Preliminary Causal Analysis Results with Software Cost Estimation Data

Anandi Hira, Bob Stoddard, Mike Konrad, Barry Boehm
Parametric Cost Models

**COCOMO® II**

\[
\text{Effort} = 2.94 \times Size^E \times \prod_{i=1}^{17} EM_i
\]

- Input: size, product and personnel attributes
- Effort in Person-Months (PM)
- Size in KSLOC (1000 SLOC)

- Domain Experts
- Data calibration
- No causal analysis
Causal Inference

- Causal Learning/Inference
  - Causal Discovery
    - Algorithms and Domain Knowledge on Data
  - Causal Estimation
    - Algorithms to quantify causal influence
## Past Causal-Type Analyses

<table>
<thead>
<tr>
<th>Dr. Boehm COCOMO® 81</th>
<th>Cuoto et al</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-depth behavioral analyses for effort factors</td>
<td>Granger’s causality test for software defect predictability</td>
</tr>
<tr>
<td>Evidence-Based SE</td>
<td>Doesn’t get to heart of causality</td>
</tr>
<tr>
<td>Experiments</td>
<td></td>
</tr>
<tr>
<td>Cause precede effect</td>
<td></td>
</tr>
<tr>
<td>Cause covaries with effect</td>
<td></td>
</tr>
<tr>
<td>Alternative explanations are implausible</td>
<td></td>
</tr>
<tr>
<td>Hu et al</td>
<td></td>
</tr>
<tr>
<td>Bayesian networks with causality constraints for software risk factors</td>
<td></td>
</tr>
</tbody>
</table>
PC Search

- Named after Peter Spirtes and Clark Glymour
- First scalable discovery algorithm

- Change in $X_1$ causes change in $X_2$
- Insufficient data to select orientation
- May be common confounder of both variables, missing from dataset
Dataset: Unified Code Count (UCC)

Project Description
- Maintained at USC
- Code metrics tool (logical SLOC, cyclomatic complexity)
- Implemented in C++
- 45 to 1425 logical SLOC
- 2010 to 2014
- Modularized architecture
- 4-month time-boxed increments

Project Types
- Add Functions
  - New language parsers
  - New features, such as GUI
- Modify Functions
  - Cyclomatic complexity support (modify existing language parsers with mathematical operation and algorithms)
Dataset Attributes

1. Equivalent SLOC
2. IFPUG Function Points
3. IFPUG Software Non-functional Assessment Process
4. COSMIC Function Points
5. Total Effort
6. Applications Experience
7. Platform Experience
8. Use of Software Tools
9. Personnel Continuity
10. Documentation Match to Needs
11. Analyst Capability
12. Programmer Capability
13. Product Complexity
All Data Points
IFPUG FPs

Normalized Effort (hours)

Enhancement Function Points

Modified Functions
Add Functions

ACAP
PCAP
TotalEffort
FPs
CPLX
DOCU
Add Functions
Normalized Effort (hrs) vs Equivalent SLOC
IFPUG FPs

Enhancement Function Points vs. Normalized Effort (hours)
IFPUG SNAP

Enhancement SNAP Points vs. Normalized Effort (hours)
COSMIC FPs

Normalized Effort (hours)

COSMIC Function Points

ACAP

TotalEffort

PCAP

CFPs

CPLX

DOCU
Modify Functions
IFPUG FPs

Enhancement Function Points

Normalized Effort (hours)
COSMIC FPs

Normalized Effort (hours)

COSMIC Function Points
Conclusion
# Conclusions

## General Conclusions
- All Data Points
  - SNAP -> Total Effort
  - CFPs -> Total Effort
  - PCAP – Total Effort
  - ACAP – PCAP
- Add Functions
  - PCAP – Total Effort
- Modify Functions
  - ESLOC – Total Effort
  - SNAP – Total Effort
  - ACAP – PCAP

## “Interesting” Results
- All Data Points
  - CFPs -> DOCU
- Modify Functions
  - CFPs -> PCAP
  - ACAP -> PCAP