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)?
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>Phase</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

**TRW IBM SEL**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>40% 35% 20% 21%</td>
</tr>
<tr>
<td>Code</td>
<td>20 30 45 28</td>
</tr>
<tr>
<td>Checkout/Test</td>
<td>40 25 28 23</td>
</tr>
<tr>
<td>Other</td>
<td>10 5 27</td>
</tr>
</tbody>
</table>

---

![effort diagram]
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
Software Resource Models and Measures

Static Single Variable Example

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: \( E = A \cdot \text{SIZE}^B \)

Effort Estimation Equation: \( E = 5.2L^{0.91} \)

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

Walston & Felix: IBM Federal Systems Division

Software Resource Models and Measures

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:

\( E = 5.2L^{0.91} \)
\( \text{DOC} = 49L^{1.01} \)
\( D = 4.1L^{0.36} \)
\( D = 2.47E^{0.35} \)
\( S = .54E^{0.6} \)

where
\( E \) = effort in staff months
\( L \) = lines of code in thousands
\( \text{DOC} \) = 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.

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, 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>less 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>
</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 programmers doing development who participated in design of functional specifications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; 25%</td>
<td>153</td>
<td>238</td>
</tr>
<tr>
<td>25-50%</td>
<td>242</td>
<td></td>
</tr>
<tr>
<td>&gt; 50%</td>
<td>391</td>
<td></td>
</tr>
<tr>
<td>Previous experience with operational computer minimal</td>
<td>146</td>
<td>166</td>
</tr>
<tr>
<td>average</td>
<td>270</td>
<td></td>
</tr>
<tr>
<td>extensive</td>
<td>312</td>
<td></td>
</tr>
<tr>
<td>Previous experience with programming languages minimal</td>
<td>122</td>
<td>263</td>
</tr>
<tr>
<td>average</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>extensive</td>
<td>385</td>
<td></td>
</tr>
<tr>
<td>Previous experience with application of similar or greater size and complexity minimal</td>
<td>146</td>
<td>264</td>
</tr>
<tr>
<td>average</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>extensive</td>
<td>410</td>
<td></td>
</tr>
<tr>
<td>Ratio of average staff size to duration (people/month) &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>Hardware under concurrent development no</td>
<td>297</td>
<td>120</td>
</tr>
<tr>
<td>yes</td>
<td>177</td>
<td></td>
</tr>
<tr>
<td>Development computer access, open under special request 0%</td>
<td>226</td>
<td>131</td>
</tr>
<tr>
<td>1-25%</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>&gt;25%</td>
<td>357</td>
<td></td>
</tr>
<tr>
<td>Development computer access, closed 0-10%</td>
<td>303</td>
<td>133</td>
</tr>
<tr>
<td>11-85%</td>
<td>251</td>
<td></td>
</tr>
<tr>
<td>&gt;85%</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>Classified security environment for computer and 25% of programs and data no</td>
<td>289</td>
<td>133</td>
</tr>
<tr>
<td>yes</td>
<td>156</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% 169</td>
<td>34-66% 301</td>
</tr>
<tr>
<td>Design and code inspections</td>
<td>0-33% 220</td>
<td>34-66% 300</td>
</tr>
<tr>
<td>Top down development</td>
<td>0-33% 196</td>
<td>34-66% 237</td>
</tr>
<tr>
<td>Chief programmer team usage</td>
<td>0-33% 219</td>
<td>34-66% 237</td>
</tr>
<tr>
<td>Overall complexity of code developed</td>
<td>&lt;average 314</td>
<td>&gt;average 185</td>
</tr>
<tr>
<td>Complexity of application processing</td>
<td>&lt;average 349</td>
<td>average 345</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>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% 91-99% 100%</td>
<td>159 327 265 106</td>
</tr>
<tr>
<td>Code classified as non-mathematical application and I/O formatting programs</td>
<td>0-33% 34-66% 67-100%</td>
<td>188 311 267 79</td>
</tr>
<tr>
<td>Number of classes of items in the data base per 1000 lines of code</td>
<td>0-15 16-80 &gt;80</td>
<td>334 243 193 141</td>
</tr>
<tr>
<td>Number of pages of delivered documentation per 1000 lines of delivered code</td>
<td>0-32 33-88 &gt;88</td>
<td>320 252 195 125</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