Do Software Architecture Patterns Reduce Security Vulnerabilities? Insight from Causal Learning

Robert Stoddard

Bill Nichols

Mike Konrad

Rick Kazman

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM19-0918

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Agenda

Introduction

Architecture Security Causal Research

Army Cost and Security Causal Research

Call to Action

Motivation for Causal Learning

Controlling items requires knowing which "independent factors" actually *cause* item outcomes, so that we may change items in a predictable manner.

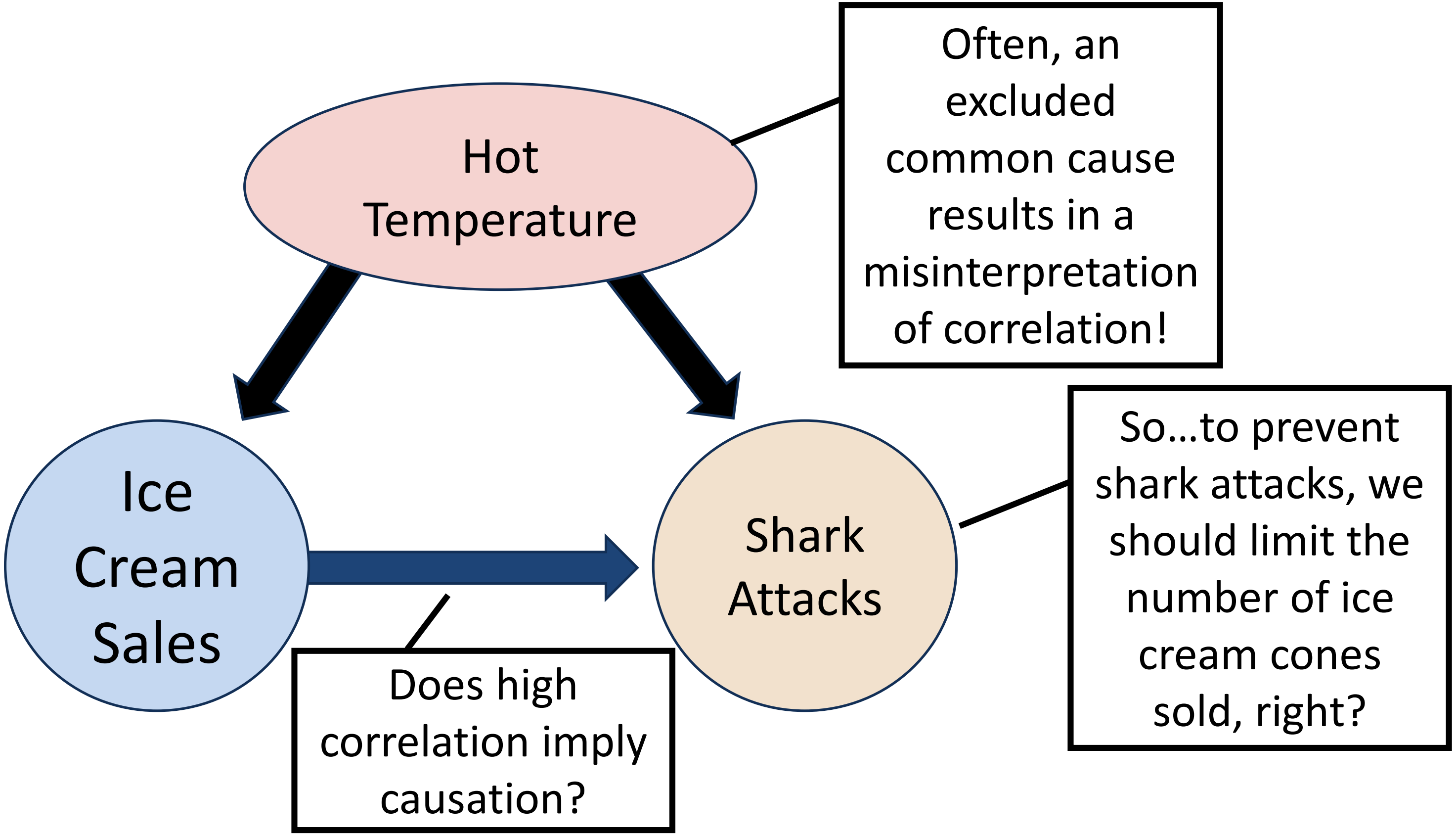
Just as correlation may be fooled by spurious association, so can regression

We must move beyond correlation to causation, if we want to make use of cause and effect relationships

We can now evaluate causation without expensive and difficult experiments

Establishing causation with observational data remains a vital need and a key technical challenge, but is becoming more feasible and practical.

Primary Reason for Spurious Association



Different Uses for Correlation versus Causation

Correlation	Causation
Classifying & identifying	Influencing & acting
Informational value of different evidence	Using evidence to guide policy or actions
Prediction & reasoning given observations	Prediction & reasoning given interventions
Probable explanations for some event or issue	Ways to produce or prevent an event or problem

Architecture Security Causal Research

Prior research:

Mo, R., Cai, Y., Kazman, R., Xiao, L., & Feng, Q. (2019). Architecture Anti-patterns: Automatically Detectable Violations of Design Principles. *IEEE Transactions on Software Engineering*.

