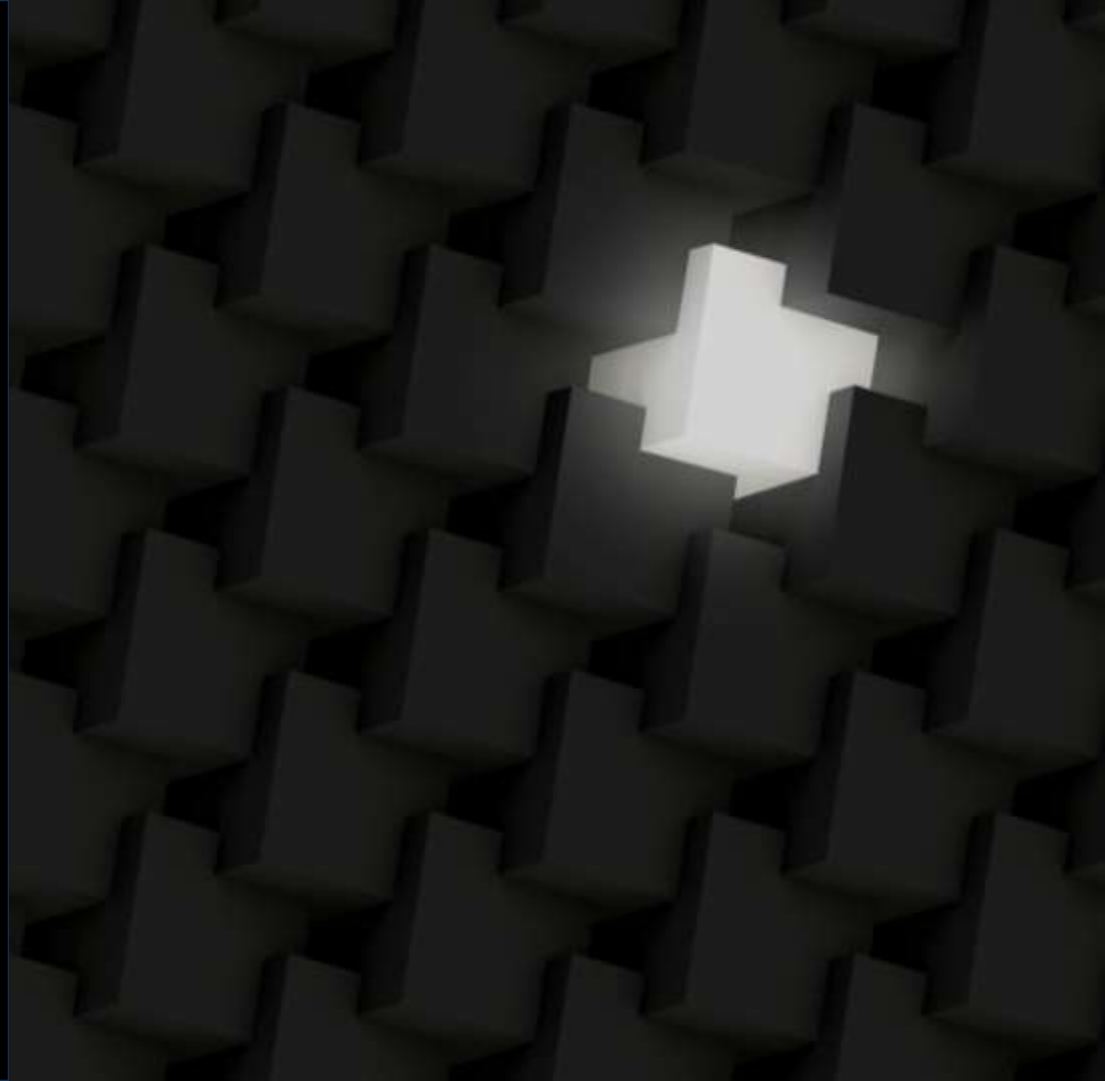
Carnegie Mellon University
Software Engineering Institute

RESEARCH REVIEW 2020

Causal Models for Software Cost Prediction & Control (SCOPE)

Mike Konrad, Robert Stoddard, Bill Nichols, Dave Zubrow, Michele Falce, Rhonda Brown, and Bryar Wassum



Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM20-0844

Why Causal Learning?

