Reading

- Finish chapter 3
- Start chapter 5

Effort estimation

- Expert judgment
  - analogy
  - proportion
  - Delphi technique
  - Wolverton model
- Algorithmic methods:  \( E = (a + bS^c) \cdot m(X) \)
  - \( S \) size, \( X \) cost vector, \( m \) multiplier \( a, b, c \) constants
  - Walston and Felix model:  \( E = 5.25S^{0.91} \)
  - Bailey and Basili model:  \( E = 5.5 + 0.73S^{1.15} \)
Table 3.6: Wolverton model cost matrix.

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Control</th>
<th>Input/output</th>
<th>Pre/post processor</th>
<th>Algorithm</th>
<th>Data management</th>
<th>Time-critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>OE</td>
<td>21</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>24</td>
<td>19</td>
</tr>
<tr>
<td>OM</td>
<td>27</td>
<td>24</td>
<td>23</td>
<td>20</td>
<td>31</td>
<td>25</td>
</tr>
<tr>
<td>OH</td>
<td>30</td>
<td>27</td>
<td>26</td>
<td>22</td>
<td>35</td>
<td>33</td>
</tr>
<tr>
<td>NE</td>
<td>33</td>
<td>28</td>
<td>28</td>
<td>25</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>NM</td>
<td>40</td>
<td>35</td>
<td>34</td>
<td>30</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>NH</td>
<td>49</td>
<td>43</td>
<td>42</td>
<td>35</td>
<td>57</td>
<td>57</td>
</tr>
</tbody>
</table>

Table 3.7: Walston and Felix model productivity factors.

1. Customer interface complexity
2. User participation in requirements definition
3. Customer-originated program design changes
4. Customer experience with the application area
5. Overall personnel experience
6. Percentage of development programmers who participated in the design of functional specifications
7. Previous experience with the operational computer
8. Previous experience with the programming language
9. Previous experience with applications of similar size and complexity
10. Ratio of average staff size to project duration (people per month)
11. Hardware under concurrent development
12. Access to development computer open under special request
13. Access to development computer closed
14. Classified security environment for computer and at least 25% of programs and data
15. Use of structured programming
16. Minimize standard error estimate to produce an equation such as:
   \[ E = 5.5 + 0.73S^{0.16} \]
   - Adjust initial estimate based on the ratio of errors.
   - If \( R \) is the ratio between the actual effort, \( E \), and the predicted effort, \( E' \), then the effort adjustment is defined as:
     \[ ER_{adj} = R - 1 \quad \text{if} \quad R \geq 1 \]
     \[ ER_{adj} = 1 - 1/R \quad \text{if} \quad R < 1 \]
   - Then adjust the initial effort estimate:
     \[ E_{adj} = (1 + ER_{adj})E \quad \text{if} \quad R \geq 1 \]
     \[ E_{adj} = E/(1 + ER_{adj}) \quad \text{if} \quad R < 1 \]

Bailey-Basili technique

- Minimize standard error estimate to produce an equation such as:
  \[ E = 5.5 + 0.73S^{0.16} \]
- Adjust initial estimate based on the ratio of errors.
- If \( R \) is the ratio between the actual effort, \( E \), and the predicted effort, \( E' \), then the effort adjustment is defined as:
  \[ ER_{adj} = R - 1 \quad \text{if} \quad R \geq 1 \]
  \[ ER_{adj} = 1 - 1/R \quad \text{if} \quad R < 1 \]
- Then adjust the initial effort estimate:
  \[ E_{adj} = (1 + ER_{adj})E \quad \text{if} \quad R \geq 1 \]
  \[ E_{adj} = E/(1 + ER_{adj}) \quad \text{if} \quad R < 1 \]
Table 3.8. Bailey-Basili effort modifiers.

<table>
<thead>
<tr>
<th>Total methodology (METH)</th>
<th>Cumulative complexity (CPLX)</th>
<th>Cumulative experience (EXP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree charts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer qualifications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top-down design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer machine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size estimation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal documentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program flow complexity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer language</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design formulations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consent-oriented</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program design changes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COCOMO model: stages of development

- Application composition:
  - Prototyping to resolve high-risk user interface issues
  - Size estimates in object points
- Early design:
  - To explore alternative architectures and concepts
  - Size estimates in function points
- Postarchitecture:
  - Development has begun
  - Size estimates in lines of code