Chromium Factors (Attributes measured at file level)

File Age: *age in days within project history*

Latest LOC: *lines of code in latest version*

Clique: *binary factor of whether or not file participates structurally in a connected graph, aka clique*

Crossing: *total count of the file fan-in and fan-out*

Modularity Violation: *number of times a file participates in a modularity violation group*

Package Cycle: number of times a file participates in a package cycle

Unhealthy Inheritance: *number of times a file participates in an unhealthy inheritance relationship*

Unstable Interface: *number of times a file participates in a change representing an unstable interface*

Bug Churn: *total count of churn associated with one issue id that a file is associated with*

CoChange: *number of times a file has been co-committed with other files*

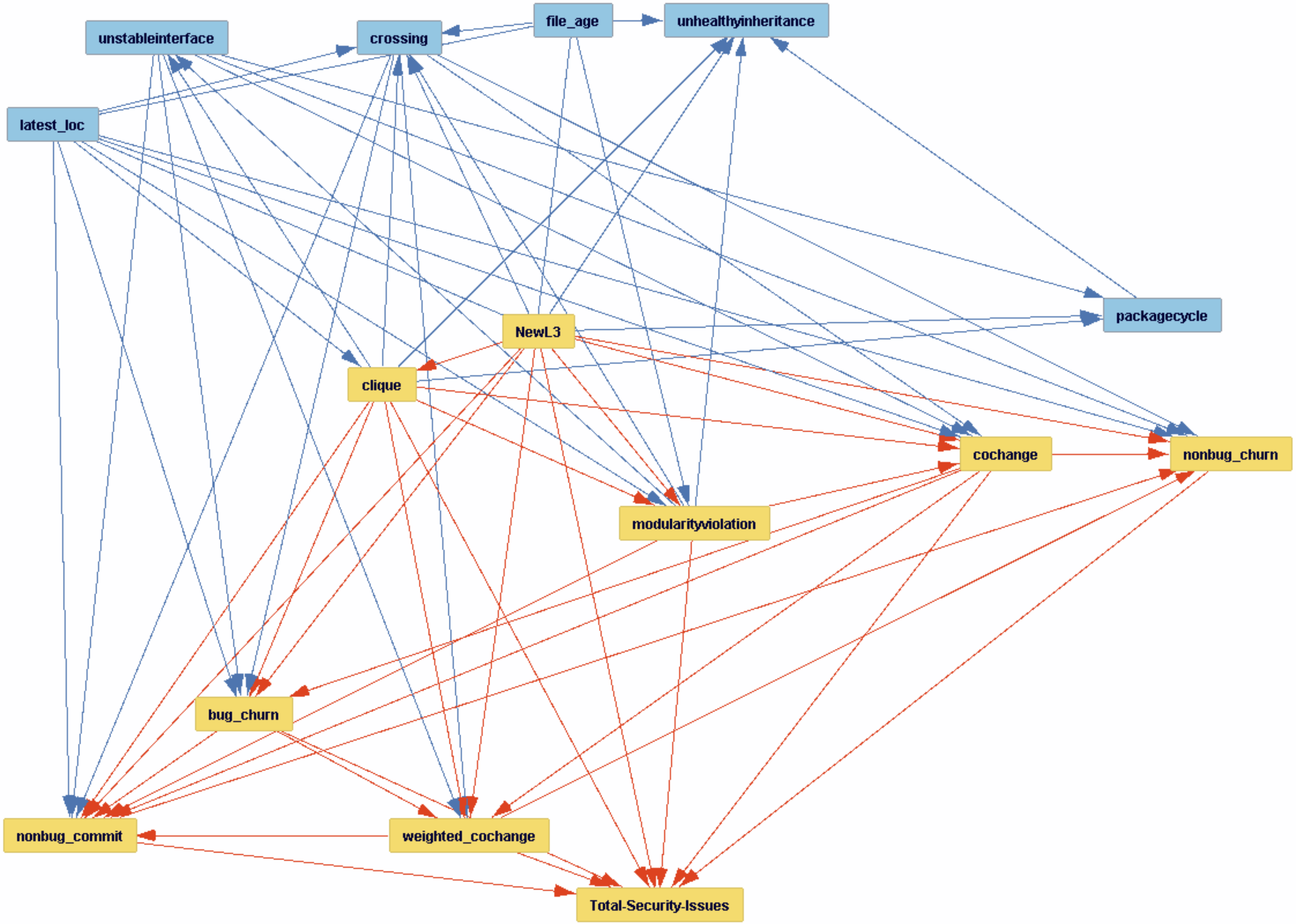
NonBug Churn: *total count of churn for a file that is not affiliated with an issueid*

NonBug Commit: *total number of file commits that are not associated with any issueids*