- Estimating and controlling program costs benefits from **causal** knowledge of program dynamics.
- Regression does not **distinguish** between correlation and causation.
- Causal knowledge is **actionable** knowledge.
- Causal discovery is becoming **practical** and is supported with **innovative** tools and algorithms.

Establishing causation with observational data is a vital need and a key technical challenge.

Detecting causation is becoming more feasible and practical.

Causal Graphs in a Nutshell

DAGs – Directed Acyclic Graphs

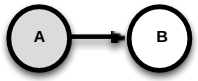
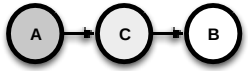
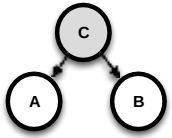
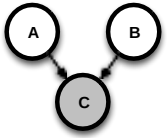
- **Nodes** are factors of interest (observed or latent).
- **Edges** are where there is direct causation.
  - If you hold other variables constant, wiggling a node at the tail of an oriented edge affects the node at its head, but not vice versa.

A DAG consists of nodes connected by directed edges. DAGs do not have cycles.

The result of a causal discovery algorithm may be a DAG or a pattern (i.e., a DAG where some edges are left undirected).

Building Blocks for DAGs

Combining the atomic units results in a causal graph (DAG). The following table describes how different causal relationships appear in a DAG.

	**Direct causation** between A and B. We say this: A is a parent of B; B is a child of A.
	**Causal chain** – A indirectly causes B through C. We say this: A is an ancestor of B; B is a descendant of A.
	**Common cause (fork)** – C directly causes both A and B, thus inducing an association between A and B.
	**Common effect (collider)** – A and B share a common effect, C. Conditioning on (i.e., adjusting for) C can induce an association between A and B.

Common Cause on Treatment and Effect

"Simpson's Paradox" can happen when there is a mediated common cause.

- A trend that appears in different groups of data may disappear or reverse when these groups are combined.

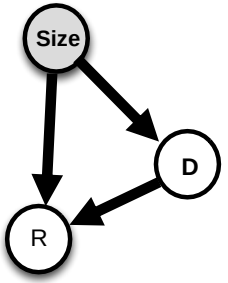
For example, the causal structure and graph on the right illustrate this paradox.

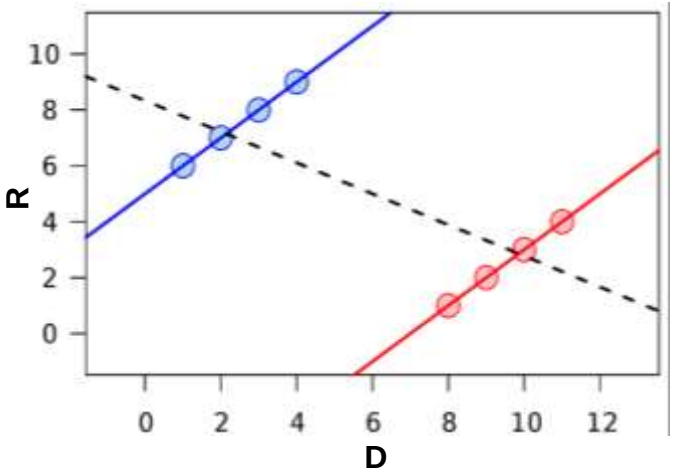
- For the whole population, dosage (D) and pain relief (R) correlate negatively.
- But, if we split into two groups according to the size of kidney stones, within each group, D and R correlate positively.
- The D and R association reverses.

Size – kidney stone size (small/large)

D – dosage (0..12)

R – days free from pain (0..10)





Causal Discovery (Structure Search) and Estimation in a Nutshell

The causal structure captured in DAGs (and patterns) is critical to correctly interpreting and analyzing observational data.

The Data

- Apply prescriptive rules to infer causality implications.
- Test the implications statistically.
- Grow (or shrink) the graph.

Background Knowledge

(What we know from our domain expertise helps resolve ambiguities.)

