NC STATE UNIVERSITY

SCIENCE OF SECURITY

Impact of Security Research on Practice

Özgür Kafalı and Rahul Pandita

Department of Computer Science
{rkafali,rpandit}@ncsu.edu

June 3, 2016

Agenda for the Morning

- Part I: Identifying the problem
  - How useful is academic research in solving industry problems?
  - Studies from the literature on impact and perception of research
- Part II: Working towards a solution
  - Industry panel: How does industry collaborate with academia regarding security research?
  - Group exercise: How do we perceive useful security research?

Motivation

- Software engineering has been around for almost 50 years
- Studies aim at identifying
  - areas of research with substantial impact
  - research methodologies with relatively more success
  - directions that software engineering research should effectively pursue
- No consensus about the impact of software engineering research as a whole upon software development practice
- Incomplete: results based on a subset of cases

Overview of Studies

- Impact Project [Osterweil et al., 2008]
- Practitioner Perception [Lo et al., 2015]
- Developer Beliefs [Devanbu et al., 2016]

Overview of Studies

- Impact Project [Osterweil et al., 2008]
- Practitioner Perception [Lo et al., 2015]
- Developer Beliefs [Devanbu et al., 2016]
- No such study for security research in particular!

Impact Project

Overview

- Objective: Determining the Impact of Software Engineering Research on Practice
- Specific aims:
  - What future impacts can we expect?
  - What future directions should SE researchers pursue?
- Team includes academic researchers, industrial researchers, and a broad spectrum of software engineering practitioners
- Areas of investigation:
  - Modern Programming languages
  - Software Configuration Management (SCM)
  - Inspections, Reviews, and Walkthroughs
  - Middleware
  - Software Testing and Analysis

---

http://www.sigsoft.org/impact.html

Overview

- Objective: Determining the Impact of Software Engineering Research on Practice
- Specific aims:
  - What future impacts can we expect?
  - What future directions should SE researchers pursue?
- Team includes academic researchers, industrial researchers, and a broad spectrum of software engineering practitioners
- Areas of investigation:
  - ~~Modern Programming languages~~
  - Software Configuration Management (SCM)
  - Inspections, Reviews, and Walkthroughs
  - Middleware
  - ~~Software Testing and Analysis~~

---

http://www.sigsoft.org/impact.html

Methodology & Results: SCM

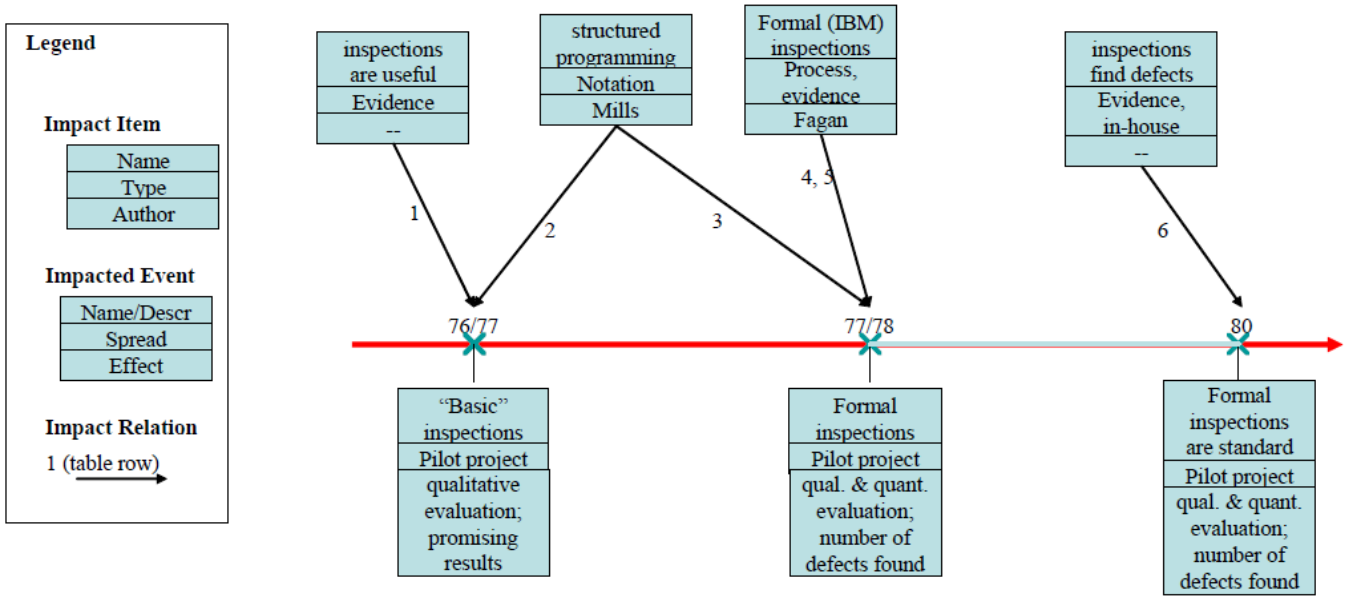
- Managing change in large, complex software systems
- History of landmark contributions: success and failure cases
- Specific case: versioning tools, change sets
- Took time to adopt in practice: cumbersome for large projects

