Resource Models and Metrics
Software Resource Models and Measures

We can define a project as a set of tasks that consume resources and produce a product.

Thus resources are consumed during a project.

What types of resources exist?

- Hardware
- Software
- Human
- Support (supplies, materials, communications, facility costs, etc.)

Are the resources estimated or actual?

Are the resources desirable (resources of value), accessible (able to be used) utilized (actually used)?
A Model of a Software Project

- project consists of task
- task consumes resource, produces product
A Model of a Software Project

- project
  - consists of
    - task
      - consumes
        - resource
          - resource type
            - types of resource
              - resource use
                - use descriptors
                  - type
                  - use descriptors
                    - estimated
                      - desirable
                        - accessible
                          - utilized
                          - actual
                            - incurrence
                              - availability

(Description, milestones, target hardware, development hardware, deliverables)

(Hardware, software, human, support plus attributes of the resource)

(Work nature, Calendar time, Measure of work)
Software Resource Models and Measures

**Resource Data**

**Human Effort** data may be measured in staff-hours, weeks, months, years . . .

**Calendar time** data may be measured in calendar hours, days, weeks, months, date to date

**Computer Time** data may be measured as calendar time, execution time

They may be associated with various

Processes:

- **phases**: requirements, design, implementation, test,...
- **activities**: reading, design inspections, making changes, meetings,...

Products:

- **documents**: requirements, design, test plan, user’s manual,...
- **program parts**: system, module, design document, requirements section,...

Other project characteristics:

- **calendar time**: from date to date
**Software Resource Models and Measures**

**Sample Resource Metrics**

The data can be aggregated to define various metrics, e.g.,

- Total Effort for the project
- Design Effort, Design Effort as % of Total
- Design Calendar Time from Requirements review to Design review
- Staff time to:
  - make a test
  - run a test and check the result
  - isolate the fault?
  - design and implement a fix
  - retest
- Machine time used to run a test suite

This can be based upon actual data or estimated data.
## Software Resource Models and Measures

### Effort by Phase

<table>
<thead>
<tr>
<th></th>
<th>Analysis and Design</th>
<th>Coding and Auditing</th>
<th>Checkout and Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sage</td>
<td>39%</td>
<td>14%</td>
<td>47%</td>
</tr>
<tr>
<td>NTDS</td>
<td>30</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>Gemini</td>
<td>36</td>
<td>17</td>
<td>47</td>
</tr>
<tr>
<td>Saturn V</td>
<td>32</td>
<td>24</td>
<td>44</td>
</tr>
<tr>
<td>OS/360</td>
<td>33</td>
<td>17</td>
<td>50</td>
</tr>
<tr>
<td>TRW Survey</td>
<td>46</td>
<td>20</td>
<td>34</td>
</tr>
</tbody>
</table>
# Software Resource Models and Measures

<table>
<thead>
<tr>
<th></th>
<th>Phase</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>40%</td>
<td>35%</td>
</tr>
<tr>
<td>Code</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Checkout/Test</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td>Other</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

![Graph showing effort over time with markers for DR and CR]
Software Resource Models and Measures

Why model/measure resources?

Initial Prediction
Given what we know (or can guess) about a project, what can we predict about effort (cost), staffing, computer use, . . .?

Description of the Development Pattern
Provides insights into what is going on
How do different parameters change the pattern?
Provides an evaluation of techniques, methodology and engineering
What can we learn about future developments?

Prediction of the Next Phase from the Current Phase
What should happen next?
If it doesn’t, why not; is it a sign of trouble, etc.?

Model Validation
Does the model explain our behavior and environment?
Do the factors (model parameters) agree with our environmental factors?
Are they calibrated correctly?
Software Resource Models and Measures

Characteristics of a Good Model

- It explains our behavior and the development environment
- Parameters are calculable from known data (or easy to guess data) e.g.,
  - Maximum staffing
  - Time to delivery
  - Complexity of software
  - Lines of code, number of modules, number of I/O formats
  - Type of software
  - Amount of old/new software (design, code, specification)
- Parameters describe and can be calibrated for our environment
- It includes redundancy checks and risk analysis factors
- When the model doesn’t work
  - One can gain insight into why and what is different in the environment
Software Resource Models and Measures

What kinds of models can we build?

