REASONS FOR SUCCESS OF FUNCTION POINTS

• Function points match standard economic definitions for productivity analysis:

  “Goods or services produced per unit of labor or expense.”

• Function points do not distort quality and productivity as do “Lines of Code” or LOC metrics.

• Function points support activity-based cost analysis, baselines, benchmarks, quality, cost, and value studies.
REASONS FOR SUCCESS OF FUNCTION POINTS

• Lines of code metrics penalize high-level programming languages.

• If used for economic studies with more than one language LOC metrics should be considered professional malpractice.

• Cost per defect metrics penalize quality and make buggy software look best. For quality economic studies cost per defect metrics are invalid. Function points are best.

• Function point metrics have the widest range of use of any software metric in history: they work for both economic and quality analyses.
MAJOR FUNCTION POINT USES CIRCA 2008

• Function points are now a standard **sizing** metric.

• Function points are now a standard **productivity** metric.

• Function points are now a powerful **quality** metric.

• Function points are now a powerful **schedule** metric.

• Function points are now a powerful **staffing** metric.

• Function points are now used in software **litigation**.

• Function points are now used for **outsourcing contracts**.

• Function points can be used for **cost analysis** (with care).

• Function points can be used for **value analysis** (with care.)
NEW FUNCTION POINT USES CIRCA 2009

- Function points used for portfolio analysis.
- Function points used for backlog analysis.
- Function points used for risk analysis.
- Function points used for real-time requirements changes.
- Function points used for software usage studies.
- Function points used for delivery analysis.
- Function points used for COTS analysis.
- Function points used for occupation group analysis.
- Function points used for maintenance and ITIL analysis.
FUNCTION POINTS FROM 2009 TO 2019

- Resolve functional vs. technical requirements issues.
- Resolve overhead, inflation, and cost issues.
- Resolve global variations in work hours and work days.
- Resolve issue of > 90 software occupation groups
- Produce conversion rules for function point variations.
- Improve cost, speed, and timing of initial sizing.
- Develop and certify “micro function points.”
- Expand benchmarks to > 25,000 projects.
INDUSTRY EVOLUTION CIRCA 2008-2018

• Moving from software development to software delivery

• Development rates < 25 function points per staff month

• Delivery rates > 500 function points per staff month

• Delivery issues: reuse taxonomy, quality, security, bandwidth

• Delivery methods:

Service Oriented Architecture (SOA)
Software as a Service (SaaS)
Commercial reusable libraries
Cloud computing
FUNCTION POINT EVOLUTION CIRCA 2008-2018

• Measure **delivered features** as well as development.

• Measure and **catalog** reusable features using formal taxonomy.

• Measure **deployment, installation, and usage**.

• Measure **quality** and **security**.

• New measurements needed:

  Sizes of reusable components and COTS packages
  Quality, security of reusable components
  Sizes of delivered applications
  Deployment and usage of delivered applications
MIGRATION TO FUNCTIONAL METRICS

Measurements Based on IFPUG Function Points
Measurements Based on Lines of Source Code
GQM measures are also expanding rapidly

Expansion of function point metrics is urgently needed!

LOC usage: Professional Malpractice

No Measurements At All

Percentage of U.S. Software Producers

Calendar Year

FUNCTIONAL METRICS IN INDUSTRY

PRODUCTION STUDIES

- Software Projects
- Software Portfolios

- 100 to 100,000 function points
- 1,500,000 to 15,000,000 function points

VALUE ANALYSIS AND USAGE STUDIES

- Individual Users & Value
  - 30,000 function points
  - Managers
  - Engineers
  - Salesmen
  - Administrators
  - Supervisors
  - Purchasing

- Organizational Users & Value
  - 500,000 function points
  - Engineering
  - Marketing & Sales
  - Manufacturing
  - Purchasing

- Enterprise Usage & Value
  - 1,000,000 to 20,000,000 function points

- Sizing
- Productivity
- Quality
- Schedules
- Costs
- Size
- Replacement Cost
- Productivity
- Quality

Copyright © 2008. All Rights Reserved.
## ESTIMATED U.S. SOFTWARE CIRCA 2008

<table>
<thead>
<tr>
<th>Size of Application</th>
<th>Number of Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>630,000</td>
</tr>
<tr>
<td>10</td>
<td>320,000</td>
</tr>
<tr>
<td>100</td>
<td>131,500</td>
</tr>
<tr>
<td>1,000</td>
<td>70,500</td>
</tr>
<tr>
<td>10,000</td>
<td>21,750</td>
</tr>
<tr>
<td>100,000</td>
<td>185</td>
</tr>
<tr>
<td>1,000,000</td>
<td>30</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1,173,965</td>
</tr>
</tbody>
</table>

- Small updates are most numerous.
- Large applications are most hazardous and expensive.
## ESTIMATED FUNCTION POINT PENETRATION

<table>
<thead>
<tr>
<th>Size of Application</th>
<th>Function Point Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>10</td>
<td>0.00%</td>
</tr>
<tr>
<td>100</td>
<td>25.00%</td>
</tr>
<tr>
<td>1,000</td>
<td>15.00%</td>
</tr>
<tr>
<td>10,000</td>
<td>5.00%</td>
</tr>
<tr>
<td>100,000</td>
<td>0.00%</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

- Function point rules do not work < 15 function points.
- Counting is too slow and costly > 15,000 function points.
**ESTIMATED PROJECTS COUNTED CIRCA 2008**

<table>
<thead>
<tr>
<th>Size of Application</th>
<th>Function Point Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>32,875</td>
</tr>
<tr>
<td>1,000</td>
<td>10,575</td>
</tr>
<tr>
<td>10,000</td>
<td>652</td>
</tr>
<tr>
<td>100,000</td>
<td>0</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>44,102</strong></td>
</tr>
</tbody>
</table>

- Function point usage declines with application size.
- Function points are not used < 15 function point size limit.
FUNCTION POINT EXPANSION CIRCA 2009

Function points need to extend their range of use:

- “Micro function points” are needed for small updates below 15 function points in size.
- About 20% of all software effort goes to small updates.
- Function point costs and speed need improvement to work effectively > 15,000 function points in size.
- More than 45% of all effort goes to large applications.
- Function point range should run from 1 to 1,000,000 function points.
### ONE YEAR FOR 10,000 FUNCTION POINTS

Starting size = 10,000 function points

<table>
<thead>
<tr>
<th>Task</th>
<th>Size (FP)</th>
<th>Staff</th>
<th>Effort</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large</td>
<td>500</td>
<td>3.33</td>
<td>33.33</td>
<td>$250,000</td>
</tr>
<tr>
<td>Small</td>
<td>90</td>
<td>1.00</td>
<td>5.29</td>
<td>$39,706</td>
</tr>
<tr>
<td>V. Small</td>
<td>180*</td>
<td>1.20</td>
<td>10.59</td>
<td>$79,412</td>
</tr>
<tr>
<td>Bugs</td>
<td>1750*</td>
<td>7.29</td>
<td>87.50</td>
<td>$656,750</td>
</tr>
<tr>
<td>TOTAL</td>
<td>2520</td>
<td>15.00</td>
<td>136.72</td>
<td>$1,025,368</td>
</tr>
</tbody>
</table>

* Can’t be measured using standard function points
THE NEED FOR “MICRO FUNCTION POINTS”

About 70% of small updates can’t be measured

Small changes run from 0.001 to 15 FP

Each change is small, but total > 25,000 per year

Total volume per year > 100,000 function points

Micro function points are needed to expand usage:

• New counting rules with negative adjustments
• Backfiring
• Pattern matching
FUNCTION POINT COUNTING CIRCA 2008

Speed of count = 400 to 600 FP per day

Consulting cost = $3,000 per day

Cost range = $5.00 to $7.50 per function point

- High costs and low speed limit usefulness of function points to small or mid-sized software applications
## FUNCTION POINT EXPANSION CIRCA 2009

<table>
<thead>
<tr>
<th>Daily Counting Speed</th>
<th>Percent of Projects Counted</th>
<th>Maximum Size Counted</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>20%</td>
<td>15,000</td>
</tr>
<tr>
<td>1,000</td>
<td>30%</td>
<td>25,000</td>
</tr>
<tr>
<td>5,000</td>
<td>45%</td>
<td>50,000</td>
</tr>
<tr>
<td>10,000</td>
<td>75%</td>
<td>75,000</td>
</tr>
<tr>
<td>50,000</td>
<td>100%</td>
<td>350,000</td>
</tr>
</tbody>
</table>
### MAJOR APPLICATION SIZES CIRCA 2009

<table>
<thead>
<tr>
<th>Applications</th>
<th>Approximate Size in Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars Missile Defense</td>
<td>350,000</td>
</tr>
<tr>
<td>ERP (SAP, Oracle, etc.)</td>
<td>300,000</td>
</tr>
<tr>
<td>Microsoft Vista</td>
<td>159,000</td>
</tr>
<tr>
<td>Microsoft Office</td>
<td>98,000</td>
</tr>
<tr>
<td>Airline Reservations</td>
<td>50,000</td>
</tr>
<tr>
<td>NASA space shuttle</td>
<td>25,000</td>
</tr>
</tbody>
</table>

- Major applications are too large for low-speed counting
## ESTIMATED SOFTWARE FAILURE RATES

<table>
<thead>
<tr>
<th>Size of Application</th>
<th>% Cancelled or delayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00%</td>
</tr>
<tr>
<td>10</td>
<td>0.01%</td>
</tr>
<tr>
<td>100</td>
<td>1.00%</td>
</tr>
<tr>
<td>1,000</td>
<td>10.00%</td>
</tr>
<tr>
<td>10,000</td>
<td>35.00%</td>
</tr>
<tr>
<td>100,000</td>
<td>75.00%</td>
</tr>
<tr>
<td>1,000,000</td>
<td>95.00%</td>
</tr>
</tbody>
</table>

- Major applications fail more often than they succeed!
- Risk of failure correlates with increasing application size!
- Major applications need fast, early function point analysis!
RESULTS FOR 10,000 FUNCTION POINTS

- No formal risk analysis = 35% chance of failure
- Risk analysis after requirements = 15% chance of failure
- Risk analysis before requirements = 5% chance of failure

- Early sizing and formal risk analysis raise the chances of success for large software projects.
FUNCTION POINTS AND REQUIREMENTS

Requirements change at 1% to 3% per month during development.

Requirements change at 5% to 8% per year after deployment.

Some large systems are used for more than 25 years.

Size at end of requirements = 10,000 Function points
Size at first delivery = 13,000 “ “
Size after 5 years of usage = 18,000 “ “
Size after 25 years of usage = 25,000 “ “

Sizing should be continuous from requirements to retirement.

Continuous sizing needs low cost, high speed function points.
FUNCTION POINTS AND ECONOMIC STUDIES

Function points are the only metric that can study all Software activities without exception.

There are two key dimensions when applying function points to economic analysis:

**Assignment Scope**

“The amount of work assigned to a specific employee.”

**Production Rate**

“The amount of work completed in a specific time period.”
## ASSIGNMENT AND PRODUCTION EXAMPLES

Application = 1,000 Function points

<table>
<thead>
<tr>
<th>Activity</th>
<th>Assignment Scope</th>
<th>Production Rate (Monthly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>500</td>
<td>200</td>
</tr>
<tr>
<td>Design</td>
<td>500</td>
<td>175</td>
</tr>
<tr>
<td>Coding</td>
<td>200</td>
<td>25</td>
</tr>
<tr>
<td>Testing</td>
<td>150</td>
<td>40</td>
</tr>
<tr>
<td>Documentation</td>
<td>1000</td>
<td>150</td>
</tr>
<tr>
<td>Management</td>
<td>1000</td>
<td>40</td>
</tr>
<tr>
<td><strong>OVERALL</strong></td>
<td><strong>222</strong></td>
<td><strong>9.25</strong></td>
</tr>
</tbody>
</table>
**ASSIGNMENT AND PRODUCTION EXAMPLES**