- Test causal models for consistency with data and knowledge.
- Test models to discover the data's underlying causal structure.

Once we have a causal structure, we can proceed to Estimation, assigning values to underlying model parameters, and enabling quantitative inferences (predictions) to be made.

SCOPE LSI Project End Products

In this presentation, we showcase selected products of the SCOPE LSI project.

1. **Methodology** for Causal Inference on **Small Data Sets**
2. **Direct Causes** of **Software Cost and Schedule**: A summary of our Causal Discovery investigation of COCOMO® II software and COSYSMO 3.0 systems engineering cost estimation
3. **Consensus Graph for U.S. Army Sustainment**: A summary of our investigation of a U.S. Army software sustainment release data set
   - Originally split by super domain and ACAT level to reduce conditions for Simpson's Paradox, but then search results were stitched together into an integrated model
4. **Other Research Artifacts and Transition**

Methodology for Causal Inference on Small Samples

This approach to causal inference is **principled** (i.e., no cherry picking) and **robust** (to outliers).

- for **small samples**: when the number of cases is < 5-10 times the number of variables

1. Inject **null variables** by appending an independently randomized copy of each original variable.
2. Search (FGES or PC with default settings except for regularization) with **bootstrap** to determine each edge's **Probability of No Edge (PNE)** across the search.
3. Set a **threshold** (e.g., 10th percentile) among the edges involving a null variable.
   - (Of edges involving a null variable, 90% have a PNE exceeding that threshold.)
4. **Trim** the remaining edges (involving only original variables) when their PNE > threshold.
   - Surviving edges are a mix of true-positive edges and a few false-positive edges.
   - The PNE of a surviving edge ~ = fraction of cases (data set rows) that have that direct causal relationship.

When proceeding to **model estimation**, repeat these four steps, as needed, for the 15th, 20th, or 25th percentile trims until a decent model fit is achieved.

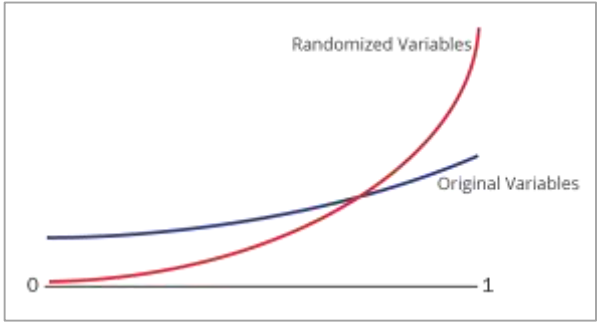
Investigating the Effects of Spurious Correlations

Experimentation revealed that randomly generated variables often form edges.

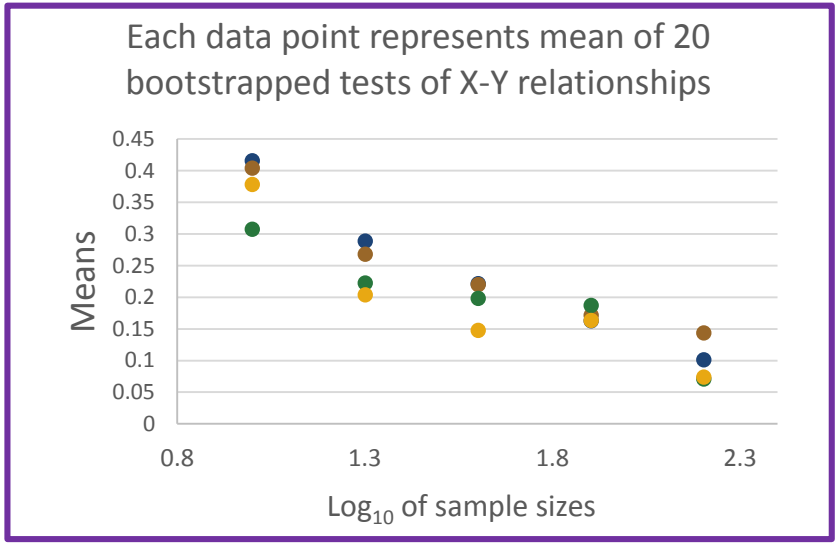
However, when we search a project data set, some edges may be spurious.

