

- Again, most software project managers did not come from the business...
- Chaos Study
- Creating software products to benefit the business and its customers
  - Who wants the software and who doesn’t?
  - Who are the users? What will the software do for them and the business?
  - When do users need (not want) the software?
  - Why does the business need the software?
  - Why is your team developing the software?
- Business investment portfolio view
Basis for IT Portfolio Investment

- **Value maintenance** — *Prevent Business Discontinuity*
  - Managing ongoing, non-discretionary investments in IT assets

- **Value enhancement** — *Invest in Existing Assets According to Business Strategy*
  - Discretionary investments in improving or growing IT asset base

- **Value exploration** — *Engage in high risk, high yield opportunities*
  - Venture into high-risk/high-payoff IT investments

Process Perspective

- **Software process has matured into a vital part of software projects**
  - Software Engineering Institute’s Capability Maturity Models
  - Best practices
  - ISO 9000, Malcolm-Baldridge, TQM...

- **At the heart of every project plan is an activity/task set derived largely from the process**
Management Secrets 😊

- Avoid having team members work in isolation
- Stay with your project team – they are the ones delivering the products of the project
- Concentrate on Tasks – not tools
- Do your homework (no this is not a subliminal message for CS6704)
  - Stay current on latest advancements in management and technical techniques for projects that you manage
- Sounds like common sense… but it is not so common! (Do your homework! - this is a subliminal message for CS6704)

The 3 P’s

- People 😊
  - Critical to all projects but the most variable in the management equation
  - Creative, complex, capable, costly
- Process
  - Most focused on in recent years
  - Manage the process through measures (next section)
  - Universe, world, and atomic views
- Product
  - Software is to be change tolerant
  - Quality measured
  - Ultimate delivery
Balancing the 3 P’s

- Difficult of the product dependent on people and process capabilities
  - Must triangulate on Pfactors...
- People Knowledge
  - Domain
  - System
  - Programming
- Process Maturity
  - Levels 1-5 on the SEI-CMM scale
- Product Maturity
  - From research prototype to packaged component

Visibility

- Software is invisible to naked eye...
  - Intangible – must be measured with a computer
- Software projects are largely invisible (until they complete) – managed as a cost
- Project managers must bring visibility to software product and project
- Two “dreaded” visibility vehicles
  - Documentation
  - Metrics
- Project team vision, commitment, and group memory
Meaningful Management Visibility

**Executive Decision View**
- ROI, ROM, EVA, ...
- Business Impact
- Price/performance
- Risk/opportunity . . . Value

**Management View**
- Costs/budget
- Schedule/effort/delay
- Utilization and loading
- Resource availability . . .

**Operational View**
- Process/activities
- Products/specs
- Policy/procedures
- Constraints/guides . . .

Key Visibility Principles for Metrics
- Clearly defined metrics, consistently applied
- Metrics are only indicators, use them accordingly
- Focus on leading indicators over lagging ones
- Recognize indicators of problems
  - Lack of change
  - Frequent change
  - Slow, steady deviation from plans

*Software metrics are navigational instruments giving position, direction, and rate of change.*
Configuration Management & Standards

No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle. *Ed Bersoff, et al, 1980*

- The wonderful thing about standards is that there are so many to choose from...
  - But when leveraged correctly, they simplify the process and management of a software project
- Standards save time and money
  - Make things manageable and reduce reinvention
- Example:
  - IEEE 1042 – SCM Standard

Iterative Models

- Prototyping
- listen to customer
- build/revise mock-up
- customer test-drives mock-up
- team #1
  - business modeling
  - data modeling
  - process modeling
  - application generation
  - testing & turnover
- team #2
  - business modeling
  - data modeling
  - process modeling
  - application generation
  - testing & turnover
- team #3
  - business modeling
  - data modeling
  - process modeling
  - application generation
  - testing & turnover
- RAD
- 60 - 90 days

*VirginiaTech*
The Incremental Model

An Evolutionary (Spiral) Model
### SEI’s Software Process Capability Maturity Model

<table>
<thead>
<tr>
<th>Level</th>
<th>Focus</th>
<th>Key Process Areas</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Optimizing</td>
<td>Continuous Process Improvement</td>
<td>Productivity &amp; Quality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defect prevention</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Technology innovation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Process change management</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Managed</td>
<td>Product and Process Quality</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Process measurement and analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quality management</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Defined</td>
<td>Engineering Process</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organization process focus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organization process definition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Peer reviews</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Training program</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intergroup coordination</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software product engineering</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Integrated software management</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Repeatable</td>
<td>Project Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software project planning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software project tracking</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software subcontract mgt.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software quality assurance</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software configuration mgt.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirements management</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Initial</td>
<td>Heroes</td>
<td></td>
</tr>
</tbody>
</table>