What is estimated: effort, staffing, cost, computer use, time

Type of Analysis:
- Least square and regression analysis
- Neural networks
- Machine learning approaches, e.g., decision trees
- Multiple Regression Models
  - Single variable vs. multi-variable
  - Adjusted baseline
  - Adjusted table driven
  - Multi-parameter

Static staffing vs. dynamic staffing
Empirical vs. theoretical
Macro vs. micro
Software Resource Models and Measures

Multivariate Modeling Solutions

Least square and logistic regression analysis
  sensitive to outliers
  requires distributional and functional assumptions
  difficult to deal with interactions
  difficult to deal with symbolic data
  models are unstable and difficult to interpret
but these models are based on a solid, well formalized theory

Neural networks
  models are very difficult to interpret
  optimal modeling strategies are still unclear in this area
but do not require explicit functional assumptions

Machine learning approaches, e.g., decision trees
  sometimes lack of solid statistical theory
but the models are easy to interpret for application domain experts
Software Resource Models and Measures

Single Variable Regression Models

Effort equation is based on a single variable, usually a measure of size.

There are several possible variations:

\[ \text{Effort} = A \times \text{size} + C \]
\[ \text{Effort} = A \times \text{size}^B \]
\[ \text{Effort} = A \times \text{size}^B + C \]

where A, B and C are constants determined by regression analysis on historical data.

Effort may be measured in

Staff: hours, weeks, months, years . . .

Size may be measured in

Lines of code, modules, I/O formats . . .
Software Resource Models and Measures

Sizing Methods Approaches

Top-down estimating

Similarities and differences estimating

Ratio estimating

Standards estimating

Bottom-up estimating

Combination of two or more basic methods

Wolverton/TRW
Goal: Measure rate of production of lines of code by projects, influenced by a number of product conditions and requirements

Data Base: 60 projects
4K to 467K source lines
12 to 11,758 staff months
Variety of task types, languages, ...

Basic Effort Equation Form: \( E = A \times \text{SIZE}^B \)

Effort Estimation Equation: \( E = 5.2L^{.91} \)

where \( E \) = effort in staff months
\( L \) = lines of code in thousands

Walston & Felix: IBM Federal Systems Division
It is possible to identify the relationship among any pair of variables by plotting the data and calculating a best fit equation.

**Other Variable Relationship Equations:**

\[
E = 5.2L^{.91}
\]

\[
DOC = 49L^{1.01}
\]

\[
D = 4.1L^{.36}
\]

\[
D = 2.47E^{.35}
\]

\[
S = .54E^{.6}
\]

where

- \(E\) = effort in staff months
- \(L\) = lines of code in thousands
- \(DOC\) = documentation in pages
- \(D\) = project duration in calendar months
- \(S\) = average staff size = \(E/D\)
After calculating the basic relationship between size and effort, how does one identify the effect of other variables?

What are the other variables?

Walston and Felix identified 69 potential influencing variables ranging from context to experience and the nature of the problem to methods used to solve the problem.

They correlated all 68 variables with productivity and selected 29 that showed a significantly high correlation with productivity.

The goal was then to find a way to measure the influence of these variables on the basic estimated effort value.
Software Resource Models and Measures

Productivity Index

The influencing variables were measured on a three point ordinal scale for

For those projects with data on those variables, they were divided into three groups of relatively equal size yielding a group of low, medium and high rating with regard to the variable

Average productivity was calculated for each for the three groups and the difference in productivity between the high and low groups was used as a base for the weighting

They then calculated an effort multiplier, called the productivity index, whose goal was to weight the effort estimate based upon the historical “influence” of the variables
The weights were calculated based upon historical data

\[
I = \sum_{i=1}^{29} W_i X_i
\]

where

- \( I \) = Productivity index
- \( W_i \) = question weight \((1/2 \log_{10} \text{(ratio of productivity change for question } i))\)
- \( X_i \) = question response \((+1, 0, \text{ or } -1)\), depending on whether the responses indicate increased, nominal or decreased productivity

The productivity index is used to adjust the initial estimator from the baseline equation by explaining deviations from the norm.
## Software Resource Models and Measures

### Productivity Index

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer interface complexity</td>
<td>&lt;normal 500</td>
<td>normal 295</td>
</tr>
<tr>
<td>User participation in the definition of requirements</td>
<td>none 491</td>
<td>some 267</td>
</tr>
<tr>
<td>Customer originated program design changes</td>
<td>few 297</td>
<td>many 196</td>
</tr>
<tr>
<td>Customer experience with the application area of the project</td>
<td>none 313</td>
<td>some 340</td>
</tr>
<tr>
<td>Overall personnel experience and qualifications</td>
<td>low 132</td>
<td>average 257</td>
</tr>
<tr>
<td>Question or Variable</td>
<td>Mean Productivity (DSL/MM)</td>
<td>Change (DSL/MM)</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Percentage of programmers doing development who participated in design of functional specifications</td>
<td>&lt; 25% 153</td>
<td>25-50% 242</td>
</tr>
<tr>
<td>Previous experience with operational computer</td>
<td>minimal 146</td>
<td>average 270</td>
</tr>
<tr>
<td>Previous experience with programming languages</td>
<td>minimal 122</td>
<td>average 225</td>
</tr>
<tr>
<td>Previous experience with application of similar or greater size and complexity</td>
<td>minimal 146</td>
<td>average 221</td>
</tr>
</tbody>
</table>
## Modeling and Measuring Resources
### Productivity Index