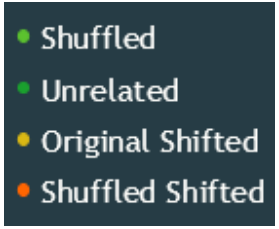
Is the frequency of spurious correlations more likely with smaller data sets?

How significant is the effect using different sample sizes? Here's what we learned:



Probability of No Edge





Key to X-Y Relationships

Thus, when performing causal search on small data sets, we apply our methodology to reduce false positives.

Direct Causes of Software Cost and Schedule

Intervening on these elements of a project may improve outcomes.

COCOMO® II - Effort

- Size (SLOC)
- Team Cohesion
- Platform Volatility
- Reliability
- Storage Constraints
- Time Constraints
- Product Complexity
- Process Maturity
- Risk and Architecture Resolution

COCOMO® II - Schedule

- Size (SLOC)
- Platform Experience
- Schedule Constraint
- Effort

COSYSMO 3.0 - Effort

- Size
- Level of Service Requirements

See the definitions of these factors in the COCOMO® II Model definition manual and COSYSMO 3.0 paper.

From among 40+ factors these sources describe, these are the ones found to be direct drivers of cost and schedule.

Using Tetrad to Generate Mini Cost-Estimation Models

We used the following six steps to generate mini-models that produce plausible cost estimates that are guided by the self-imposed structure of the existing estimating models:

1. Force cost drivers to have no direct causal relationships with one another.
2. Instead of including each scale factor as a variable (as we do in effort multipliers), replace them with a new variable—**scale factor** times **LogSize**.
3. Apply causal discovery to obtain a plausible causal graph.
4. Use Tetrad model estimation to obtain **parent-child edge coefficients**.
5. Lift the equations from the resulting graph to form the mini-model, re-applying estimation to properly determine the intercept.
6. Evaluate the fit of the resulting model and its predictability.

COSYSMO 3.0 Estimation

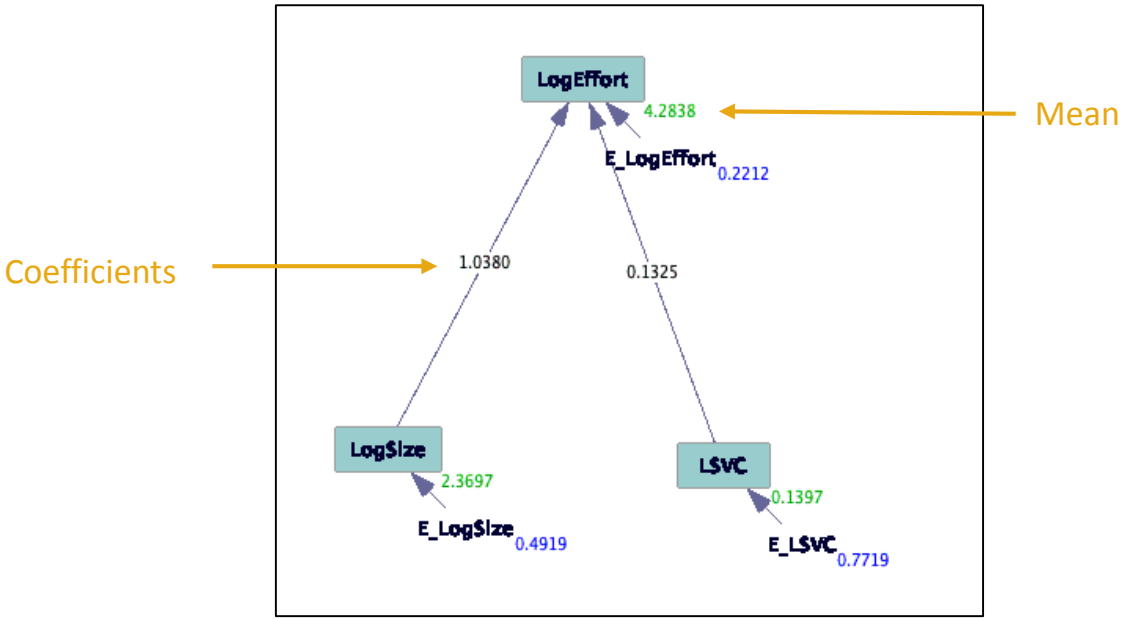
Overall Model Fit Statistics

Chi-Square Test(s) of Model Fit
P-Value = 2.3667E-5
(want > 0.01, or at the very least > 0);
Chi-Square/DF = 17.8688 (want < 5)