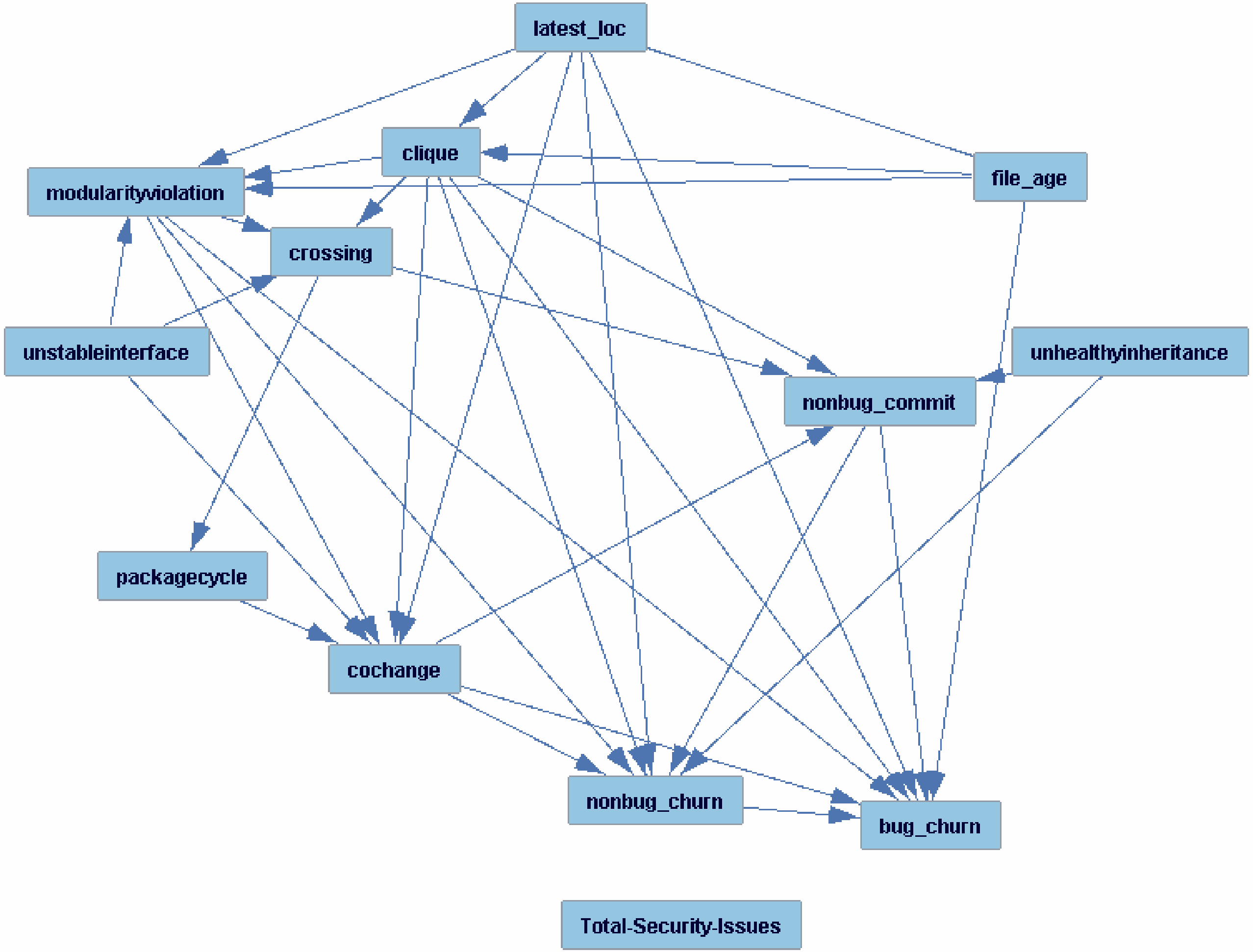
Weighted CoChange: *specific weighting approach of (1/number of files fixed) per commit*

Total Security Issues: *Total count of cve and other security fixes involving a file*