Table 3.9. Three stages of COCOMO II

<table>
<thead>
<tr>
<th>Stage 1: Application Composition</th>
<th>Stage 2: Early Design</th>
<th>Stage 3: Postarchitecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Aspect</td>
<td>Model Aspect</td>
<td>Model Aspect</td>
</tr>
<tr>
<td>Composition</td>
<td>Design</td>
<td>Post-architecture</td>
</tr>
<tr>
<td>Size</td>
<td>Application</td>
<td>Function points</td>
</tr>
<tr>
<td>Function points</td>
<td>FP</td>
<td>FP and language</td>
</tr>
<tr>
<td>Equivalent SLOC</td>
<td>Equivalent SLOC</td>
<td>Function of other variables</td>
</tr>
<tr>
<td>Implicit in model</td>
<td>% change expressed as a cost factor</td>
<td></td>
</tr>
<tr>
<td>Implicit in requirements</td>
<td>% change expressed as a cost factor</td>
<td></td>
</tr>
<tr>
<td>Implicit in maintenance</td>
<td>Annual unfamiliarity</td>
<td></td>
</tr>
<tr>
<td>Traffic Scale</td>
<td>1.0 to 0.91, depending on precedentedness, software size, and service conformance, early architecture, software architecture, risk resolution, team cohesion, and SEI process maturity</td>
<td></td>
</tr>
<tr>
<td>Product cost drivers</td>
<td>Documentation needs, required reuse, and product complexity</td>
<td></td>
</tr>
<tr>
<td>Platform cost</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Personnel cost</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Project cost</td>
<td>Use of software tools, schedule, and environment multisite development</td>
<td></td>
</tr>
</tbody>
</table>

3
Table 3.10. Application point complexity levels.

<table>
<thead>
<tr>
<th>For Screens</th>
<th>For Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of data tables contained</td>
<td>Number of data tables contained</td>
</tr>
<tr>
<td>Total ≤ 4 server, ≤ 3 client</td>
<td>Total ≤ 4 server, ≤ 3 client</td>
</tr>
<tr>
<td>Number of views contained</td>
<td>Number of views contained</td>
</tr>
<tr>
<td>Total ≤ 4 server, ≤ 3 client</td>
<td>Total ≤ 4 server, ≤ 3 client</td>
</tr>
<tr>
<td>Number of sections contained</td>
<td>Number of sections contained</td>
</tr>
<tr>
<td>Total ≤ 4 server, ≤ 3 client</td>
<td>Total ≤ 4 server, ≤ 3 client</td>
</tr>
</tbody>
</table>

- Simple: simple
- Medium: medium
- Difficult: difficult

Table 3.11. Complexity weights for application points.

<table>
<thead>
<tr>
<th>Object type</th>
<th>Simple</th>
<th>Medium</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Report</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>3GL component</td>
<td>-</td>
<td>-</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 3.12. Productivity estimate calculation.

<table>
<thead>
<tr>
<th>Developers' experience and capability</th>
<th>Very low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very high</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE maturity and capability</td>
<td>Very low</td>
<td>Low</td>
<td>Nominal</td>
<td>High</td>
<td>Very high</td>
</tr>
<tr>
<td>Productivity factor</td>
<td>4</td>
<td>3</td>
<td>2.5</td>
<td>2</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Table 3.13. Tool use categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very low</td>
<td>Simple, little integration</td>
</tr>
<tr>
<td>Low</td>
<td>Simple front-end, back-end CASE, little integration</td>
</tr>
<tr>
<td>High</td>
<td>Strong, mature life-cycle tools, well-integrated with processes, methods, case</td>
</tr>
<tr>
<td>Very high</td>
<td>Strong, mature life-cycle tools, large integration, adaptable</td>
</tr>
</tbody>
</table>

Machine learning techniques

- Example: case-based reasoning
  - user identifies new problem as a case
  - system retrieves similar cases from repository
  - system reuses knowledge from previous cases
  - system suggests solution for new case
- Example: neural network
  - cause-effect network "trained" with data from past history
Evaluating models

- Mean magnitude of relative error (MMRE)
  - absolute value of mean of \( \frac{\text{actual} - \text{estimate}}{\text{actual}} \)
  - goal: should be .25 or less
- Pred(x/100): percentage of projects for which estimate is within x% of the actual
  - goal: should be .75 or greater for x = .25