RMSEA (Root Mean Square Error of Approximation)
RMSEA = 0.5018 (want < 0.08)

CFI (Comparative Fit Index)
0.9967 (want > 0.95)

The conclusion is that the model fit is poor to fair.



COCOMO® II Estimation

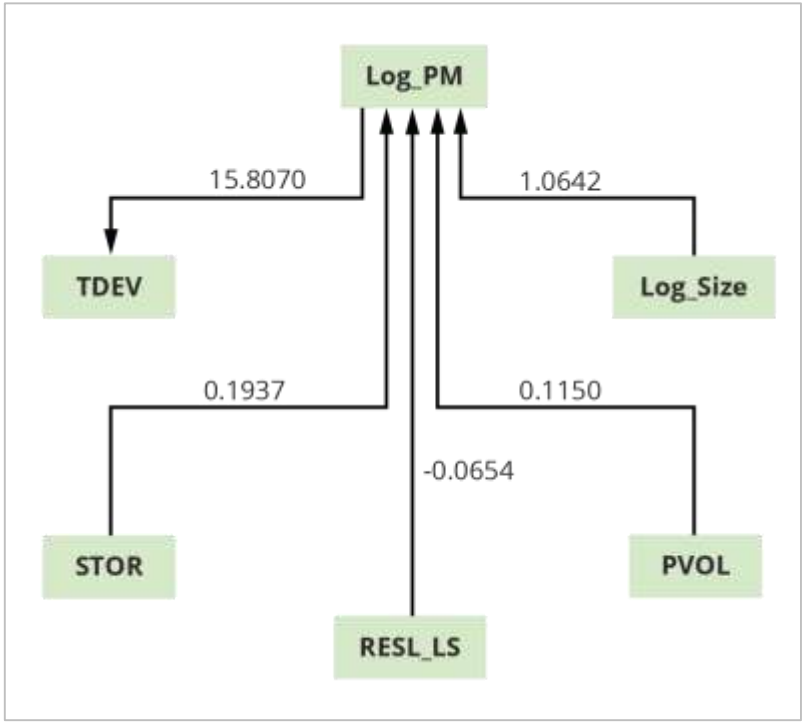
Overall Model Fit Statistics

Chi-Square Test(s) of Model Fit
P-Value = 8.9571E-5
(want > 0.01, or at the very least > 0);
Chi-Square/DF = 3.58 (want < 5)

RMSEA (Root Mean Square Error of Approximation)
RMSEA = 0.1271 (want < 0.08)

CFI (Comparative Fit Index)
0.9993 (want > 0.95)

The conclusion is that the model fit is fair to good.



Prediction Accuracy: Mini-Models vs. Estimating Models

COSYSMO 3.0 – Effort

	Mini-Model	Original
Max MRE	285.4%	234.8%
MMRE	45.9%	57.3%
PRED(25)	41.2%	23.5%
PRED(30)	48.5%	23.5%