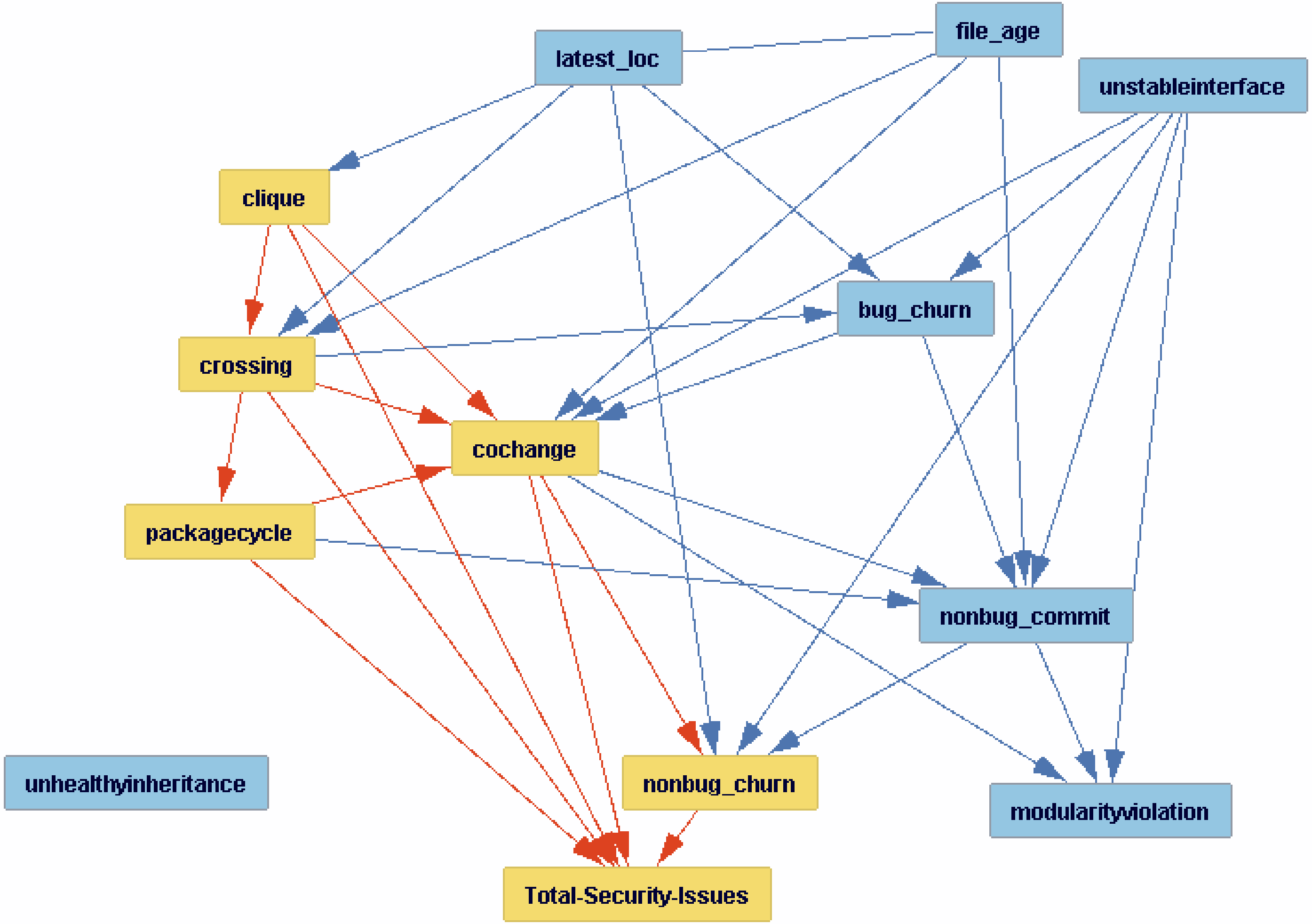
Chromium Entirety (FGES)



Chromeos Partition (FGES)



Extension Partition (FGES)



Architecture Pattern Causes of Total Security Issues

Legend:
Green = *Direct Causal Evidence*
Yellow = *Indirect Causal Evidence*
Red = No Causal Evidence
Grey = *Not Applicable*

Outcome: File Affiliation with Total Security Issues

Direction of Causality

		Entirety of Chromium	Chromeos Partition	Resources Partition	Extensions Partition	UI Partition	Other Partition
Layer 1: Exogenous	Architecture Partition						
	File Age						
	Latest LOC						
Layer 2: Architecture Pattern Violations	Clique						
	Crossing						
	ModularityViolation						
	PackageCycle						
	UnhealthyInheritance						
	UnstableInterface						
Layer 3: Interim Outcomes	Bug Churn						
	CoChange						
	NonBug Churn						
	NonBug Commit						
	Weighted CoChange						
Layer 4: Final Outcome	Total Security Issues						

Architecture Pattern Data Size and Rare Events

Percent of Files within Column Header having Value = 0

Legend:
Green = *Direct Causal Evidence*
Yellow = *Indirect Causal Evidence*
Red = No Causal Evidence
Grey = *Not Applicable*

