Software Engineering
CS5704: Class 2 - 1/26/01

Instructor: Shawn A. Bohner
Voice: (703) 538-8374
Email: bohner@nvc.cs.vt.edu
Teaching Assistant: Sepna Georges
Voice: (703) 538-8381
Email: sgeorge@vt.edu

Agenda

▲ Review Reading Material
   ● Discussion
▲ Chapter 3 – Project Management Concepts
   ● Break
▲ Chapter 4 – Software Process & Project Metrics
▲ Homework Assignment
▲ Project Discussion
Spring Semester Timeline

15 weeks, 12 sessions to go…
So much to do and so little time…

What is Software?

▲ Software is part of a computer system that is intended to change
▲ Business changes and its computer systems must respond
▲ Software is engineered, not manufactured
▲ Software is intangible
▲ Software is complex

What makes software intangible?
Why is software complex?
Software Doesn’t Wear Out

Software doesn’t change with age or “wear out” with use! However, ...

- Software “ages” or becomes “obsolete” with a changing environment
- Software deteriorates or “degrades” with continued changes

Why does software change?
Why does “change” deteriorate software?

Software Design Degradation

The Original Software Design...

Easy to Understand
Components well isolated to facilitate change
Isolation supports change validation

...Plus a few “Changes”

Increased size and complexity ...but it works (for awhile)
Reliability of system degrades, errors creep in
At some point, it’s unmaintainable ...effort to make the next change becomes prohibitive
The Linear Model

What is an other name for the Linear Model?
When is the Linear Model most appropriate?
What are some examples of workproducts from each of the activities?
Why does this approach sometimes fail?

Iterative Models

Prototyping
When is prototyping appropriate?
Should a prototype be extended into a production system? Why (not)?
Why use RAD approach?
What are some drawbacks?
The Incremental Model

When is the Incremental Model appropriate?
What is an increment?

An Evolutionary (Spiral) Model

When is the evolutionary spiral approach appropriate?
What does WINWIN offer for this approach?
### 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></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>

Why is it bad to focus on activities 2 levels above your current maturity level?

*Source: Software Engineering Institute*
**Purpose**

▲ The purpose of this chapter is to provide an overview of software project management principles.
- Synergy of People, product, process, and project
- Management is an optimization process

▲ Objectives
- Outline key project management principles
- Understand 4 P’s in context of software
- Examine managing engineering of software in the large

---

**Project Management is about Business**

▲ Business initiatives on the rise around active markets and emerging technologies
- CRM and e-Business lead the pack

▲ Still, M&A and Y2K initiatives require oversight

▲ Software project failure rate: 1 in 3 ...
- costs Billions every year

---

**LOB Priorities for Emerging Technology**

<table>
<thead>
<tr>
<th>Category</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRM</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e-Biz B2B</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e-Biz B2C</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFA</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERP</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMO</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

Source: 2000 META Group Worldwide Benchmark

*Must manage increasing business risks due to growing volume and complexity of Business initiatives*
Software Projects start with the IT Asset Portfolio

Information Assets
- Information Continuum (data -> knowledge)
- Business Models/Rules
- Repositories
- Intelligence/Measures

Technology Assets
- Business Applications
- IT Infrastructure/OPS
- Methods and Tools
- Relevant Architectures

Human Capital Assets
- Staff/Skills Mix
- Relationships Corporate, LOB, Customers, Suppliers

Process Assets
- IT Processes
- Business Process
- Value Stream
- Innovation
- Customer

Basis for IT Portfolio Investment

▲ Value maintenance — managing ongoing, non-discretionary investments in IT assets
▲ Value enhancement — discretionary investments in improving or growing IT asset base
▲ Value exploration — venture into high-risk/high-payoff IT investments
Maintain Existing IT Asset Value

▲ IT liability avoidance and value retention
▲ Fund baseline costs for critical business operations, maintenance, and support
▲ Skeleton funding based on minimum headcount and costs to keep system running
▲ Incentives to reduce baseline costs

Enhancing Existing IT Asset Value

▲ Strategic priorities
  ● Investments criteria
  ● Investment process
▲ Phased funding on projects with interim deliverables
  ● Jump-start
  ● Initial capability
  ● Advanced function
▲ Continued allocations through project value and delivery commitments being met and communicated
Exploring Future IT Asset Value