	Academic Research	Industrial Research	Industrial Product
1972		SCCS (Bell Labs)	
1976		Diff (Bell Labs)	
1977		Make (Bell Labs)	
1980	Variants, RCS (Purdue University)		
1980		Change-sets (Xerox Parc)	
1982	Merging, and/or graph (Purdue University)		
1983		Change-sets (Aide-de-Camp)	
1984	Selection (Grenoble University)		
1985		System model (DSEE)	
1988	*First International SCM workshop*		
1988	Process support (Grenoble University)		
1988	NSE Workspaces (Carnegie Mellon University; Sun)		
1990		3DFS, nDFS virtual file system (Bell Labs)	
1994		Virtual file system (ClearCase)	
1994		MultiSite (ClearCase)	
1996		Activity-oriented SCM (Asgard, Bell Core)	
2000	WebDAV/DeltaV (University of California, Irvine, Microsoft, ClearCase, . . .)		

Methodology & Results: Inspections, Reviews ...

- Methodology
  - Identify research on reviews and trace forward organizations that apply them
  - Identify success cases in practice and trace back the impact of research on them
- Success measures from companies such as Allianz, Motorola or IBM up to
  - 95% defect detection rates
  - 50% cost reduction
  - 50% delivery time reduction

Impact Trace: NASA Software Engineering Laboratory

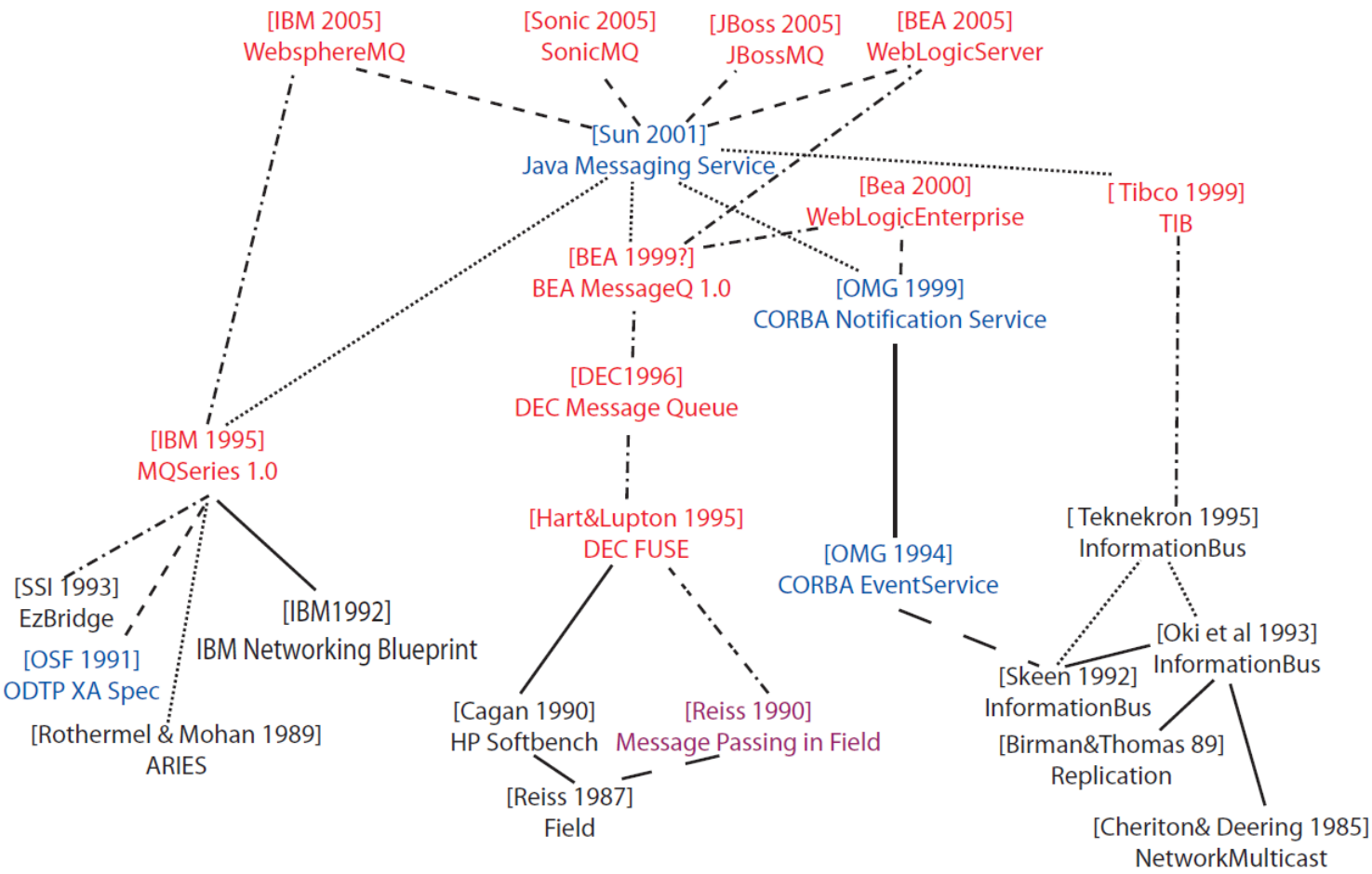


#	Impact Item	Medium	Proximity	Impacted event	Documentation
1	Evidence "inspections are useful"	"in the air"	external	Introduction of "basic" inspections	McGarry
2	Structured Programming	Class (Mills/Basili)	close		McGarry
3	Structured Programming	Class (Mills/Basili)	close	Introduction of Fagan / formal inspections	McGarry
4	Formal IBM inspections	Fagan paper [Fagan76]	external		McGarry
5	Fagan's visit and talks at SEL	Tutorial	Close / In-house		McGarry
6	Inspections useful (in terms of number of defects found)	(internal results)	In-house	Declaration of formal inspections as standard	McGarry

Methodology & Results: Middleware

- Where does successful middleware products originate from?
- Report impact trees as proof
- Resources:
  - Articles
  - Phd theses
  - Technical reports
  - Meeting notes

Impact Tree: Java Message Service



Key Findings

- Technology transfer
  - Takes time: 15-20 years from publication to product
  - Impact usually connected to PhD thesis
  - People movement most effective (in either direction)
- Putting ideas “in the air” via meetings / workshops
- Interdisciplinary research
  - Impact traces often include different CS disciplines
  - Sometimes larger impact in an area different than intended by publication, e.g., from operating systems to databases and eventually to object-oriented concepts and application servers
- Challenges (specifically for reviews, but probably generalizable)
  - Management support (some ideas take longer time to adopt)
  - Technology champion (drive technology, maintain training)