Application = 1,000 Function points

<table>
<thead>
<tr>
<th>Activity</th>
<th>Staff</th>
<th>Effort (months)</th>
<th>Schedule (months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>2</td>
<td>5</td>
<td>2.5</td>
</tr>
<tr>
<td>Design</td>
<td>2</td>
<td>6</td>
<td>3.0</td>
</tr>
<tr>
<td>Coding</td>
<td>5</td>
<td>40</td>
<td>8.0</td>
</tr>
<tr>
<td>Testing</td>
<td>7</td>
<td>25</td>
<td>7.0</td>
</tr>
<tr>
<td>Documentation</td>
<td>1</td>
<td>7</td>
<td>7.0</td>
</tr>
<tr>
<td>Management</td>
<td>1</td>
<td>25</td>
<td>24.0</td>
</tr>
<tr>
<td><strong>OVERALL</strong></td>
<td>4.5</td>
<td>108</td>
<td>24.0</td>
</tr>
</tbody>
</table>
USEFUL FUNCTION POINT RULES OF THUMB

• Function points $^0.40$ power = calendar months in schedule
• Function points $^1.15$ power = pages of paper documents
• Function points $^1.20$ power = number of test cases
• Function points $^1.25$ power = software defect potential
• Function points / 150 = development technical staff
• Function points / 1,500 = maintenance technical staff

NOTE: These rules assume IFPUG Version 4.2 counting rules.
# FUNCTION POINTS AND USAGE ANALYSIS

<table>
<thead>
<tr>
<th>Occupation</th>
<th>Function Points Available</th>
<th>Software Packages</th>
<th>Daily usage (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Military planners</td>
<td>5,000,000</td>
<td>30</td>
<td>7.5</td>
</tr>
<tr>
<td>2 Physicians</td>
<td>3,000,000</td>
<td>20</td>
<td>3.0</td>
</tr>
<tr>
<td>3 FBI Agents</td>
<td>1,500,000</td>
<td>15</td>
<td>3.5</td>
</tr>
<tr>
<td>4 Attorneys</td>
<td>325,000</td>
<td>10</td>
<td>4.0</td>
</tr>
<tr>
<td>5 Air-traffic controllers</td>
<td>315,000</td>
<td>3</td>
<td>8.5</td>
</tr>
<tr>
<td>6 Accountants</td>
<td>175,000</td>
<td>10</td>
<td>5.0</td>
</tr>
<tr>
<td>7 Pharmacists</td>
<td>150,000</td>
<td>6</td>
<td>4.0</td>
</tr>
<tr>
<td>8 Electrical engineers</td>
<td>100,000</td>
<td>20</td>
<td>5.5</td>
</tr>
<tr>
<td>9 Software engineers</td>
<td>50,000</td>
<td>20</td>
<td>7.0</td>
</tr>
<tr>
<td>10 Project managers</td>
<td>35,000</td>
<td>10</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Computers and software are now vital tools for all knowledge-based occupations. Usage contributes to software value.
### PERSONAL FUNCTION POINTS

<table>
<thead>
<tr>
<th>Products</th>
<th>Circa 2008 Available</th>
<th>Circa 2018 Available</th>
<th>Daily usage (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Home computer</td>
<td>1,000,000</td>
<td>2,000,000</td>
<td>2.5</td>
</tr>
<tr>
<td>2 Automobile</td>
<td>300,000</td>
<td>750,000</td>
<td>3.0</td>
</tr>
<tr>
<td>3 Smart appliances</td>
<td>100,000</td>
<td>750,000</td>
<td>24.0</td>
</tr>
<tr>
<td>4 Televisions</td>
<td>25,000</td>
<td>125,000</td>
<td>4.0</td>
</tr>
<tr>
<td>5 Home alarms</td>
<td>5,000</td>
<td>15,000</td>
<td>24.0</td>
</tr>
<tr>
<td>6 Home music</td>
<td>7,500</td>
<td>20,000</td>
<td>2.5</td>
</tr>
<tr>
<td>7 I-Phone</td>
<td>20,000</td>
<td>30,000</td>
<td>3.0</td>
</tr>
<tr>
<td>8 Digital camera</td>
<td>2,000</td>
<td>5,000</td>
<td>0.5</td>
</tr>
<tr>
<td>9 Electronic books</td>
<td>10,000</td>
<td>20,000</td>
<td>2.5</td>
</tr>
<tr>
<td>10 Social networks</td>
<td>25,000</td>
<td>75,000</td>
<td>2.5</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>1,494,500</strong></td>
<td><strong>3,790,000</strong></td>
<td><strong>20.5</strong></td>
</tr>
</tbody>
</table>

Personal software is expanding rapidly in all developed countries.
## FUNCTION POINTS AND MANAGEMENT TOOLS

<table>
<thead>
<tr>
<th>SELECTED TOOLS</th>
<th>Lagging</th>
<th>Average</th>
<th>Leading</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Project planning</td>
<td>1,000</td>
<td>1,500</td>
<td>3,000</td>
</tr>
<tr>
<td>2 Cost estimating</td>
<td>--</td>
<td>300</td>
<td>3,000</td>
</tr>
<tr>
<td>3 Project office</td>
<td>--</td>
<td>--</td>
<td>3,000</td>
</tr>
<tr>
<td>4 Statistical analysis</td>
<td>--</td>
<td>750</td>
<td>3,000</td>
</tr>
<tr>
<td>5 Personnel mgt.</td>
<td>500</td>
<td>1,000</td>
<td>2,000</td>
</tr>
<tr>
<td>6 Quality estimating</td>
<td>--</td>
<td>--</td>
<td>2,000</td>
</tr>
<tr>
<td>7 Process Assessment</td>
<td>--</td>
<td>500</td>
<td>2,000</td>
</tr>
<tr>
<td>8 Risk analysis</td>
<td>--</td>
<td>--</td>
<td>1,500</td>
</tr>
<tr>
<td>9 Value analysis</td>
<td>--</td>
<td>250</td>
<td>1,500</td>
</tr>
<tr>
<td>10 Department budgets</td>
<td>500</td>
<td>700</td>
<td>1,000</td>
</tr>
</tbody>
</table>