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ratio of average staff size to duration (people/month)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;0.5</td>
<td>305</td>
<td>132</td>
</tr>
<tr>
<td>0.5-0.9</td>
<td>310</td>
<td></td>
</tr>
<tr>
<td>&gt;0.9</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td><strong>Hardware under concurrent development</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>297</td>
<td>177</td>
</tr>
<tr>
<td>yes</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td><strong>Development computer access, open under special request</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>226</td>
<td>357</td>
</tr>
<tr>
<td>1-25%</td>
<td>274</td>
<td>131</td>
</tr>
<tr>
<td>&gt;25%</td>
<td>131</td>
<td></td>
</tr>
<tr>
<td><strong>Development computer access, closed</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0-10%</td>
<td>303</td>
<td>170</td>
</tr>
<tr>
<td>11-85%</td>
<td>251</td>
<td>133</td>
</tr>
<tr>
<td>&gt;85%</td>
<td>133</td>
<td></td>
</tr>
<tr>
<td><strong>Classified security environment for computer and 25% of programs and data</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>289</td>
<td>156</td>
</tr>
<tr>
<td>yes</td>
<td>133</td>
<td></td>
</tr>
</tbody>
</table>
## Modeling and Measuring Resources

**Productivity Index**

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured programming</td>
<td>0-33% 34-66% &gt;66%</td>
<td>169 - 301 132</td>
</tr>
<tr>
<td>Design and code inspections</td>
<td>0-33% 34-66% &gt;66%</td>
<td>220 300 339 119</td>
</tr>
<tr>
<td>Top down development</td>
<td>0-33% 34-66% &gt;66%</td>
<td>196 237 321 125</td>
</tr>
<tr>
<td>Chief programmer team usage</td>
<td>0-33% 34-66% &gt;66%</td>
<td>219 - 408 189</td>
</tr>
<tr>
<td>Overall complexity of code developed</td>
<td>&lt;average &gt;average</td>
<td>314 185 129</td>
</tr>
<tr>
<td>Complexity of application processing</td>
<td>&lt;average average &gt;average</td>
<td>349 345 168 181</td>
</tr>
<tr>
<td>Question or Variable</td>
<td>Mean Productivity (DSL/MM)</td>
<td>Change (DSL/MM)</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Complexity of program flow</td>
<td>&lt;average 289</td>
<td>average 299</td>
</tr>
<tr>
<td>Overall constraints on program design</td>
<td>minimal 293</td>
<td>average 286</td>
</tr>
<tr>
<td>Program design constraints on main storage</td>
<td>minimal 391</td>
<td>average 277</td>
</tr>
<tr>
<td>Program design constraints on timing</td>
<td>minimal 303</td>
<td>average 317</td>
</tr>
<tr>
<td>Code for real-time or interactive operation, or executing under severe timing constraint</td>
<td>&lt;10% 279</td>
<td>10-40% 337</td>
</tr>
</tbody>
</table>
### Modeling and Measuring Resources
#### Productivity Index

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage of code for delivery</td>
<td>0-90% 159</td>
<td>91-99% 327</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code classified as non-mathematical application and I/O</td>
<td>0-33% 188</td>
<td>34-66% 311</td>
</tr>
<tr>
<td>formatting programs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of classes of items in the data base per 1000</td>
<td>0-15 334</td>
<td>16-80 243</td>
</tr>
<tr>
<td>lines of code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of pages of delivered documentation per 1000</td>
<td>0-32 320</td>
<td>33-88 252</td>
</tr>
<tr>
<td>lines of delivered code</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software Resource Models and Measures

Issues with the model

Contributions

• Empirical model based upon historical data
• Looks at the relationship between several variables
• Show the relationships between size and effort can be used for characterization, evaluation, and prediction
• Takes into account many variable classes, e.g., experience, methodology, customer interface, context
• Uses subjective metrics (ordinal scale)
• Shows the relations are not always monotonic
Software Resource Models and Measures

Issues with the model

Concerns
• Definition of values on an ordinal scale metrics
• What’s the underlying distribution for each metric
• There are lots of variables relative to the degrees of freedom, the data points
• The values for size or inconsistent across languages
• Doesn’t take into account the effect of combined variables
• Many of the variables are interdependent
• Correlation is not cause effect