<table>
<thead>
<tr>
<th>Model</th>
<th>PRED(0.25)</th>
<th>MMRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walston-Felix</td>
<td>0.30</td>
<td>0.48</td>
</tr>
<tr>
<td>Basic COCOMO</td>
<td>0.27</td>
<td>0.60</td>
</tr>
<tr>
<td>Intermediate COCOMO</td>
<td>0.65</td>
<td>0.22</td>
</tr>
<tr>
<td>Intermediate COCOMO (estimation)</td>
<td>0.76</td>
<td>0.19</td>
</tr>
<tr>
<td>Bailey-Basili</td>
<td>0.78</td>
<td>0.18</td>
</tr>
<tr>
<td>Pfleeger</td>
<td>0.78</td>
<td>0.18</td>
</tr>
<tr>
<td>SLIM</td>
<td>0.06-0.24</td>
<td>0.70-1.04</td>
</tr>
<tr>
<td>Jensen</td>
<td>0.06-0.33</td>
<td>0.70-1.01</td>
</tr>
<tr>
<td>COPMO</td>
<td>0.38-0.63</td>
<td>0.23-5.7</td>
</tr>
<tr>
<td>General COPMO</td>
<td>0.78</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Risk management requirements

- Risk impact: the loss associated with the event
- Risk probability: the likelihood that the event will occur
- Risk control: the degree to which we can change the outcome

Risk exposure = (risk probability) x (risk impact)
Three strategies for risk reduction

- avoiding the risk: change requirements for performance or functionality
- transferring the risk: transfer to other system, or buy insurance
- assuming the risk: accept and control it

Risk leverage = difference in risk exposure divided by cost of reducing the risk

Boehm’s top ten risk items

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions
- Developing the wrong user interfaces
- Gold-plating
- Continuing stream of requirements changes
- Shortfalls in externally-performed tasks
- Shortfalls in externally-furnished components
- Real-time performance shortfalls
- Straining computer science capabilities

Project plan contents

- project scope
- project schedule
- project team organization
- technical description of system
- project standards and procedures
- quality assurance plan
- configuration management plan
- documentation plan
- data management plan
- resource management plan
- test plan
- training plan
- security plan
- risk management plan
- maintenance plan
**Digital Alpha AXP: Enrollment management model**

- Establish an appropriately large shared vision
- Delegate completely and elicit specific commitments from participants
- Inspect vigorously and provide supportive feedback
- Acknowledge every advance and learn as the program progresses

**Lockheed Martin: Accountability modeling**

- Matrix organization
  - Each engineer belongs to a functional unit based on type of skill
- Integrated product development team
  - Combines people from different functional units into interdisciplinary work unit
- Each activity tracked using cost estimation, critical path analysis, schedule tracking
  - Earned value a common measure for progress

**Anchoring milestones**

- Objectives: Why is the system being developed?
- Milestones and schedules: What will be done by when?
- Responsibilities: Who is responsible for a function?
- Approach: How will the job be done, technically and managerially?
- Resources: How much of each resource is needed?
- Feasibility: Can this be done, and is there a good business reason for doing it?
Designing the System

Conceptual design
- Tells the customer what the system will do
- Answers:
  - Where will the data come from?
  - What will happen to the data in the system?
  - What will the system look like to users?
  - What choices will be offered to users?
  - What is the timing of events?
  - What will the reports and screens look like?
- Characteristics of good conceptual design
  - in customer language with no technical jargon
  - describes system functions
  - independent of implementation
  - linked to requirements

Technical design
- Tells the programmers what the system will do
- Includes:
  - major hardware components and their function
  - hierarchy and function of software components
  - data structures
  - data flow
Five ways to create designs
- Modular decomposition
- Data-oriented decomposition
- Event-oriented decomposition
- Outside-in design
- Object-oriented design

Three design levels
- Architecture: associates system components with capabilities
- Code design: specifies algorithms and data structures for each component
- Executable design: lowest level of design, including memory allocation, data formats, bit patterns

Design styles
- Pipes and filters
- Object-oriented design
- Implicit invocation
- Layering
- Repositories
- Interpreters
- Process control
- Client-server
Example of implicit invocation

```plaintext
DEBUG VALUE <system><file><line><var><value>
DEBUG ENTER <system><file><func><line><value>
DEBUG EXIT <system><file><func><line><value>
EVENT ADD <system><event_type><file><line><value>
EVENT REMOVE <system><event_type><file><line><text>
STOP-ERROR <signal><file><line>
DEBUG AT <system><file><func><line>
DEBUG FOCUS <system><file><func><line>
DEBUG CLEAR <system>
DEBUG RESET <system>
WHERE <system><level><function><line><addr><args>
WHERE_DUMP <system><level><name><value>
WHERE_BEGIN <system>
WHERE_END <system><level>
DEBUG SYSTEM <system>
DEBUG NOSYSTEM <system>
UPDATE <system><file><line>
```