▲ Requires dynamic mindset for quick response to value and market changes
  ● Digital Planning for agility
▲ Value Enhancement investment criteria plus:
  ● Business/Market advantage
  ● 2-3 year ROI/IRR projection
  ● Clear exit strategy
▲ Manage using a venture funding model – close and regular interactions

Investment Model IT Portfolio

<table>
<thead>
<tr>
<th>Cost to Conform</th>
<th>Followers</th>
<th>Cost to Excel</th>
<th>Leaders</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minimum Investment</strong></td>
<td>Value Maintenance</td>
<td>Excess Investment</td>
<td>Leveraged Value</td>
</tr>
<tr>
<td><strong>Value Exploration</strong></td>
<td>Lost Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Value Enhancement</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Engage in high risk, high yield opportunities
The 4 P’s

▲ People — the most important element of a successful project (in software projects, success is spelled “TEAM”)
▲ Product — the software to be built (life cycle products from requirement to code to operational application)
▲ Process — the set of framework activities and software engineering tasks to get the job done
▲ Project — all work required to make the product a reality (execution of the plan)

Software Projects

Some Factors that influence the end result...

- size
- delivery deadline
- budgets and costs
- application domain
- technology to be implemented
- system constraints
- user requirements
- available resources
Project Management Concerns

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

Why Projects Fail?

- An unrealistic deadline is established
- Changing customer requirements
- An honest underestimate of effort
- Predictable and/or unpredictable risks
- Technical difficulties
- Miscommunication among project staff
- Failure in project management
Projects are about Risk

▲ Risk . . . lack of information, time, and/or control . . . impact X exposure
▲ Value at risk — risk precedes value
▲ Must prepare for the future and simultaneously harvest returns of past investments
▲ Failure to act on opportunity can create more future risk

Banker’s Risk
Investor’s Risk
Gambler’s Risk

Must Effectively Balance Technology Risk and Opportunity

Software Teams

The following factors must be considered 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
Organizational Paradigms

▲ Closed paradigm—structures a team along a traditional hierarchy of authority
▲ Random paradigm—structures a team loosely and depends on individual initiative of the team members
▲ Open paradigm—attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
▲ Synchronous paradigm—relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

Melding Problem and Process

<table>
<thead>
<tr>
<th>COMMON PROCESS FRAMEWORK ACTIVITIES</th>
<th>customer</th>
<th>communication</th>
<th>planning</th>
<th>risk analysis</th>
<th>engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineering Tasks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text input</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Editing and formatting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic copy edit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page layout capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic indexing and TOC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document production</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To Get to the Essence of a Project

▲ Why is the system being developed?
▲ What will be done? By when?
▲ Who is responsible for a function?
▲ Where are they organizationally located?
▲ How will the job be done technically and managerially?
▲ How much of each resource (e.g., people, software, tools, database) will be needed?

Barry Boehm

Critical Practices

▲ Formal risk analysis
▲ Empirical cost and schedule estimation
▲ Metrics-based project management
▲ Earned value tracking
▲ Defect tracking against quality targets
▲ People aware project management
Moving From Projects to Programs

**Common Practice**

- 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 alter 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

From Reactive to Anticipatory Management

---

**Chapter 4**

Software Process and Project Metrics
Purpose

▲ The purpose of this chapter is to introduce metrics as a mechanism for improving the software development/maintenance process and managing software projects
  ● Create enough visibility to see where you are going!
▲ Objectives
  ● Outline software process and metrics
  ● Outline key aspects of measurement
  ● Examine how to software measures support management of software

Measurement-in-the-Small: Metric Basics

Measurement is the process of objectively assigning values to entities to characterize specific attributes
  ● An Entity is an object (product) or event (process)
  ● An Attribute is a feature or property of an entity
  ● An Objective is a well-defined rule for assigning the numbers

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attribute</th>
<th>Measurement</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>height</td>
<td>6 feet</td>
<td>bigger number is “taller height”</td>
</tr>
</tbody>
</table>
Measurement-in-the-Small: How are Metrics Used?

