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**
    - consumes
    - produces
    - **Resource**
    - resource type
      - resource use
        - **Use Descriptors**
          - type
            - Availability
              - estimated
                - desirable
                - accessible
                - utilized
            - Incurrence
              - actual
                - Work nature, Calendar time, Measure of work
            - Type
              - Hardware, software, human, support plus attributes of the resource
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><strong>Sage</strong></td>
<td>39%</td>
<td>14%</td>
<td>47%</td>
</tr>
<tr>
<td><strong>NTDS</strong></td>
<td>30</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td><strong>Gemini</strong></td>
<td>36</td>
<td>17</td>
<td>47</td>
</tr>
<tr>
<td><strong>Saturn V</strong></td>
<td>32</td>
<td>24</td>
<td>44</td>
</tr>
<tr>
<td><strong>OS/360</strong></td>
<td>33</td>
<td>17</td>
<td>50</td>
</tr>
<tr>
<td><strong>TRW Survey</strong></td>
<td>46</td>
<td>20</td>
<td>34</td>
</tr>
</tbody>
</table>

---

### Software Resource Models and Measures

<table>
<thead>
<tr>
<th></th>
<th>TRW</th>
<th>IBM</th>
<th>SEL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design</strong></td>
<td>40%</td>
<td>35%</td>
<td>20%</td>
</tr>
<tr>
<td><strong>Code</strong></td>
<td>20</td>
<td>30</td>
<td>45</td>
</tr>
<tr>
<td><strong>Checkout/Test</strong></td>
<td>40</td>
<td>25</td>
<td>28</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>10</td>
<td>5</td>
<td>27</td>
</tr>
</tbody>
</table>

**Diagram:**

- ** effort**
- DR
- 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

Six Traditional Sizing Methods

- 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

E = A * SIZE^B

Walston & Felix: IBM Federal Systems Division

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, ...

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

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

Software Resource Models and Measures

Static Single Variable Example

Walston & Felix 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
Software Resource Models and Measures

Productivity Index

29 out of 68 variables showed a significantly high correlation with productivity. These can be used in estimating and are combined into an index:

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

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.

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Response Group</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer interface complexity</td>
<td>&lt;normal</td>
<td>500</td>
<td>376</td>
</tr>
<tr>
<td>User participation in the definition of</td>
<td>normal</td>
<td>295</td>
<td></td>
</tr>
<tr>
<td>requirements</td>
<td>&gt;normal</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>Customer originated program design changes</td>
<td>none</td>
<td>491</td>
<td>286</td>
</tr>
<tr>
<td>Customer experience with the application</td>
<td>some</td>
<td>267</td>
<td></td>
</tr>
<tr>
<td>area of the project</td>
<td>much</td>
<td>205</td>
<td></td>
</tr>
<tr>
<td>Overall personnel experience and qualifications</td>
<td>low</td>
<td>132</td>
<td>278</td>
</tr>
<tr>
<td></td>
<td>average</td>
<td>257</td>
<td></td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>410</td>
<td></td>
</tr>
</tbody>
</table>
### Modeling and Measuring Resources

#### Productivity Index

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Response Group</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Percentage of programmers doing development who participated in design of functional specifications</td>
<td>&lt; 25%</td>
<td>25-50%</td>
</tr>
<tr>
<td></td>
<td>153</td>
<td>242</td>
<td>391</td>
</tr>
<tr>
<td></td>
<td>Previous experience with operational computer</td>
<td>minimal</td>
<td>average</td>
</tr>
<tr>
<td></td>
<td>146</td>
<td>270</td>
<td>312</td>
</tr>
<tr>
<td></td>
<td>Previous experience with programming languages</td>
<td>minimal</td>
<td>average</td>
</tr>
<tr>
<td></td>
<td>122</td>
<td>225</td>
<td>385</td>
</tr>
<tr>
<td></td>
<td>Previous experience with application of similar or greater size and complexity</td>
<td>minimal</td>
<td>average</td>
</tr>
<tr>
<td></td>
<td>146</td>
<td>221</td>
<td>410</td>
</tr>
<tr>
<td></td>
<td>Ratio of average staff size to duration (people/month)</td>
<td>&lt;0.5</td>
<td>0.5-0.9</td>
</tr>
<tr>
<td></td>
<td>305</td>
<td>310</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>Hardware under concurrent development</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>297</td>
<td>177</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Development computer access, open under special request</td>
<td>0-10%</td>
<td>11-85%</td>
</tr>
<tr>
<td></td>
<td>226</td>
<td>274</td>
<td>357</td>
</tr>
<tr>
<td></td>
<td>Development computer access, closed</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>303</td>
<td>251</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>Classified security environment for computer and 25% of programs and data</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>289</td>
<td>156</td>
<td>133</td>
</tr>
</tbody>
</table>
### Modeling and Measuring Resources