Example of abstraction

Rearrange L in non-decreasing order

```plaintext
DO WHILE I is between 1 and (length of L)-1:
  Set LOW to index of smallest value in L(I), ..., L(length of L)
  Interchange L(I) and L(LOW)
END DO

DO WHILE J is between I+1 and (length of L)-1:
  IF L(LOW) is greater than L(J)
    THEN set LOW to current value of J
  ENDIF
ENDDO
Set TEMP to L(LOW)
Set L(LOW) to L(I)
Set L(I) to TEMP
```

Important design issues

- Modularity and levels of abstraction
- Collaborative design
- Designing the user interface
  - metaphors, mental model, navigation rules, look and feel
  - cultural issues
  - user preferences
- Concurrency
- Design patterns and reuse
Table 5.1. Issues to consider in trade-off analysis. (Lane, in Shaw and Garlan 1996)

<table>
<thead>
<tr>
<th>External event handling</th>
<th>Application interface abstraction level</th>
<th>User customizability</th>
<th>Abstract device variability</th>
<th>User interface adaptability across devices</th>
<th>Notation for user interface definition</th>
<th>Computer system organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>• No external events</td>
<td>• Monolithic program</td>
<td>• High</td>
<td>• Ideal device</td>
<td>• None</td>
<td>• Implicit in shared user interface code</td>
<td>• Uniprocessing</td>
</tr>
<tr>
<td>• Process events while waiting for input</td>
<td>• Abstract device</td>
<td>• Medium</td>
<td>• Parameterized device</td>
<td>• Local behavior changes</td>
<td>• Implicit in application code</td>
<td>• Multiprocessing</td>
</tr>
<tr>
<td>• External events preempt user commands</td>
<td>• Toolkit</td>
<td>• Low</td>
<td>• Device with variable operations</td>
<td>• Global behavior change</td>
<td>• External declarative notation</td>
<td>• Distributed processing</td>
</tr>
<tr>
<td></td>
<td>• Interaction manager with fixed data types</td>
<td></td>
<td></td>
<td></td>
<td>• External procedural notation</td>
<td>• Basis of communication</td>
</tr>
<tr>
<td></td>
<td>• Interaction manager with extensible data types</td>
<td></td>
<td></td>
<td></td>
<td>• Internal declarative notation</td>
<td>• System calls</td>
</tr>
<tr>
<td></td>
<td>• Extensible interaction manager</td>
<td></td>
<td></td>
<td></td>
<td>• Internal procedural notation</td>
<td>• External services</td>
</tr>
</tbody>
</table>

Characteristics of good design

- Component independence
  - coupling
  - cohesion
- Exception identification and handling
- Fault prevention and tolerance
  - active
  - passive

Techniques for improving design

- Reducing complexity
- Design by contract
- Prototyping design
- Fault-tree analysis
Design evaluation and validation

- Mathematical validation
- Measuring design quality
- Comparing designs
  - one specification, many designs
  - comparison table
- Design reviews

Table 5.5. Weighted comparison of Shaw and Garlan designs.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Priority</th>
<th>Shared data</th>
<th>Shared data type</th>
<th>Deploys inheritance</th>
<th>Pipe and filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to change algorithm</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Easy to change data</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Easy to change data</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Easy to change function</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Good performance</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Design reviews

- Preliminary design review
  - examines conceptual design with customer and users
- Critical design review
  - presents technical design to developers
- Program design review
  - programmers get feedback on their designs before implementation
Questions for any design review

Is it a solution to the problem?
Is it modular, well-structured, and easy to understand?
Can we improve the structure and understandability?
Is it portable to other platforms?
Is it reusable?
Is it easy to modify or expand?
Does it support ease of testing?
Does it maximize performance, where appropriate?
Does it reuse components from other projects, where appropriate?
Are the algorithms appropriate, or can they be improved?
If this system is to have a phased development, are the phases interfaced sufficiently so that there is an easy transition from one phase to the next?
Is it well-documented, including design choices and rationale?
Does it use appropriate techniques for handling faults and preventing failures?

Documenting the design

• Design rationale
  • menus and other display-screen formats
  • human interfaces: function keys, touch screen descriptions, keyboard layouts, use of a mouse or joystick
  • report formats
  • input: where data come from, how they are formatted, on what media they are stored
  • output: where data are sent, how they are formatted, on what media they are stored
  • general functional characteristics
  • performance constraints
  • archival procedures
  • fault-handling approach