Direction of Causality

Layer	Variable	Entirety of Chromium # Files = 26,281	Chromeos Partition # Files = 3,866	Resources Partition # Files = 2,071	Extensions Partition # Files = 2,024	UI Partition # Files = 6,233	Other Partition # Files = 12,087
Layer 1: Exogenous	Architecture Partition	**0.0%**					
	File Age	**0.2%**	**0.4%**	**0.3%**	**0.0%**	**0.1%**	**0.1%**
	Latest LOC	**0.0%**	**0.0%**	**0.0%**	**30.8%**	**0.0%**	**0.0%**
Layer 2: Architecture Pattern Violations	Clique	**94.0%**	**93.9%**	**99.9%**	**79.6%**	**90.4%**	**95.0%**
	Crossing	**93.5%**	**92.9%**	**99.9%**	**93.2%**	**90.7%**	**94.3%**
	ModularityViolation	**36.9%**	**37.7%**	**24.4%**	**0.0%**	**36.7%**	**38.8%**
	PackageCycle	**80.4%**	**77.3%**	**99.5%**	**98.6%**	**74.9%**	**80.9%**
	UnhealthyInheritance	**93.0%**	**89.9%**	**99.5%**	**92.0%**	**88.7%**	**94.8%**
	UnstableInterface	**98.3%**	**99.7%**	**99.8%**	**38.1%**	**96.8%**	**98.2%**
Layer 3: Interim Outcomes	Bug Churn	**4.8%**	**9.6%**	**4.8%**	**0.0%**	**3.0%**	**4.8%**
	CoChange	**0.0%**	**0.0%**	**0.0%**	**0.2%**	**0.0%**	**0.0%**
	NonBug Churn	**28.8%**	**28.7%**	**28.7%**	**1.2%**	**28.4%**	**28.8%**
	NonBug Commit	**28.7%**	**28.7%**	**28.0%**	**30.9%**	**28.2%**	**28.7%**
	Weighted CoChange	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
Layer 4: Final Outcome	Total Security Issues	**96.7%**	**99.6%**	**98.8%**	**12.2%**	**96.3%**	**96.2%**

Conclusions and Takeaways

- Despite previous statistical attempts with mixed results, to model relationship of architecture pattern violations to security outcomes at the file level, causal search was able to detect some key causal signals
- This latest causal study took advantage of a newly-derived factor of the architecture partition from analyzing the full pathname of each file
- We believe the rare event issue (e.g. number of zero values in the dependent and independent factors) is also making it more difficult for any analytical approach
- Future investigation will include assessing whether using data balancing techniques such as over- and under-sampling might improve the ability to detect causal signals
- The advantage of analyzing causal structures at the architecture pattern level has heightened our sensitivity to whether Simpson's or Lord's paradoxes could be at play. This may necessitate us looking at additional ways to segment the data in further sub-populations.

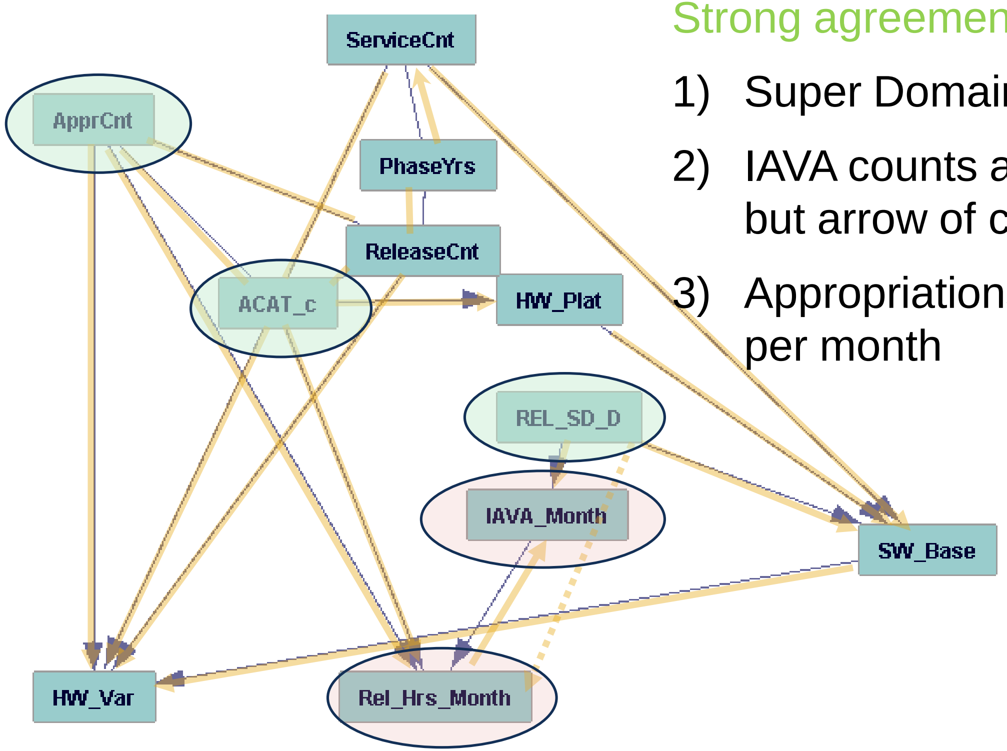
Army Cost and Security Causal Research

Information Assurance Vulnerability Alert (IAVA) Data

Some factors are historical and thus completely exogenous. The middle tier variables may be release dependent. [Legend: Outcome Causality Direct InDirect]

Knowledge	Factor	Description	All	AIS	RT	Eng
Historical	Rel_SD	Super Domain				
Historical	ACAT	Acquisition Category				
Historical	ApprCnt	Number of Funding Sources				
Historical	SericeCnt	Number of Services Supported				
Historical	PhaseYrs	Number of Years in Phase				
Historical	ReleaseCnt	Running count of releases				
Release	HW_Plat	Number of physical platforms				
Release	HW_Var	Number of hardware variations				
Release	SW_Base	Number of software baselines maintained				
Outcome	IAVA_Month	IAVA processed per month				
Outcome	Rel_hrs_Month	Cost in hours per month (Effort)				