<p>| TOTALS                       | 2,000   | 5,000   | 22,000  |
| Tools used                   | 3       | 7       | 10      |</p>
<table>
<thead>
<tr>
<th>SELECTED AREAS</th>
<th>Lagging</th>
<th>Average</th>
<th>Leading</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Project mgt.</td>
<td>2,000</td>
<td>5,000</td>
<td>22,000</td>
</tr>
<tr>
<td>2 Quality Assur.</td>
<td>0</td>
<td>2,500</td>
<td>10,000</td>
</tr>
<tr>
<td>3 Development</td>
<td>14,000</td>
<td>19,000</td>
<td>25,000</td>
</tr>
<tr>
<td>4 Maintenance</td>
<td>2,500</td>
<td>10,500</td>
<td>20,000</td>
</tr>
<tr>
<td>5 Testing</td>
<td>2,000</td>
<td>4,000</td>
<td>15,000</td>
</tr>
<tr>
<td>6 Documentation</td>
<td>3,000</td>
<td>5,000</td>
<td>12,000</td>
</tr>
<tr>
<td>7 Training</td>
<td>2,000</td>
<td>4,500</td>
<td>10,000</td>
</tr>
<tr>
<td>8 Nationalization</td>
<td>0</td>
<td>3,000</td>
<td>7,500</td>
</tr>
<tr>
<td>9 Customer supp.</td>
<td>2,500</td>
<td>5,500</td>
<td>10,500</td>
</tr>
<tr>
<td>10 Change control</td>
<td>3,000</td>
<td>6,000</td>
<td>12,000</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>31,000</td>
<td>65,000</td>
<td>144,000</td>
</tr>
<tr>
<td>Tools used</td>
<td>20</td>
<td>35</td>
<td>75</td>
</tr>
</tbody>
</table>
## PORTFOLIO FUNCTION POINT ANALYSIS

### Size in terms of IFPUG function points

<table>
<thead>
<tr>
<th>Unit</th>
<th>In-house</th>
<th>COTS</th>
<th>Open-Source</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headquarters</td>
<td>500,000</td>
<td>500,000</td>
<td>50,000</td>
<td>1,050,000</td>
</tr>
<tr>
<td>Sales/Market.</td>
<td>750,000</td>
<td>750,000</td>
<td>75,000</td>
<td>1,575,000</td>
</tr>
<tr>
<td>Manufacture.</td>
<td>1,500,000</td>
<td>500,000</td>
<td>100,000</td>
<td>2,100,000</td>
</tr>
<tr>
<td>Acct/Finance</td>
<td>500,000</td>
<td>1,000,000</td>
<td>10,000</td>
<td>1,510,000</td>
</tr>
<tr>
<td>Research</td>
<td>750,000</td>
<td>250,000</td>
<td>150,000</td>
<td>1,150,000</td>
</tr>
<tr>
<td>Cust. Support</td>
<td>300,000</td>
<td>250,000</td>
<td>50,000</td>
<td>600,000</td>
</tr>
<tr>
<td>Human Res.</td>
<td>100,000</td>
<td>250,000</td>
<td></td>
<td>350,000</td>
</tr>
<tr>
<td>Purchasing</td>
<td>250,000</td>
<td>500,000</td>
<td></td>
<td>750,000</td>
</tr>
<tr>
<td>Legal</td>
<td>200,000</td>
<td>750,000</td>
<td></td>
<td>950,000</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>4,850,000</td>
<td>4,750,000</td>
<td>435,000</td>
<td>10,035,000</td>
</tr>
</tbody>
</table>
Software portfolios are major business assets.

Software portfolios are large and growing: > 10,000,000 FP

Portfolios include:

- IT, systems, embedded, and web applications
- In-house applications
- Outsourced applications
- Commercial applications (COTS)
- Open-source applications
- SOA and SaaS
Usage analysis is a key factor in value analysis.

Portfolio analysis is a key factor in market planning.

Portfolio analysis is a key factor in both corporate and government operations.

When function points are used for development, delivery, maintenance, usage studies, and portfolio analysis they will become a major metric for global economic studies.

Usage analysis and portfolio analysis deal with millions of Function points. Speed and cost of counting are barriers that must be overcome.
## U. S. SOFTWARE QUALITY AVERAGES

(Defects per Function Point)

<table>
<thead>
<tr>
<th></th>
<th>System Software</th>
<th>Commercial Software</th>
<th>Information Software</th>
<th>Military Software</th>
<th>Overall Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect Potentials</td>
<td>6.0</td>
<td>5.0</td>
<td>4.5</td>
<td>7.0</td>
<td>5.6</td>
</tr>
<tr>
<td>Defect Removal Efficiency</td>
<td>94%</td>
<td>90%</td>
<td>73%</td>
<td>96%</td>
<td>88%</td>
</tr>
<tr>
<td>Delivered Defects</td>
<td>0.4</td>
<td>0.5</td>
<td>1.2</td>
<td>0.3</td>
<td>0.65</td>
</tr>
<tr>
<td>First Year Discovery Rate</td>
<td>65%</td>
<td>70%</td>
<td>30%</td>
<td>75%</td>
<td>60%</td>
</tr>
<tr>
<td>First Year Reported Defects</td>
<td>0.26</td>
<td>0.35</td>
<td>0.36</td>
<td>0.23</td>
<td>0.30</td>
</tr>
</tbody>
</table>
SOFTWARE QUALITY RANGES

Defects per FP

Defect Removal Efficiency

Malpractice

U.S. Average

Best in Class
### U.S. AVERAGES FOR SOFTWARE QUALITY

(Data expressed in terms of defects per function point)

<table>
<thead>
<tr>
<th>Defect Origins</th>
<th>Defect Potential</th>
<th>Removal Efficiency</th>
<th>Delivered Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1.00</td>
<td>77%</td>
<td>0.23</td>
</tr>
<tr>
<td>Design</td>
<td>1.25</td>
<td>85%</td>
<td>0.19</td>
</tr>
<tr>
<td>Coding</td>
<td>1.75</td>
<td>95%</td>
<td>0.09</td>
</tr>
<tr>
<td>Documents</td>
<td>0.60</td>
<td>80%</td>
<td>0.12</td>
</tr>
<tr>
<td>Bad Fixes</td>
<td>0.40</td>
<td>70%</td>
<td>0.12</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>5.00</strong></td>
<td><strong>85%</strong></td>
<td><strong>0.75</strong></td>
</tr>
</tbody>
</table>

