Object Points
4th Generation Languages

• A set of high level languages (each “generation” moves away from machine code)

• They can be distinguished between:
  – Data management tools: that allows sophisticated functions for data manipulation (examples: SAS, SPSS, Mathematica, Matlab, ABAP)
  – Report and form generation languages, that specify the data format and the output to be generated (e.g. SQL, Oracle Forms)
  – Automatic code generators from CASE tools (e.g. Rails)
Object Points

- Function-related metric for 4th Generation Languages
- Computation is simpler than FP

- Some references:
Object Points Computation

• Similar to FP

• Compute the number of screens and classify them as simple, medium, complex

• Compute the number of reports and classify them as simple, medium, complex

• Count the number of modules that have to be developed

• Use weight matrices to sum the values above, taking into account reused code

• A formula translates OPs into productivity measures
## Screen and Report Classification

### Object point complexity levels for **screens**

<table>
<thead>
<tr>
<th>Number of Views Contained</th>
<th>Total &lt; 4</th>
<th>Total &lt; 8</th>
<th>Total 8+</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;3</td>
<td>simple</td>
<td>simple</td>
<td>medium</td>
</tr>
<tr>
<td>3-7</td>
<td>simple</td>
<td>medium</td>
<td>difficult</td>
</tr>
<tr>
<td>8+</td>
<td>medium</td>
<td>difficult</td>
<td>difficult</td>
</tr>
</tbody>
</table>

### Object point complexity levels for **reports**

<table>
<thead>
<tr>
<th>Number of Sections Contained</th>
<th>Total &lt; 4</th>
<th>Total &lt; 8</th>
<th>Total 8+</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>simple</td>
<td>simple</td>
<td>medium</td>
</tr>
<tr>
<td>2-3</td>
<td>simple</td>
<td>medium</td>
<td>difficult</td>
</tr>
<tr>
<td>4+</td>
<td>medium</td>
<td>difficult</td>
<td>difficult</td>
</tr>
</tbody>
</table>

Source: [http://yunus.hacettepe.edu.tr/~sencer/objectp.html](http://yunus.hacettepe.edu.tr/~sencer/objectp.html)
Object Points Computation

\[ NOP = \sum_{i=1}^{3} \left[ \begin{array}{ccc} k_i^s & k_i^m & k_i^d \end{array} \right] \cdot \left[ \begin{array}{c} n_i^s \\ n_i^m \\ n_i^d \end{array} \right] \cdot \frac{(100 - r)}{100} \]

\[ EFFORT = \frac{NOP}{PROD} \]

where:

\( r \) = percentage of components reused

\( PROD \) = productivity
Productivity estimates

• Productivity **between 4 and 50 object points/month**, depending on tool support and developer capability:

<table>
<thead>
<tr>
<th>Developer Experience</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROD</td>
<td>4</td>
<td>7</td>
<td>13</td>
<td>25</td>
<td>50</td>
</tr>
</tbody>
</table>

Adapted from: [http://yunus.hacettepe.edu.tr/~sencer/objectp.html](http://yunus.hacettepe.edu.tr/~sencer/objectp.html)

• Other works indicates up to **75 OP/MM**
Remarks

• Simple to estimate
  (remember Joel Henry’s approach to FP computation)

• OP takes into account reused code
  (something which will also appear in some elaborations of
  the COCOMO model)