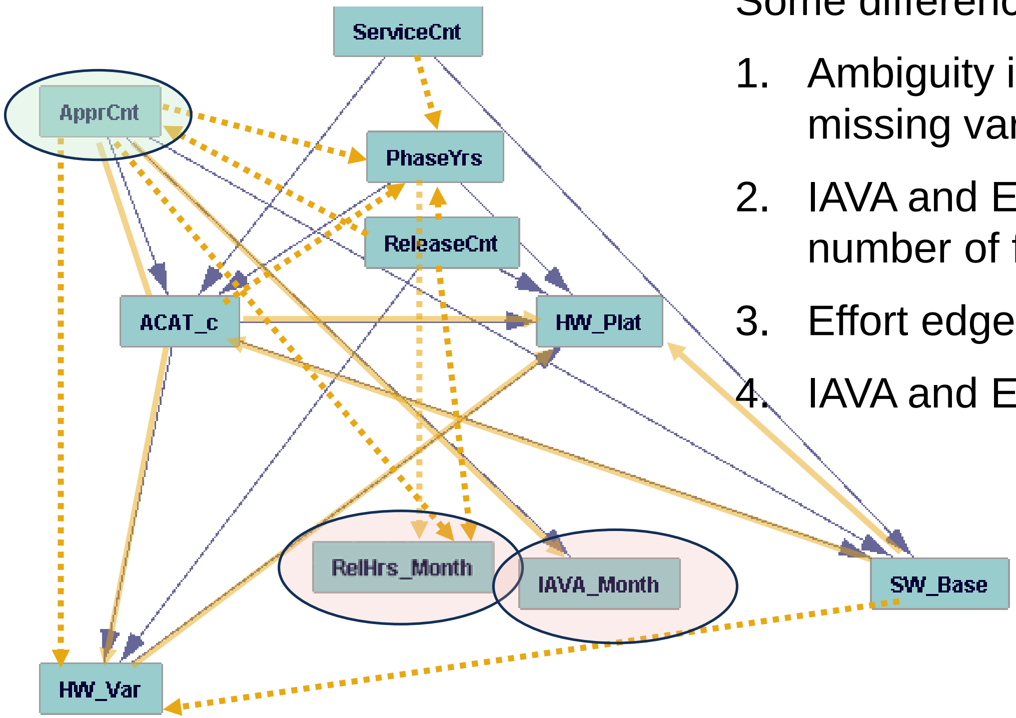
Search including Super Domains



Strong agreement between search algorithms.

1) Super Domain causes both IAVA rate and cost
2) IAVA counts and effort are causally related, but arrow of causality varies by search.
3) Appropriations count and ACAT cause Effort per month

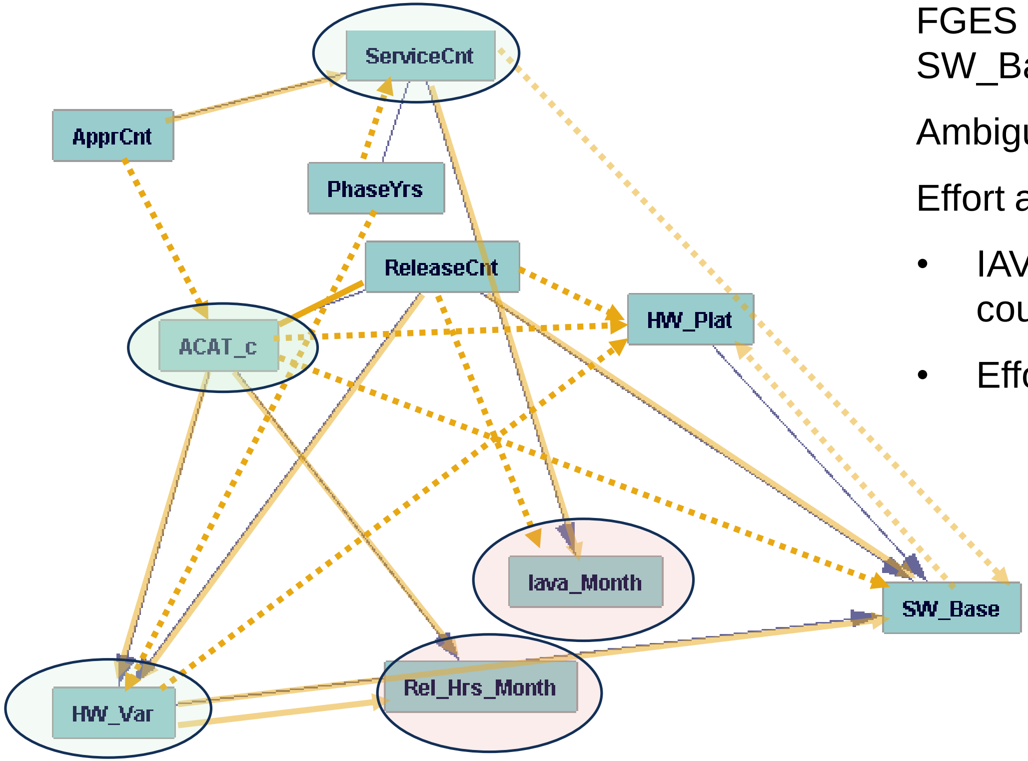
AIS Super Domain



Some differences between the searches

1. Ambiguity in direction of edges can result from missing variables (among other things)
2. IAVA and Effort (RelHrs) both caused by number of funding sources (ApprCnt)
3. Effort edges only appear with FGES (not PC)
4. IAVA and Effort no direct causal connection

Real Time Super Domain



FGES has more edges to HW_Platform and SW_Base, but does not contradict PC.