*(Function points show all defect sources - not just coding defects)*
# BEST IN CLASS SOFTWARE QUALITY

(Data expressed in terms of defects per function point)

<table>
<thead>
<tr>
<th>Defect Origins</th>
<th>Defect Potential</th>
<th>Removal Efficiency</th>
<th>Delivered Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>0.40</td>
<td>85%</td>
<td>0.08</td>
</tr>
<tr>
<td>Design</td>
<td>0.60</td>
<td>97%</td>
<td>0.02</td>
</tr>
<tr>
<td>Coding</td>
<td>1.00</td>
<td>99%</td>
<td>0.01</td>
</tr>
<tr>
<td>Documents</td>
<td>0.40</td>
<td>98%</td>
<td>0.01</td>
</tr>
<tr>
<td>Bad Fixes</td>
<td>0.10</td>
<td>95%</td>
<td>0.01</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>2.50</strong></td>
<td><strong>96%</strong></td>
<td><strong>0.13</strong></td>
</tr>
</tbody>
</table>

**OBSERVATIONS**

- Most often found above Level 3 on the SEI CMM scale
- Most often found in systems and military software
THIRTEEN SOFTWARE METRICS CRITERIA

A Useful Software Metric Should:

1. Be standardized
2. Be unambiguous
3. Have a formal user group
4. Have adequate published data
5. Have tools available for new projects
6. Have tools available for legacy projects
7. Have conversion rules for related metrics
8. Deal with all deliverables
9. Support all kinds of software
10. Support all programming languages plus mixed
11. Support all sizes of applications
12. Support new + reused artifacts
13. Be cost-effective and fast to apply
CRITERIA FOR SOFTWARE METRICS SELECTION

IFPUG Function Point Metric Meets 11 of 13 Criteria:

1. Are standardized
2. Are unambiguous
3. Has a formal user group
4. Has adequate published data
5. Has tools available for new projects
6. Has tools available for legacy projects
7. Has conversion rules for related metrics
8. Deals with all deliverables
9. Supports all kinds of software
10. Supports all programming languages plus mixed languages
11. Counting inaccurate < 15 function points
12. Can be used with reused artifacts
13. Are somewhat costly and slow to apply
CRITERIA FOR SOFTWARE METRICS SELECTION

Pattern Matching Meets 9 of 13 Criteria:

1. Are not standardized
2. Are unambiguous (with exceptions)
3. Has no formal user group
4. Has little published data
5. Has tools available for new projects
6. Has tools available for legacy projects
7. Has conversion rules for related metrics
8. Deals with all deliverables
9. Supports all kinds of software
10. Supports all programming languages plus mixed languages
11. Counting accurate < 15 function points
12. Can be used with reused artifacts
13. Is very fast and inexpensive, but of uncertain accuracy
CRITERIA FOR SOFTWARE METRICS SELECTION

Legacy Function Point Metrics Meets 8 of 13 Criteria:

1. Are not standardized
2. Are unambiguous
3. Has no formal user group
4. Has no published data
5. Has no tools available for new projects
6. Has tools available for legacy projects
7. Has conversion rules for related metrics
8. Deals with all deliverables
9. Supports primarily IT projects; not embedded software
10. Supports some legacy programming languages
11. Counting inaccurate < 15 function points
12. Can be used with reused artifacts
13. Is very fast and inexpensive
CRITERIA FOR SOFTWARE METRICS SELECTION

Function Point Variants (COSMIC etc.) meet 7 criteria:

1. Are standardized (conflicting standards exist)
2. Are unambiguous (with exceptions)
3. Have formal user groups
4. Lack adequate published data
5. Have tools available for new projects
6. Have tools available for legacy projects
7. Lack conversion rules for related metrics
8. Deal with all deliverables
9. Support all kinds of software
10. Support all programming languages plus mixed languages
11. Counting inaccurate < 15 function points
12. Can be used with reused artifacts
13. Are not particularly fast or inexpensive
CRITERIA FOR SOFTWARE METRICS SELECTION

The Backfiring Method Meets 7 of 13 Criteria:

1. Is not standardized
2. Is very ambiguous
3. Has no formal user groups
4. Has inconsistent published data
5. Has no tools available until code is complete
6. Has tools available for legacy projects
7. Has conversion rules for related metrics
8. Deals with all deliverables
9. Supports all kinds of software
10. Supports procedural programming languages; plus mixed
11. Can be used < 15 function point size limit
12. Can be used for reused artifacts
13. Is very fast and inexpensive
CRITERIA FOR SOFTWARE METRICS SELECTION

The Logical Statement Metric Meets 6 of 13 Criteria:

1. **Is not** standardized (conflicts between IEEE, SPR, and SEI)
2. **Is somewhat** ambiguous and may be incorrect
3. **Does not** have a formal user group
4. **Does not** have adequate published data (data is misleading)
5. Has tools available for new projects
6. Has tools available for legacy projects
7. Has conversion rules for related metrics
8. **Does not** deal with all deliverables
9. Does support all kinds of software
10. **Supports procedural** programming languages; mixed is possible
11. Does support all application sizes
12. Can be used with reusable code artifacts
13. **Is fast, but difficult to count accurately without automated tools**
The Physical LOC Metric Meets 5 of 13 Criteria:

1. Is not standardized
2. Is very ambiguous with multiple languages and is often incorrect
3. Does not have a formal user group
4. Does not have adequate published data (data is misleading)
5. Has tools available for new projects
6. Has tools available for legacy projects
7. Does not have conversion rules for related metrics
8. Does not deal with all deliverables
9. Does support all kinds of software
10. Does not support all or mixed programming languages (no VB)
11. Does support all sizes of software
12. Hard to use with reusable code artifacts
13. Is fast but difficult to count accurately without automated tools
CRITERIA FOR SOFTWARE METRICS SELECTION