COCOMO® II – Effort

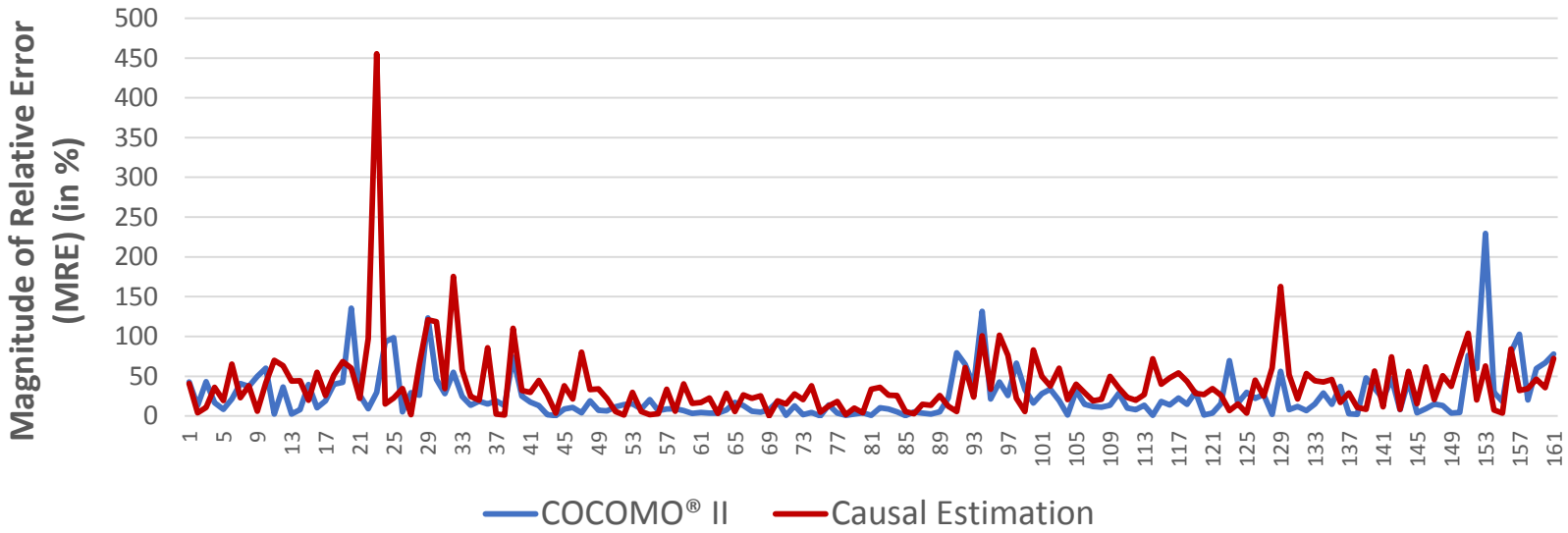
	Mini-Model	Original
Max MRE	455.4%	229.41%
MMRE	38.64%	25.67%
PRED(25)	44.72%	67.08%
PRED(30)	52.8%	74.53%

COCOMO® II – Schedule

	Mini-Model*	Original
Max MRE	628.6%	130.95%
MMRE	42.28%	50.88%
PRED(25)	45.34%	9.94%
PRED(30)	52.8%	12.42%

* Analysis done with TDEV; but realized Log(TDEV) might have been better.

COCOMO® II vs. Mini-Model Effort MRE Comparison



Using more factors for a cost estimate (as with the full model) tends to reduce the frequency of way-off predictions. (Of course, on any given project, either model might be more accurate.) The advantage of the mini-model is that it identifies which factors, among many, are more likely to drive cost and schedule.

Consensus Graph for U.S. Army Software Sustainment –1

A U.S. Army AFC-CCDC data set was segmented into (Superdomain, ACAT-level) pairs, resulting in five sets with barely sufficient data to search and estimate.

- Splitting was needed to address high fan-out for several common causes, which would lead to structures typical of Simpson's Paradox.