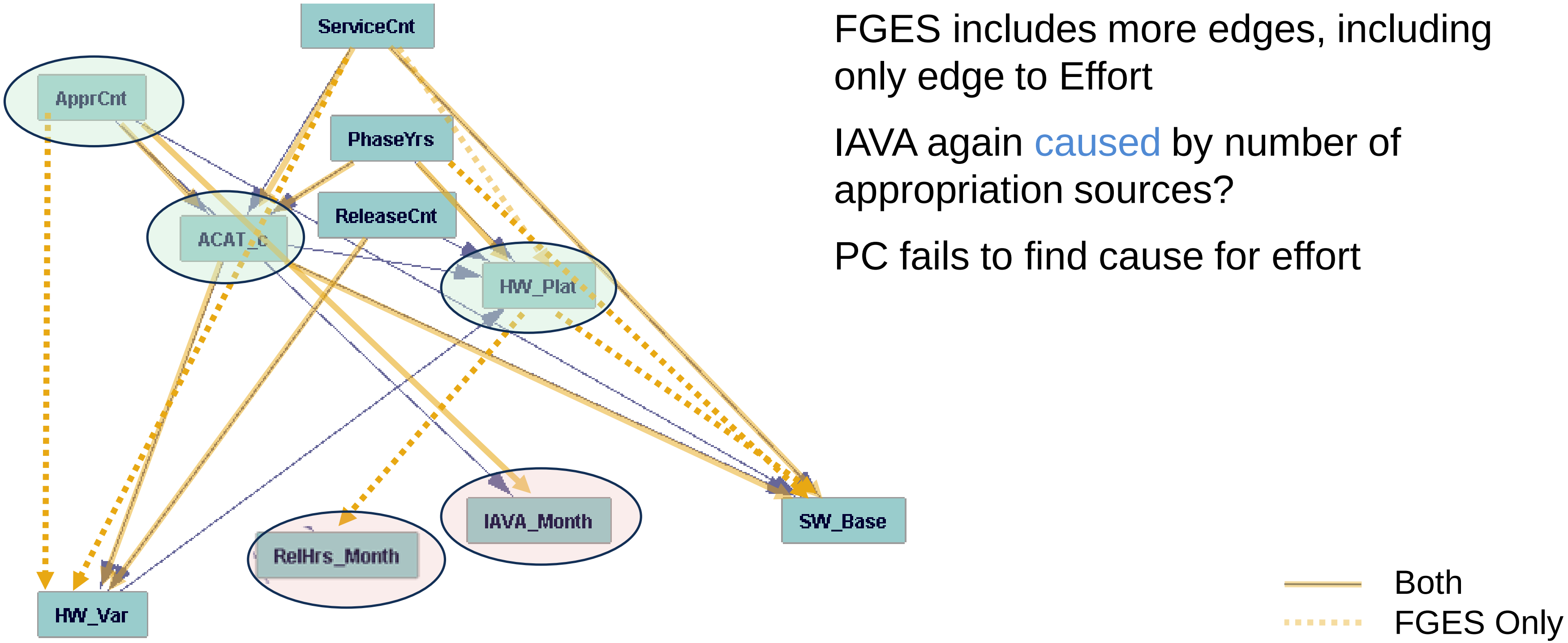
Ambiguous causal directions depending on search

Effort and IAVA are not directly connected

- IAVA caused by Service Count (and Release count in FGES)
- Effort caused by HW_Variants and ACAT

Both
FGES Only
PC Only

Engineering Super Domain



FGES includes more edges, including only edge to Effort

IAVA again caused by number of appropriation sources?

PC fails to find cause for effort

Conclusion and Takeaways from IAVA

Causal search algorithms are mostly, but not always consistent.

Super Domain appears to be an important “cause” for the number of IAVA

Within Super Domain, Appropriation Count is the most common cause. But other causal structures vary. That different Super Domains have different causal structures does not seem surprising.

IAVA and Effort monthly only directly related by Super Domain. (not HW_Plat, HW_Var, or SW_Base) This does seem suprising.

Question: Do funding-related factors really drive IAVA-only Sustainment behavior the way described here? They seem unlikely to be a direct cause. How do they influence other decisions that affect the causal structure? There may be missing (latent) mediators affected by SD and ApprCnt.

Are there other variables we should measure?