Goal-Question Metrics Meet 3 of 13 Criteria:

1. **Is not** standardized
2. Is often unambiguous
3. **Does not** have a formal user group
4. Does have adequate published data
5. **Does not** have tools available for new projects
6. **Does not** have tools available for legacy projects
7. **Does not** have conversion rules for related metrics
8. Can be designed for use with all deliverables
9. **Does not** support all kinds of software
10. **Does not** support all or mixed programming languages
11. Does support all application sizes
12. **Partial support** for reusable artifacts
13. **Is somewhat slow and expensive to use**
CRITERIA FOR SOFTWARE METRICS SELECTION

Object-Oriented Metrics Meet 2 of 13 Criteria:

1. **Is not** standardized
2. **Is somewhat** unambiguous
3. **Does not** have a formal user group
4. **Does not** have adequate published data
5. Does have tools available for new projects
6. **Does not** have tools available for legacy projects
7. **Does not** have conversion rules for related metrics
8. **Cannot** deal with all deliverables
9. **Does not** support all kinds of software (**ONLY** supports OO)
10. **Does not** support all or mixed programming languages
11. Does support all application sizes
12. **Partial support** for reusable artifacts
13. **Is somewhat** slow and expensive to use
CRITERIA FOR SOFTWARE METRICS SELECTION

Agile Metrics (Story points, RTF) Meet 1 of 13 Criteria:

1. Is not standardized
2. Is somewhat unambiguous
3. Does not have a formal user group
4. Does not have adequate published data
5. Does have tools available for new projects
6. Does not have tools available for legacy projects
7. Does not have conversion rules for related metrics
8. Cannot deal with all deliverables
9. Does not support all kinds of software (only supports Agile)
10. Does not support all or mixed programming languages
11. Does not support all application sizes (< 5000 function points)
12. Partial support for reusable artifacts
13. Is somewhat slow and expensive to use
CRITERIA FOR SOFTWARE METRICS SELECTION

Cost per Defect metrics Meet 0 of 13 Criteria:

1. Is not standardized
2. Is highly unambiguous
3. Does not have a formal user group
4. Does not have adequate published data
5. Does not have tools available for new projects
6. Does not have tools available for legacy projects
7. Does not have conversion rules for related metrics
8. Cannot deal with all deliverables
9. Does not support all kinds of software (cannot handle zero-defects)
10. Does not support all or mixed programming languages
11. Does not support all application sizes or deliverables
12. Partial support for reusable artifacts
13. Is somewhat slow and expensive to use; penalizes quality
METRICS CONVERSION - A CRITICAL GAP

N-DIRECTION METRICS CONVERSION IS NEEDED BETWEEN:

• IFPUG VERSIONS 1, 2, 3, 4, 4.1, and 4.2 COUNTING RULES

• CURRENT IFPUG, MARK II, 3D, COSMIC, AND OTHER VARIATIONS

• FUNCTION POINTS AND PHYSICAL LOC METRICS

• FUNCTION POINTS AND LOGICAL STATEMENT METRICS

• PHYSICAL LOC AND LOGICAL STATEMENTS

• OBJECT-ORIENTED, AGILE, GQM METRICS AND OTHERS

ALL OTHER SCIENTIFIC AND ENGINEERING METRICS HAVE PUBLISHED CONVERSION RULES BETWEEN VARIANT METRICS.
COMPLEXITY - A TECHNICAL GAP

FUNCTION POINT COMPLEXITY IS DECOUPLED FROM:

• COMBINATORIAL COMPLEXITY
• COMPUTATIONAL COMPLEXITY
• CYCLOMATIC COMPLEXITY
• ESSENTIAL COMPLEXITY
• FLOW COMPLEXITY
• SYNTACTIC COMPLEXITY

OF THE 24 KINDS OF COMPLEXITY NOTED IN ENGINEERING AND SCIENTIFIC STUDIES, NONE ARE CURRENTLY INCLUDED IN FUNCTION POINT COUNTS.
STRENGTHS AND WEAKNESSES OF FUNCTION POINTS

The main strengths of IFPUG function point metrics are:

1. Function points stay constant regardless of programming languages used
2. Function points are a good choice for full life-cycle analysis
3. Function points are a good choice for benchmarks and economic studies
4. Function points are supported by many software estimating tools
5. Function points can be converted into logical code statements

The main weaknesses of IFPUG function point metrics are:

1. Accurate counting requires certified function point specialists
2. Function point counting can be time-consuming and expensive
3. There is a need for automated function point counts from requirements
4. There is a need for automated function point counts from code
5. IFPUG has no rules dealing with backfiring
6. IFPUG needs “micro function points” for small updates
STRENGTHS AND WEAKNESSES OF VARIANTS

The main strengths of function point variants are:

1. Variants stay constant regardless of programming languages used
2. Variants are a good choice for full life-cycle analysis

The main weaknesses of function point variants are:

1. Accurate counting requires certified function point specialists
2. Variant counting can be time-consuming and expensive
3. Variants have little published benchmark data
4. Variant counts are erratic for projects below 15 function points
5. Variants lack rules for backfiring
6. Variants lack conversion rules to IFPUG function points:
   - 3D function points, Mark II function points; Object points;
   - COSMIC function points; full function points; Story points; etc.