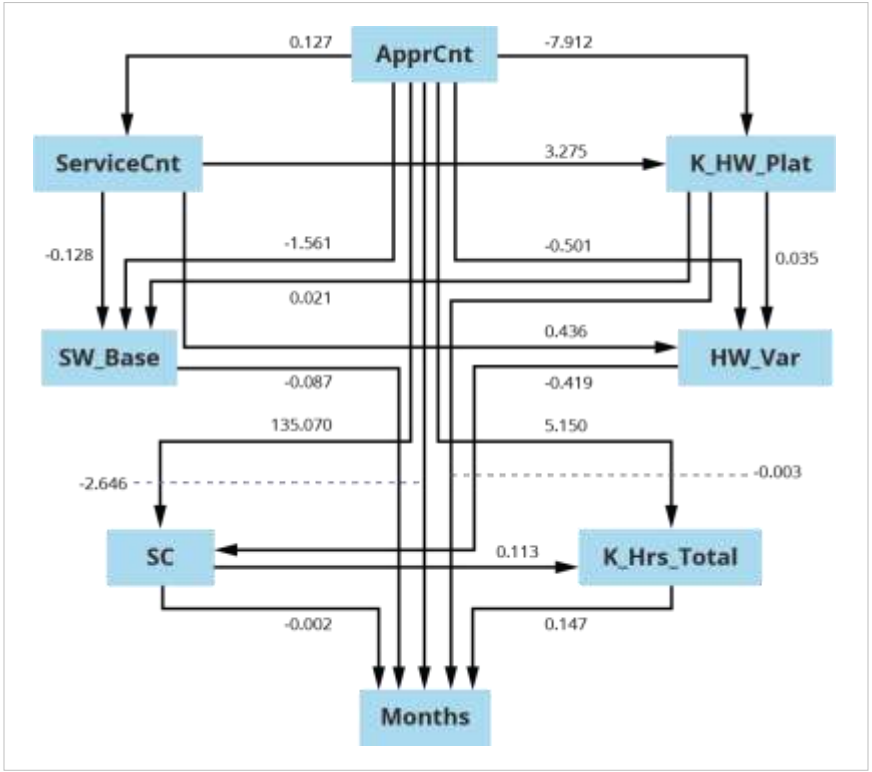
A consensus graph was built from the resulting five fitted models.

For consensus estimation, the data from individual searches was pooled with data that was previously excluded because of missing values. The resulting 337 releases were then used to estimate the consensus graph using Mplus with Bootstrap in estimation.

The next slide shows the resulting structural equation model, coefficients, and model fit. There was no cherry picking or re-do's—this model is a direct out-of-the-box estimation, achieving good model fit on the first try.

*This slide and the next two are from analyses of a data set kindly provided by Cheryl Jones, U.S. Army AFC-CCDC, and DASA-CE of software releases from the past seven years. The program identification was removed. We thank her and her team for their insight into the data set and meaning of certain variables as the work progressed.

Consensus Graph for U.S. Army Software Sustainment –2



Model Fit Statistics

Chi-Square Test(s) of Model Fit

P-Value 0.0279 is not > 0.05, but with 337 observations, not too worrisome

Chi-Square/DF = 20.15/10 < 5

RMSEA (Root Mean Square Error of Approximation):

0.055 < 0.08

CFI (Comparative Fit Index)

0.978 > 0.95

SRMR (Standardized Root Mean Square Residual):

0.033 < 0.08

Other Research Artifacts and Transition

In the rest of this presentation, we identify other research artifacts and transition vehicles for the SCOPE LSI project:

- other data sets analyzed
- lessons learned and caveats
- reflections and impacts
- SCOPE transition products

Other Data Sets Analyzed

Personal Software Process (PSP). We examined (1) inter-programmer vs. intra-programmer variation in productivity and (2) evidence for a quality-conscientiousness trait.

Open Source Software Cyber Vulnerabilities. We examined (1) Project Chromium for drivers of cyber vulnerabilities and (2) OpenSSL for social smell-based drivers of excessive commits and churn.

SSDR. We identified drivers of software effort and cost growth.

Sources of Complexity vs. Program Success (Sarah Sheard's Ph.D. data set). We identified drivers of program success.

Process Improvement vs. Program Outcomes. We analyzed an USAF data set to identify which of about 40 process and team factors were direct drivers of quality, cost, and schedule.

Lessons Learned and Caveats –1

Lessons Learned. By performing causal inference (causal discovery and model estimation), we did the following:

- We identified factors important for cost estimation and attained insight into cost-estimating relationships, and how they might be improved.
- We obtained an integrated and estimated causal model that indicates how release effort and duration are impacted by changes to appropriations, the number of hardware platforms, etc.
- More broadly, by analyzing a dozen data sets, we identified from among hundreds of correlations, which factors are actually direct causes of DoD program outcomes (cost, schedule, and security/quality). Intervening on these causes is more likely to achieve intended outcomes.