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

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

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

Applying the Goal-Driven Process

- The goal-driven process begins with identifying business goals and breaking them down into manageable subgoals
- It ends with deploying a plan for implementing well-defined measures and indicators that support the goals
- Along the way it maintains traceability back to the goals, so that those who collect and process measurement data do not lose sight of the objectives

Key Principles for Analyzing 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

Meaningful Measures for the Right Management Levels

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 . . .
Framework for Measures

**Project Management**
- Estimates vs. Actuals
  - Size (change and build)
  - Cost/Budget
  - Effort/Schedule
- Risks (impact & exposure)
  - Resource Availability
  - Technology
  - Delivery

**Product Management**
- Size in KLOCs or FPs
  - Maintenance and Development
- Quality/Reliability
  - Pre and Post-Delivery Defects
  - Severity
- Change actions
  - Type (fix, enhancement, update)
  - Impact of change (small -> large)
  - Priority (emergency -> routine)

**Business Management**
- Business Value of IT
- Business Risks
- Financial breakeven point
- ROI/ROA/ROE

**Process Management**
- Throughput - rate of changes delivered
- Cost of operations
- Inventory
- # of concurrent changes
- Testing Efficiency

**Core Metrics**

**Process Metrics**

- majority focus on quality achieved as a consequence of a repeatable or managed process
- statistical SQA data
  - error categorization & analysis
- defect removal efficiency
  - propagation from phase to phase
- reuse data
Project Metrics

▲ Effort/time per SE task
▲ Errors uncovered per review hour
▲ Scheduled vs. actual milestone dates
▲ Changes (number) and their characteristics
▲ Distribution of effort on SE tasks

Product Metrics

▲ focus on the quality of deliverables
▲ measures of analysis model
▲ complexity of the design
  ● internal algorithmic complexity
  ● architectural complexity
  ● data flow complexity
▲ code measures (e.g., Halstead)
▲ measures of process effectiveness
  ● e.g., defect removal efficiency
Typical Size-Oriented Metrics

- Errors per KLOC (thousand lines of code)
- Defects per KLOC
- $ per LOC
- Page of documentation per KLOC
- Errors / person-month
- LOC per person-month
- $ / page of documentation

Typical Function-Oriented Metrics

- Errors per FP (thousand lines of code)
- Defects per FP
- $ per FP
- Pages of documentation per FP
- FP per person-month
Why Opt for FP Measures?

- independent of programming language
- uses readily countable characteristics of the "information domain" of the problem
- does not "penalize" inventive implementations that require fewer LOC than others
- makes it easier to accommodate reuse and the trend toward object-oriented approaches

Computing Function Points

1. Analyze information domain of the application and develop counts
2. Weight each count by assessing complexity
3. Assess influence of global factors that affect the application
4. Establish count for input domain and system interfaces
5. Assign level of complexity or weight to each count
6. Grade significance of external factors, $F_i$ such as reuse, concurrency, OS, ...

$$\text{function points} = \sum (\text{count} \times \text{weight}) \times C$$

where:
- complexity multiplier: $C = (0.65 + 0.01 \times N)$
- degree of influence: $N = \sum F_i$
Analyzing the Information Domain

<table>
<thead>
<tr>
<th>measurement parameter</th>
<th>weighting factor</th>
<th>count</th>
<th>simple</th>
<th>avg.</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of user inputs</td>
<td></td>
<td>X 3</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>number of user outputs</td>
<td></td>
<td>X 4</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>number of user inquiries</td>
<td></td>
<td>X 3</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>number of files</td>
<td></td>
<td>X 7</td>
<td>10</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>number of ext.interfaces</td>
<td></td>
<td>X 5</td>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>count-total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>complexity multiplier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Taking Complexity into Account

Factors are rated on a scale of 0 (not important) to 5 (very important):

- data communications
- distributed functions
- heavily used configuration
- transaction rate
- on-line data entry
- end user efficiency
- on-line update
- complex processing
- installation ease
- operational ease
- multiple sites
- facilitate change
Measuring Quality

▲ Correctness — the degree to which a program operates according to specification
▲ Maintainability—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

Defect Removal Efficiency

DRE = (errors) / (errors + defects)

where
errors = problems found before release
defects = problems found after release
Sample: Zoning Project Performance

Effectiveness: Project Delivery Commitment

<table>
<thead>
<tr>
<th>Unit 1</th>
<th>Actual = Est</th>
<th>Unit 2</th>
<th>Unit 3</th>
</tr>
</thead>
</table>