IT IS NOT IFPUG’S JOB TO CREATE CONVERSION RULES!
STRENGTHS AND WEAKNESSES OF BACKFIRING

The main strengths of backfiring are:

1. A fairly convenient method for aging legacy applications
2. One method for small applications < 15 function points
3. Costs less than < $0.25 per function point
4. Backfiring is supported by many software estimating tools
5. Function points can be converted into code statements and vice versa

The main weaknesses of backfiring are:

1. Accuracy of backfiring is never equal to counts by function point specialists
2. Backfiring varies between physical LOC and logical statements
3. Backfiring varies between individual programming styles
4. Backfiring varies between published lists for same languages
5. Backfiring is not endorsed by any function point association
6. Backfiring research is sparse and literature is scarce
7. Backfiring needs adjustment for dead code and complexity
## BACKFIRING: RATIOS OF LOGICAL SOURCE CODE STATEMENTS TO FUNCTION POINTS

<table>
<thead>
<tr>
<th>Language</th>
<th>Nominal Level</th>
<th>Source Statements Per Function Point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>Basic assembly</td>
<td>1.00</td>
<td>200</td>
</tr>
<tr>
<td>Macro assembly</td>
<td>1.50</td>
<td>130</td>
</tr>
<tr>
<td>C</td>
<td>2.50</td>
<td>60</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>3.00</td>
<td>75</td>
</tr>
<tr>
<td>COBOL</td>
<td>3.00</td>
<td>65</td>
</tr>
<tr>
<td>PASCAL</td>
<td>3.50</td>
<td>50</td>
</tr>
<tr>
<td>PL/I</td>
<td>4.00</td>
<td>65</td>
</tr>
<tr>
<td>Ada 83</td>
<td>4.50</td>
<td>60</td>
</tr>
<tr>
<td>C++</td>
<td>6.00</td>
<td>30</td>
</tr>
<tr>
<td>Ada 95</td>
<td>6.50</td>
<td>28</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>10.00</td>
<td>20</td>
</tr>
<tr>
<td>SMALLTALK</td>
<td>15.00</td>
<td>15</td>
</tr>
<tr>
<td>SQL</td>
<td>27.00</td>
<td>7</td>
</tr>
</tbody>
</table>

Note: Assumes Version 4 of IFPUG Counting Rules
STRENGTHS AND WEAKNESSES OF PATTERN MATCHES

The main strengths of pattern matching are:

1. Only convenient method for applications > 100,000 function points
2. Also useful for small applications < 15 function points
3. Fastest, cheapest known approach for sizing software
4. Costs less than $0.01 per function point
4. Pattern matching takes < 1 minute to size any known application
5. Only known method for sizing software with hidden requirements
6. Only known method for sizing COTS or Open Source by customers

The main weaknesses of pattern matching are:

1. Requires large volumes of historical data to get started
2. Requires a full taxonomy of software classes and types
3. Pattern matching is experimental in 2008
4. Pattern matching is proprietary in 2008
5. Pattern matching is not endorsed by any function point association
### EXAMPLES OF SIZING WITH PATTERN MATCHING

<table>
<thead>
<tr>
<th>Application</th>
<th>Size in IFPUG Function Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars Missile Defense</td>
<td>372,086</td>
</tr>
<tr>
<td>SAP</td>
<td>296,574</td>
</tr>
<tr>
<td>Microsoft Vista</td>
<td>159,159</td>
</tr>
<tr>
<td>Microsoft Office Professional 2007</td>
<td>97,165</td>
</tr>
<tr>
<td>FBI Carnivore</td>
<td>24,242</td>
</tr>
<tr>
<td>Skype</td>
<td>21,202</td>
</tr>
<tr>
<td>Apple I Phone</td>
<td>19,366</td>
</tr>
<tr>
<td>Google Search engine</td>
<td>18,640</td>
</tr>
<tr>
<td>Linux</td>
<td>17,505</td>
</tr>
</tbody>
</table>
STRENGTHS AND WEAKNESSES OF LOGICAL STATEMENTS

The main strengths of logical statements are:

1. Logical statements exclude dead code, blanks, and comments
2. Logical statements can be converted into function point metrics
3. Logical statements are used in a number of software estimating tools

The main weaknesses of logical statements are:

1. Logical statements can be difficult to count
2. Logical statements are not extensively automated
3. Logical statements are a poor choice for full life-cycle studies
4. Logical statements are ambiguous for some “visual” languages
5. Logical statements may be ambiguous for software reuse
6. Logical statements maybe erratic for conversion to the physical LOC metric
7. Logical statements are professional malpractice for software economics
STRENGTHS AND WEAKNESSES OF PHYSICAL LINES OF CODE (LOC)

The main strengths of physical lines of code (LOC) are:

1. The physical LOC metric is easy to count
2. The physical LOC metric has been extensively automated for counting
3. The physical LOC metric is used in some software estimating tools

The main weaknesses of physical lines of code are:

1. The physical LOC metric may include substantial “dead code”
2. The physical LOC metric may include blanks and comments
3. The physical LOC metric is ambiguous for mixed-language projects
4. The physical LOC metric is a poor choice for full life-cycle studies
5. The physical LOC metric does not work for some “visual” languages
6. The physical LOC metric is erratic for backfiring to function points
7. The physical LOC metric is professional malpractice for economics
STRENGTHS AND WEAKNESSES OF OO METRICS

The main strengths of OO metrics are:

1. The OO metrics are psychologically attractive within the OO community
2. The OO metrics can distinguish simple from complex OO projects

The main weaknesses of OO metrics are:

1. The OO metrics do not support studies outside of the OO paradigm
2. The OO metrics do not deal with full life-cycle issues
3. The OO metrics have no conversion rules to lines of code metrics
4. The OO metrics have no conversion rules to function point metrics
5. The OO metrics lack automation for counting
6. The OO metrics are not supported by software estimating tools
7. The OO metrics have no benchmarks or support organization
STRENGTHS AND WEAKNESSES OF AGILE METRICS

The main strengths of Agile metrics are:

1. Agile metrics are useful within the Agile community
2. Agile metrics can create Agile benchmarks but have no yet done so
3. Agile metrics include RTF, Story Points, and LOC metrics

The main weaknesses of Agile metrics are:

1. Agile metrics do not support studies outside of the Agile paradigm
2. Agile metrics do not deal with full life-cycle issues
3. Agile metrics have no conversion rules to lines of code metrics
4. Agile metrics have no conversion rules to function point metrics
5. Agile metrics lack automation for counting
6. Agile metrics are not supported by software estimating tools
7. Agile metrics have no benchmarks available
STRENGTHS AND WEAKNESSES OF GQM METRICS

The main strengths of Goal-Question-Metrics (GQM) are:

1. GQM metrics are useful for a wide range of problems
2. GQM metrics are useful for executive reporting
3. GQM metrics support innovative solutions

The main weaknesses of GQM metrics are:

1. GQM metrics are unique, and difficult to use for benchmarks
2. GQM metrics collect random kinds of data
3. GQM metrics have no conversion rules to lines of code metrics
4. GQM metrics have no conversion rules to function point metrics
5. GQM metrics lack automation for counting
6. GQM metrics are not supported by software estimating tools
7. GQM metrics have no user group or support organization
STRENGTHS AND WEAKNESSES OF COST PER DEFECT

The main strengths of cost per defect metrics are:

1. Cost per defect is easy to count

The main weaknesses of cost per defect metrics are:

1. Cost per defect metrics penalize quality!
2. Cost per defect metrics ignore fixed costs
3. Cost per defect metrics violate standard economic principles
4. Cost per defect has no conversion rules to function point metrics
5. Cost per defect lacks automation for counting
6. Cost per defect has no user group or support organization
7. Cost per defect is professional malpractice for quality analysis
FUNCTION POINTS ALONE ARE NOT ENOUGH!!