#### Productivity Index

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Response Group</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structured programming</strong></td>
<td>0-33%</td>
<td>34-66%</td>
<td>66%</td>
</tr>
<tr>
<td></td>
<td>169</td>
<td>-</td>
<td>301</td>
</tr>
<tr>
<td><strong>Design and code inspections</strong></td>
<td>0-33%</td>
<td>34-66%</td>
<td>&gt;66%</td>
</tr>
<tr>
<td></td>
<td>220</td>
<td>300</td>
<td>339</td>
</tr>
<tr>
<td><strong>Top down development</strong></td>
<td>0-33%</td>
<td>34-66%</td>
<td>&gt;66%</td>
</tr>
<tr>
<td></td>
<td>196</td>
<td>237</td>
<td>321</td>
</tr>
<tr>
<td><strong>Chief programmer team usage</strong></td>
<td>0-33%</td>
<td>34-66%</td>
<td>&gt;66%</td>
</tr>
<tr>
<td></td>
<td>219</td>
<td>-</td>
<td>408</td>
</tr>
<tr>
<td><strong>Overall complexity of code developed</strong></td>
<td>&lt;average</td>
<td>average</td>
<td>&gt;average</td>
</tr>
<tr>
<td></td>
<td>314</td>
<td>300</td>
<td>339</td>
</tr>
<tr>
<td><strong>Complexity of application processing</strong></td>
<td>&lt;average</td>
<td>average</td>
<td>&gt;average</td>
</tr>
<tr>
<td></td>
<td>349</td>
<td>345</td>
<td>168</td>
</tr>
</tbody>
</table>

### Modeling and Measuring Resources

#### Productivity Index

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Response Group</th>
<th>Mean Productivity (DSL/MM)</th>
<th>Change (DSL/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complexity of program flow</strong></td>
<td>&lt;average</td>
<td>average</td>
<td>&gt;average</td>
</tr>
<tr>
<td></td>
<td>289</td>
<td>299</td>
<td>209</td>
</tr>
<tr>
<td><strong>Overall constraints on program design</strong></td>
<td>minimal</td>
<td>average</td>
<td>severe</td>
</tr>
<tr>
<td></td>
<td>293</td>
<td>286</td>
<td>166</td>
</tr>
<tr>
<td><strong>Program design constraints on main storage</strong></td>
<td>minimal</td>
<td>average</td>
<td>severe</td>
</tr>
<tr>
<td></td>
<td>391</td>
<td>277</td>
<td>193</td>
</tr>
<tr>
<td><strong>Program design constraints on timing</strong></td>
<td>minimal</td>
<td>average</td>
<td>severe</td>
</tr>
<tr>
<td></td>
<td>303</td>
<td>317</td>
<td>171</td>
</tr>
<tr>
<td><strong>Code for real-time or interactive operation, or executing under severe timing constraint</strong></td>
<td>&lt;10%</td>
<td>10-40%</td>
<td>&gt;40%</td>
</tr>
<tr>
<td></td>
<td>279</td>
<td>337</td>
<td>203</td>
</tr>
</tbody>
</table>
## Modeling and Measuring Resources

### Productivity Index

<table>
<thead>
<tr>
<th>Question or Variable</th>
<th>Response Group</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%</td>
<td>159</td>
<td>265</td>
</tr>
<tr>
<td></td>
<td>91-99%</td>
<td>327</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>Code classified as non-mathematical application and I/O formatting programs</td>
<td>0-33%</td>
<td>188</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>34-66%</td>
<td>311</td>
<td></td>
</tr>
<tr>
<td></td>
<td>67-100%</td>
<td>267</td>
<td></td>
</tr>
<tr>
<td>Number of classes of items in the data base per 1000 lines of code</td>
<td>0-15</td>
<td>334</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td>16-80</td>
<td>243</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&gt;80</td>
<td>193</td>
<td></td>
</tr>
<tr>
<td>Number of pages of delivered documentation per 1000 lines of delivered code</td>
<td>0-32</td>
<td>320</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>33-88</td>
<td>252</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&gt;88</td>
<td>195</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
- 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