Prior Months – Actual Months/Estimated Months

Project Management Dashboard

Source: Software Project Manager’s Network
**Example Management Dashboards**

Source: Motorola Software Solution

**Chart 1: Software Development Process and Product Quality**

**Chart 2: Customer Satisfaction**

**Chart 3: Cycle Time and Productivity**

**Chart 4: Software Engineering Technology**

**Chart 5: SEI Key Process Area Profile**

---

**Prog. Mngt. Capability Maturity Model**

<table>
<thead>
<tr>
<th>Maturity Level</th>
<th>Key Process Area Concentrations</th>
<th>Strategic Inflection Point</th>
<th>Effective Span</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Incorporated</td>
<td>Value Management, Business Continuity Planning, Procurement Management, Outsourcing and Contract Management, PM Center of Excellence</td>
<td>Integration with Business Industry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Managed</td>
<td>Program Process Management, Project Integration Management, Project Performance Management, Vendor Management, PM Career Path, Staff Performance Management, Customer Relationship Management, Contingency Management, Communications Management</td>
<td>Dynamic Micro-Level Change</td>
<td>Multiple Business Units</td>
<td></td>
</tr>
<tr>
<td>3 Defined</td>
<td>PM Methodology, Skill Management, PM Training, Risk Management, Change Management, Staff Resource Management, Environment Resource Management, Conflict/Issue Management</td>
<td>Static Macro-Level Change</td>
<td>Multiple Project</td>
<td></td>
</tr>
<tr>
<td>1 - Initial</td>
<td>Acquiring New PMs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Maturing PM Dashboard Metrics

<table>
<thead>
<tr>
<th>Maturity Level</th>
<th>Dashboard Metrics Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Incorporated</td>
<td>Value Leveraged/Value at Risk, Contract Cycle Time</td>
</tr>
<tr>
<td></td>
<td>Return on IT, Portfolio Capitalization Level</td>
</tr>
<tr>
<td></td>
<td>Return on Management, PM Intellectual Capital</td>
</tr>
<tr>
<td></td>
<td>Business Risk Reserve, Return on Training</td>
</tr>
<tr>
<td></td>
<td>Contract Cost Savings</td>
</tr>
<tr>
<td>4 Managed</td>
<td>Delivery Throughput, Project Approval Delays</td>
</tr>
<tr>
<td></td>
<td>Process Improvement Rate, Integration Testing Cycle Time</td>
</tr>
<tr>
<td></td>
<td>Applications Life Cycle Cost, PM Advancement Rate</td>
</tr>
<tr>
<td></td>
<td>Cost of Applications Management, Cost of Integration</td>
</tr>
<tr>
<td></td>
<td>Total Program Performance, Project Success/Failure Rate</td>
</tr>
<tr>
<td></td>
<td>Volume Purchase Savings, Customer Satisfaction</td>
</tr>
<tr>
<td></td>
<td>Maintenance/Development, Workload Ratio</td>
</tr>
<tr>
<td>3 Defined</td>
<td>Schedule Productivity, Voluntary Turnover</td>
</tr>
<tr>
<td></td>
<td>Cost Productivity, Remaining Risk Reserve</td>
</tr>
<tr>
<td></td>
<td>Resource Utilization, Configuration Churn</td>
</tr>
<tr>
<td></td>
<td>Experience and Skills Growth, Issue Resolution Rate</td>
</tr>
<tr>
<td></td>
<td>Average Team Size</td>
</tr>
<tr>
<td>2 Stable</td>
<td>Planned versus Actual Effort, Cost, Scope Creep</td>
</tr>
<tr>
<td></td>
<td>and Schedule, Number of Risk Exposures (i.e., red, amber, green)</td>
</tr>
<tr>
<td></td>
<td>Slip Rates, Requirements Change Rate</td>
</tr>
<tr>
<td></td>
<td>Remaining Risk Reserve, Project Approval Delays</td>
</tr>
<tr>
<td></td>
<td>Integration Testing Cycle Time, Project Success/Failure Rate</td>
</tr>
<tr>
<td></td>
<td>Volume Purchase Savings, Customer Satisfaction</td>
</tr>
<tr>
<td></td>
<td>Maintenance/Development, Workload Ratio</td>
</tr>
</tbody>
</table>

### Homework Assignment for 2/2/01

▲ **Read Pressman Chapters**

- 5.1-5.10
  - Do problems 5.1-5.4, 5.6
  [http://sunset.usc.edu/COCOMOII/cocomo.html](http://sunset.usc.edu/COCOMOII/cocomo.html)
- 6.1-6.9
  - Do problems 6.1-6.5, 6.12

▲ Have a great week!