To become a true engineering discipline, many metrics and measurement approaches are needed:

- Accurate Effort, Cost, and Schedule Data
- Accurate Defect, Quality, and User-Satisfaction Data
- Accurate Usage data
- Source code volumes for all languages
- Types and volumes of paper documents
- Volume of data and information used by software
- Consistent and reliable complexity information
EVEN QUANTITATIVE DATA IS NOT ENOUGH!!

Reliable qualitative information is also needed. For every project, “soft” data should be collected:

- Complete software process assessments
- Data on SEI capability maturity levels
- Extent of creeping requirements over entire life
- Methods, Tools, and Approaches Used
- Geographic factors
- Organizational Factors
- Experience levels of development team
- Project risks and value factors
FUNCTION POINTS AND FUTURE METRICS

• Software is only one technology used in modern business, in manufacturing, in government, and in military organizations.

• Can function points be expanded to other business activities?

• Can we build a suite of metrics similar to function points, for other aspects of modern business?
FUNCTION POINTS AND OTHER METRICS

POTENTIAL BUSINESS METRICS

- Function points - Measures software size
- Data points - Measures data base size
- Service points - Measures support size
- Engineering points - Measures hardware size
- Value points - Measures tangible value
# Proposed Suite of Metrics

<table>
<thead>
<tr>
<th>Function Points</th>
<th>Data Points</th>
<th>Value Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Entities</td>
<td>Time to market</td>
</tr>
<tr>
<td>Outputs</td>
<td>Sets</td>
<td>Cost reduction</td>
</tr>
<tr>
<td>Inquiries</td>
<td>Attributes</td>
<td>Revenue increase</td>
</tr>
<tr>
<td>Logical files</td>
<td>Interfaces</td>
<td>Market share</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Constraints</td>
<td>Morale</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Health/Safety</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Risk reduction</td>
</tr>
<tr>
<td>Service Points</td>
<td>Engineering Points</td>
<td>National security</td>
</tr>
<tr>
<td>Customers</td>
<td>Algorithms</td>
<td>Mandates/statutes</td>
</tr>
<tr>
<td>Inputs</td>
<td>Inventions</td>
<td>Customer satisfaction</td>
</tr>
<tr>
<td>Outputs</td>
<td>References</td>
<td>Synergy</td>
</tr>
<tr>
<td>Inquiries</td>
<td>Feedback</td>
<td>Competitiveness</td>
</tr>
<tr>
<td>Logical files</td>
<td>Constraints</td>
<td>Performance increases</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Inputs</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td>Outputs</td>
<td></td>
</tr>
<tr>
<td>References</td>
<td>Interfaces</td>
<td></td>
</tr>
</tbody>
</table>
# POTENTIAL INTEGRATED COST ANALYSIS

<table>
<thead>
<tr>
<th>Unit of Measure</th>
<th>Size</th>
<th>Unit $</th>
<th>Total Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function points</td>
<td>1,000</td>
<td>$500</td>
<td>$500,000</td>
</tr>
<tr>
<td>Data points</td>
<td>2,000</td>
<td>$300</td>
<td>$600,000</td>
</tr>
<tr>
<td>Service points</td>
<td>1,500</td>
<td>$250</td>
<td>$375,000</td>
</tr>
<tr>
<td>Engineering points</td>
<td>1,500</td>
<td>$700</td>
<td>$1,050,000</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>6,000</td>
<td>$420</td>
<td>$2,525,000</td>
</tr>
<tr>
<td>Value points</td>
<td>10,000</td>
<td>$600</td>
<td>$6,000,000</td>
</tr>
<tr>
<td><strong>ROI</strong></td>
<td></td>
<td></td>
<td><strong>$2.37 per $1.00 spent</strong></td>
</tr>
</tbody>
</table>
SUMMARY AND CONCLUSIONS

• Function Points have aided software management but > 25 variations are excessive.

• Scientific and software engineering complexity should be included in function point analysis.

• The speed and cost of function point analysis need to be improved.

• Function points should aim for 100% usage between 1 function point and 1,000,000 function points!

• Function points should aim for usage analysis and portfolio analysis to become a major metric for global economic studies!
SUMMARY AND CONCLUSIONS

- Software is a major business tool for all industries, all levels of government, and all military organizations.
- Function points are the only metric that can measure quality, productivity, costs, value, and economics without distortion.
- Function points can become a key tool for large-scale economic analysis in every industry and country in the world!
- Function points can also become a key factor in software risk analysis and risk avoidance. No other metric can do this.
- Function points can improve the professional status of software and turn “software engineering” into a true engineering discipline instead of a craft as it is today.
REFERENCES TO FUNCTION POINT MEASUREMENTS

• IFPUG; *IT Measurement*; Addison Wesley, 2002
• Jones, Capers; *Assessments, Benchmarks, and Best Practices*; Addison Wesley, 2000.
• Putnam, Larry; *Measures for Excellence*; Yourdon Press, Prentice Hall, 1992
• Web sites:
  • ITMPI.ORG  COSMIC.ORG  Relativity.COM  CSPIN.ORG
  • IFPUG.ORG  ISO.ORG  ISBSG.ORG  FISMA.FI
  • SPR.COM  ISMA.ORG  IFPUG.gr.jp  GUFPI.ORG