*Source: Software Engineering Institute*

### Process Improvement Maturity Levels

1. **Ad hoc**
   - High variability/Low predictability
   - Heroes => Success
   - High Risk

2. **Repeatable**
   - Project metrics focus
   - Low variability/Medium predictability
   - PM => Success
   - Medium-High Risk

3. **Defined**
   - Process metrics focus
   - High predictability
   - Process => Success
   - Medium Risk
More Traction at Upper levels...

First, What doesn’t work?

- Applying a waterfall process to a project where there is considerable unknowns and risk
  - Or... Applying an evolutionary (learning) process to a project that is well-understood
- Not having Measurable/Visible results
  - Or... Too much measurement overhead
- Not applying effective configuration control (losing a grip on baselines)
  - Or... Over-control despite little change
- No standards – reinvention and rework
  - Or... Standards spaghetti integration
Managing a Project Day by Day: Creating a Good Environment

- **Emotional Safety**
  - Integrity, values, communication...leaving your human condition out of others’ lives

- **Emphasize TEAM**
  - Empowerment to succeed (and to make mistakes) together

- **Lots of Personal Interaction**
  - Walk-about techniques

- **Balance Work and Rest**
  - Crunch mode mixed with stable times

- **Structure for Success (professionalism)**
  - Collegial, trust, keeping current, visibility, and mistakes

---

Visibility

- **Control = Plan + Status + Corrections**
  - Plan => Targets or Goals/Objectives
  - Status => Position + Direction + Rate of Change
  - Correction => Tactics or Adjustments

- **Getting Status Information**
  - Informal – management by walking around...
    - Encouragement
    - Localized problem resolution
  - Formal – Scheduled...Communication
    - Reporting
    - Reviews
    - Inspection
Project Management Visibility Concerns

- product quality?
- risk assessment?
- measurement?
- cost estimation?
- project scheduling?
- customer communication?
- staffing?
- other resources?
- project monitoring?

Source: Software Project Manager’s Network

Project Management Dashboard

Source: Software Project Manager’s Network
Status Reporting Vehicles

- Gantt Charts
- PERT/CPM Charts
- Software Size Status (Deviation From Plan)
- Cumulative Cost (DFP)
- Staffing (DFP)
- Earned Value
- Requirement Stability (Volatility)
- Slippage Charts

Productivity Management Theories

- Management Theories:
  - X: Fredrick Taylor (1911) professes that more productivity with efficient production
  - Y: Evans, Piazza, and Dolkas (1983) profess that stimulating more creativity and individual initiative brings better results
  - Z: Gellerman (1978) profess that corporate culture and conflict resolution bring better results
- They are all right – in their own respect
- Theory-W moves the productivity equation outside of the internal organization
Theory-W: Win-Win?

- Theory-W moves the productivity equation outside of the internal organization
- What makes a “win” for key stakeholders
- Basic Steps
  - Establish a set of win-win preconditions
    - Understand how people want to win
    - Establish reasonable expectations
    - Match peoples’ tasks to their win conditions
    - Provide supportive environment
  - Structure a win-win software process
    - Establish a realistic process plan
    - Use plan to control project
    - Identify project’s “win-lose” and “lose-lose” risks
    - Keep people involved
  - Structure a win-win software product
    - Match product to users and maintainers win conditions

Basic Software Planning Steps

- Scoping—understand the problem and the work that must be done
- Estimation
  - How much effort, cost, and time?
- Risk
  - What can go wrong? How can we avoid it?
    - What can we do about it?
- Schedule
  - How do we allocate resources along the timeline?
  - What are the milestones?
- Control strategy
  - How do we control quality?
  - How do we control change?
Elements of a Good Plan

- WRITE IT DOWN! -- Planning is the important part! Basic Elements:
  - Task List, Resources, Task Network, Estimates, Risks/Contingencies

- Do’s
  - Plan time for planning (otherwise you are behind from the beginning)
  - Manage with measures
  - Manage the value deliver through risks
  - Allow time for Preliminary Design

- Don’ts
  - Accept arbitrary plans (unless you can negotiate the product)
  - Play estimation games