Research and validate improved data sets, for example

- Clarify the accounting
  - Is project funding is fixed or variable?
  - Are time periods fixed or variable? Are they connected to releases?
  - Is effort reported as a fixed cost or variable with production (IAVA or features)
  - Counts of full time and part time staff
- Get data for both incoming and closed IAVA, or effort for individual IAVA
- Collect data for smaller batches (aggregation loses variability)
- Explicit release data (frequency, number of IAVA)
- Technical stack (platform type, deploy code changes or binary patches) and so forth

Compute factor loadings to measure effect size.

The key is to find consistent causal systems and sufficient data to characterize them.

Some Practical Next Steps

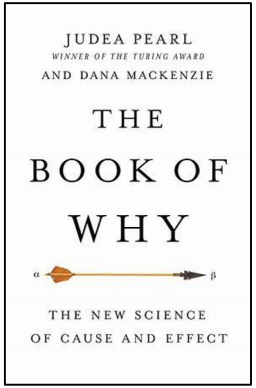
Improve consistency of work and data collection by implementing some of the techniques now common in DevOps (most have been around for years).

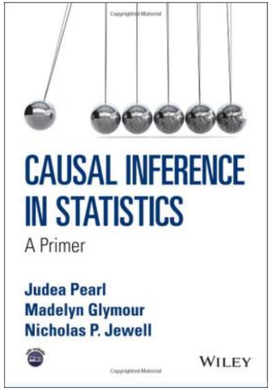
- Replace manual work in build, test, and deployment with automation and scripts.
- Automate collection of metrics (from version control, configuration management, bug report systems) that are currently tedious and/or error prone.

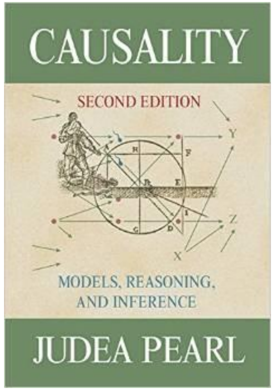
Call To Action

Causal Learning as a Discipline

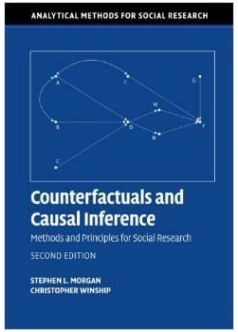
Judea Pearl

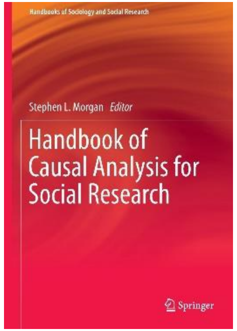


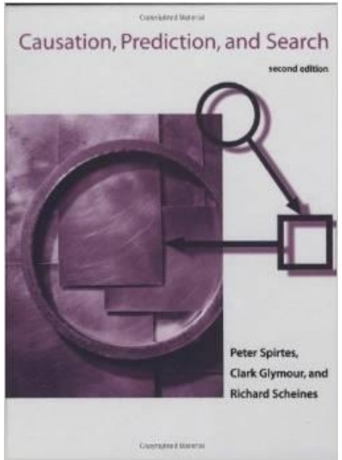




Stephen Morgan







Richard Scheines

David Danks

Clark Glymour

Peter Spirtes

Joe Ramsey

Kun Zhang

Potential SEI Causal Learning Research Applications

Affordable
- Acquisition practice improved using causal models
- Cost estimates and budget execution using causal models
- Simpler but more effective ROI models based on causal factors (e.g. Model Based Engineering, Architecture practice, Technical Debt)

Trustworthy
- Causal factors threatening cyber defenses
- Causal factors limiting resilience
- CL combined with ML tools for more affordable and trustworthy SW technologies (e.g. DOD initiative in Digital Engineering)
- Expected behavior from autonomous systems   (e.g. "*Explainable AI*"; Jensen, UMass)

Capable
- Causal drivers of workforce performance
- SW architecture strategies and tactics driving system performance
- More efficient experimentation of technical solutions
- Increased realism of complex system simulation
- Autonomous systems controlling consequences
- Machine learning with human-like intelligence (e.g. *"Strong AI"; Pearl, "The Book of Why"*)

Timely
- Causal structures from DevOps information stream to control process and lifecycle
- Agile causal systems situationally prescribe practices aligned with goals
- Project risks controlled through causal structures of project parameters

Call to Action

Demand causal knowledge to guide interventions

Engage with SEI causal researchers

Motivate data collection and sharing for more repeatable and reproducible causal studies

Build causal learning competency in your organization

Contact Information

Robert Stoddard
rws@sei.cmu.edu
412-268-1121

Bill Nichols
wrn@sei.cmu.edu
412-268-1727

Mike Konrad
mdk@sei.cmu.edu
412-268-5813

Rick Kazman
Kazman@sei.cmu.edu
412-268-1588

Other SEI Team Members

Chris Miller

Sarah Sheard

Dave Zubrow

Other CMU Collaborators

David Danks

Madelyn Glymour

Joe Ramsey

Kun Zhang