Lessons Learned and Caveats –2

Caveats. In these analyses, the following algorithmic assumptions were made for imputing missing data, discovering causal structure, and estimating parameters:

- The data set has no critical missing (unmeasured) confounders.
- Software sustainment releases are independent of one another. (In reality, releases from the same program are more alike.)
- Data is randomly drawn from a multi-variate normal distribution.

Reflections and Impacts

Three years ago we hypothesized, “Among the dozens of factors that predict cost, few will prove causal.”

- Our research confirms this hypothesis.
- Intervening on other factors might not only be ineffective, but it might even be counter-productive.

Having a method for applying causal discovery to DoD program data sets, which typically are small, is critical to making progress with causal discovery.

- Our method eliminates cherry picking and provides robustness to data outliers.

Collaborating with data owners is the key to successful research. We thank the following people for their insights:

- Cheryl Jones and her team from the U.S. Army AFC-CCDC, and DASA-CE
- Dr. Anandi Hira and Dr. James Alstad from Barry Boehm’s Center for Systems and Software Engineering at USC

SCOPE Transition Products

In addition to a number of causal graphs—estimated and not estimated—SCOPE can be used by project management with minimal training. The SCOPE project developed the following transition products:

- **SCOPE Project Web Page.** This web page describes the other research we performed in this project with corresponding data sets.
- **Specification of a Causal Structure for Program Risks/Change Drivers.** SEI staff of the Agile Collaboration Group (ACG) defined a causal structure for program risks, providing a basis for designing interventions for program risk mitigation.
- **Report on Change Drivers Identifying Common Cascading Consequences.** SEI staff participated in an internal survey to identify (1) events that led to cost growth and (2) the most common cascading consequences responsible for non-linear cost growth.
- **QUELCE Workshop.** Results of these and other causal relationships identified in over a dozen program data sets were incorporated into the Quantifying Early Lifecycle Costs for Cost Estimation (QUELCE) workshop for reliable cost estimation throughout the life of a program, starting early in the life of a program.

Project Artifacts List

1. Accepted publications (e.g., ESEC_FSE_20 Journal First; ICEAA 2020)
2. Accepted presentations (e.g., Joint Software & IT-Cost Forum [with USC] and IEEE Reliability, Availability, Maintainability Symposium [RAMS] 2021)
3. A summer intern presentation on pseudo-random number generators (PRNGs), which enable simulation, numeric integration, etc. used by our tool chain
4. An integrated causal model for U.S. Army sustainment
5. Mini-Models for Cost Estimation for Systems Engineering (COSYSMO 3.0) and Software Engineering (COCOMO® II)
6. A method for causal discovery and model estimation from small data sets
7. A causal model for program risks
8. Quantifying Early Lifecycle Costs for Cost Estimation (QUELCE) artifacts enriched with knowledge gained from the SCOPE project
9. The PAIRS processing tool, used by DoD cost estimators to compile SRDR reports
10. Code and tools for automating the analysis pipeline (e.g., Python, R, VBA, Mplus)
11. New causal discovery algorithms developed by CMU (Multi-FASK and implementation of CD-NOD for Tetrad)

Recent Project Bibliography

W. Nichols, “The Myth of Individual Programmer Productivity” accepted for publication at *Foundations of Software Engineering (FSE), 2020*

This work originated from a SCOPE study of performance factors in PSP. The results were largely negative (no clear productivity factors). Following up found previously unstudied results in personal performance variation. That is, the negative result from causal learning led to discovering that individual performance variation was comparable to performance ranges between programmers.

B. Stoddard, "A New Science for Reliability" accepted for the *2021 IEEE Reliability, Availability, Maintainability Symposium (RAMS)*

A. Hira, J. Alstad, and M. Konrad, “Investigating Causal Effects of Software and Systems Engineering Effort” presented at the *Joint Software and IT Cost Forum*, September 2020

This work is the culmination of a nearly two-year collaboration with Barry Boehm’s Center for Systems and Software Engineering at USC. USC has proprietary data sets that it cannot share with the SEI. However, since the SEI has causal discovery and model estimation expertise, the two teams worked together to derive causal structure models and estimate them.