Basic Software Project Plan

1.0 GOALS AND OBJECTIVES

1.1 Statement of Scope
  1.1.1 General Requirements
  1.1.2 Extended Enhancement

1.2 System Context

1.3 Major Constraints

2.0 PROJECT ESTIMATES

2.1 Estimation Techniques And Results
  2.1.1 Process-Based Estimation
  2.1.2 LOC-Based Estimation
  2.1.3 Historical Data Used for Estimates

2.4 Project Resources
  2.4.1 Staffing
  2.4.2 Minimal Hardware Requirements
  2.4.3 Minimal Software Requirements
Basic Software Project Plan (continued)

3.0 RISK MANAGEMENT
   3.1 Scope And Intent of Risk Management Activities
   3.2 Risk Management Organizational Role
   3.3 Functional Data Description
      3.3.1-n Description of Risk n
   3.4 Project Risk Table (Probability and Impact)

4.0 PROJECT SCHEDULE
   4.1 Deliverables and Milestones
   4.2 Work Breakdown Structure

5.0 PROJECT TEAM ORGANIZATION
   5.1 Team Structure
   5.2 Member Responsibilities

6.0 TRACKING AND CONTROL MECHANISMS
   6.1 Quality Assurance Mechanisms
   6.2 Change Management and Control

Evolution of Planning Fidelity

<table>
<thead>
<tr>
<th>Inception</th>
<th>Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>High</td>
</tr>
<tr>
<td>Environment</td>
<td>Mod.</td>
</tr>
<tr>
<td>Requirements</td>
<td>High</td>
</tr>
<tr>
<td>Design</td>
<td>Mod.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Low</td>
</tr>
<tr>
<td>Assessment</td>
<td>Low</td>
</tr>
<tr>
<td>Deployment</td>
<td>Low</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Elaboration</th>
<th>Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>High</td>
</tr>
<tr>
<td>Environment</td>
<td>High</td>
</tr>
<tr>
<td>Requirements</td>
<td>High</td>
</tr>
<tr>
<td>Design</td>
<td>High</td>
</tr>
<tr>
<td>Implementation</td>
<td>Mod.</td>
</tr>
<tr>
<td>Assessment</td>
<td>Mod.</td>
</tr>
<tr>
<td>Deployment</td>
<td>Low</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transition</th>
<th>Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>High</td>
</tr>
<tr>
<td>Environment</td>
<td>High</td>
</tr>
<tr>
<td>Requirements</td>
<td>Low</td>
</tr>
<tr>
<td>Design</td>
<td>Low</td>
</tr>
<tr>
<td>Implementation</td>
<td>Mod.</td>
</tr>
<tr>
<td>Assessment</td>
<td>High</td>
</tr>
<tr>
<td>Deployment</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Construction</th>
<th>Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>High</td>
</tr>
<tr>
<td>Environment</td>
<td>High</td>
</tr>
<tr>
<td>Requirements</td>
<td>Low</td>
</tr>
<tr>
<td>Design</td>
<td>Mod.</td>
</tr>
<tr>
<td>Implementation</td>
<td>High</td>
</tr>
<tr>
<td>Assessment</td>
<td>High</td>
</tr>
<tr>
<td>Deployment</td>
<td>Mod.</td>
</tr>
</tbody>
</table>
Cost Estimation

- Project scope must be explicitly defined
- Task and/or functional decomposition is necessary
- Historical measures (metrics) are very helpful
- To assure fit, use two or more estimation techniques
- Uncertainty is inherent in most estimation endeavors – plan on it!

Creating a Task Matrix

Obtained from “Process Framework”

Application Functions

Effort required to accomplish each framework activity for each application function
Parametric Estimates

▸ A parametric cost estimating relates the cost of a system to one or more parameters (variables) of the system such as physical or performance characteristics

▸ Attributes
  • Moderate to high level of detail
  • Based on actual costs of many systems
  • Uses system parameters to estimate costs

▸ Advantages
  • Sensitive to significant design changes
  • Quantifies effects of cost drivers
  • Based on “real world” experience of many systems
  • Gives quick, reproducible results

Parametric Estimates (Concluded)

▸ Disadvantages
  • Inputs are subjective
  • Results not as precise as bottom-up
  • Requires skilled analyst to develop

▸ Uses
  • Cost estimates and trade-offs for systems in early development
  • Quick reaction estimates
  • Independent check on other estimates
Analogy Estimating

- Estimating by Analogy relates the cost of a system to the cost of a known similar system through comparisons of such items as complexity, technical characteristics, and producibility

- Attributes
  - Direct relationship to costs for a similar item
  - May be adjusted by subjective factors
  - Requires engineering judgment to quantify differences between proposed system and the analogue system

- Advantages
  - Requires limited data collection
  - Estimates when little data available
  - Based on “actual” costs (depending on adjustments)

Analogy Estimating (Concluded)

- Disadvantages
  - Based on limited data
  - Subjective adjustments may be difficult to validate
  - Accuracy highly dependent on similarity between items
  - Difficult to assess impact of design changes
  - Does not identify cost drivers
  - Requires that similar items must have been procured

- Uses
  - When little data available
  - Quick, rough order of magnitude
  - Check on other techniques
**Bottom-Up Estimating**

- **Bottom-up (grass roots) estimating builds up an estimate of a system using detailed information**

- **Attributes**
  - Low level of detail
  - Based on detailed specifications

- **Advantages**
  - Sensitive to changes in design
  - Provides visibility into cost drivers
  - The most precise cost estimating methodology

---

**Bottom-Up Estimating (Concluded)**

- **Disadvantages**
  - Requires detailed design information
  - Time consuming
  - Difficult to perform cost trade-offs
  - Accuracy depends on stability of design, and skill of team

- **Uses**
  - Production estimating
  - Estimate development of firm designs
Sound Estimation Guidelines

- Estimate using at least two techniques
- Get estimates from independent sources
- Avoid over-optimism, assume difficulties
- You’ve arrived at an estimate, sleep on it
- Adjust for the people who’ll be doing the job—they have the highest impact

Continuing with Software Estimation

- Parametric Estimation is the most common among estimation techniques
- A simple formula for estimating FTE's parametrically might be:
  \[
  \text{# of FTEs} = \frac{\text{EquivalentLOC}}{\text{ProductivityRate}}
  \]
  where:
  - \( \text{EquivalentLOC} = \text{LOC} \cdot f(\text{Complexity}) \)
  - \( \text{ProductivityRate} = \frac{\text{AvgLOC} \cdot f(\text{Skill}) \cdot f(\text{Language})}{\text{StaffYear}} \)
- Since all code and staff are not created equal, we need to clarify some parameters
Complexity Parameter

- \( f(\text{Complexity}) = \left(\%\text{simple} \times \text{SimpleWeight}\right) + \left(\%\text{moderate} \times \text{ModerateWeight}\right) + \left(\%\text{complex} \times \text{ComplexWeight}\right) / 100 \)

- SimpleWeight => .5
- ModerateWeight => 1
- ComplexWeight => 1.5

Basic Parametric Approach

- Obtain Scope and Requirements
- Identify Models for Estimation
  - Parametric and others
- Establish assumptions & collect data
- Develop initial estimates with available data
  - Size
  - Effort and Cost
  - Schedule
- Assess Risks
- Review assumptions & validate data
- Refine estimate
- Refine estimate
- REFINE ESTIMATE...
Software Parametric Estimation
Process-in-the-Large (MITRE)

Calibration with Historical Measures

Key productivity factors and effect ratios:

- **Staff ability**: 20:1 - increases with increasing ability
- **Change rate**: 10:1 - decreases with increasing change
- **Program complexity**: 8:1 - decreases with increasing complexity
- **Program size**: 8:1 - decreases with increasing size
- **Team size**: 8:1 - decreases with increasing team size
- **Application type**: 8:1 - depends on type
- **Reused code**: 5:1 - increases with reuse
- **Language level**: 4:1 - increases with higher level language
- **Methods/techniques**: 4:1 - increases with effective methods
- **Project duration**: 3:1 - decreases with increasing duration
Delphi Technique

1. Facilitator presents each expert with the project’s specification and an estimation form
2. Facilitator calls a group meeting in which the experts discuss product issues related to size
3. Each expert fills out the form anonymously
4. Facilitator prepares a summary of the estimates on an Iteration Form and returns them to the experts
5. Facilitator assembles group to discuss the most widely varied estimates
6. 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

Homework Assignment for 10/15/01

◆ Prepare for Midterm Exam!
◆ Get a head start on next week's assignment:
  • Read text Chapter 7 on Risk Management
  • Read Paper entitled “Large-Scale Project Management is Risk Management” by Robert Charette
◆ Start thinking about a Risk Management Plan for the “Project Management Metrics Database”
◆ Have a great week!