  - Convincing developers (time pressure makes adoption harder)

Key Findings

- Technology transfer
  - Takes time: 15-20 years from publication to product Be patient!
  - Impact usually connected to PhD thesis
  - People movement most effective (in either direction)
- Putting ideas “in the air” via meetings / workshops
- Interdisciplinary research
  - Impact traces often include different CS disciplines
  - Sometimes larger impact in an area different than intended by publication, e.g., from operating systems to databases and eventually to object-oriented concepts and application servers
- Challenges (specifically for reviews, but probably generalizable)
  - Management support (some ideas take longer time to adopt)
  - Technology champion (drive technology, maintain training)
  - Convincing developers (time pressure makes adoption harder)

Key Findings

- Technology transfer
  - Takes time: 15-20 years from publication to product Be patient!
  - Impact usually connected to PhD thesis Support students!
  - People movement most effective (in either direction)
- Putting ideas “in the air” via meetings / workshops
- Interdisciplinary research
  - Impact traces often include different CS disciplines
  - Sometimes larger impact in an area different than intended by publication, e.g., from operating systems to databases and eventually to object-oriented concepts and application servers
- Challenges (specifically for reviews, but probably generalizable)
  - Management support (some ideas take longer time to adopt)
  - Technology champion (drive technology, maintain training)
  - Convincing developers (time pressure makes adoption harder)

Practitioner Perception

"How practitioners perceive the relevance of software engineering research"

10th ESEC-FSE 2015

Number of Software Engineering papers grow over time:

- How do practitioners view software engineering research as a whole?
- What research ideas do practitioners consider to be most important?
- Why practitioners view some research ideas as unwise?

Adapted from author ESEC-FSE presentation slides with permission from authors

"How practitioners perceive the relevance of software engineering research"

10th ESEC-FSE 2015

Number of Software Engineering papers grow over time:

- How do practitioners view software engineering research as a whole?
- ~~What research ideas do practitioners consider to be most important?~~
- Why practitioners view some research ideas as unwise?

Adapted from author ESEC-FSE presentation slides with permission from authors

Study Methodology

- Use practitioners as a sounding board of high-level research ideas
- Get practitioners feedback on the relevancy of software engineering studies from their perspectives
- Assess the degree-of-disconnect between researcher and practitioners

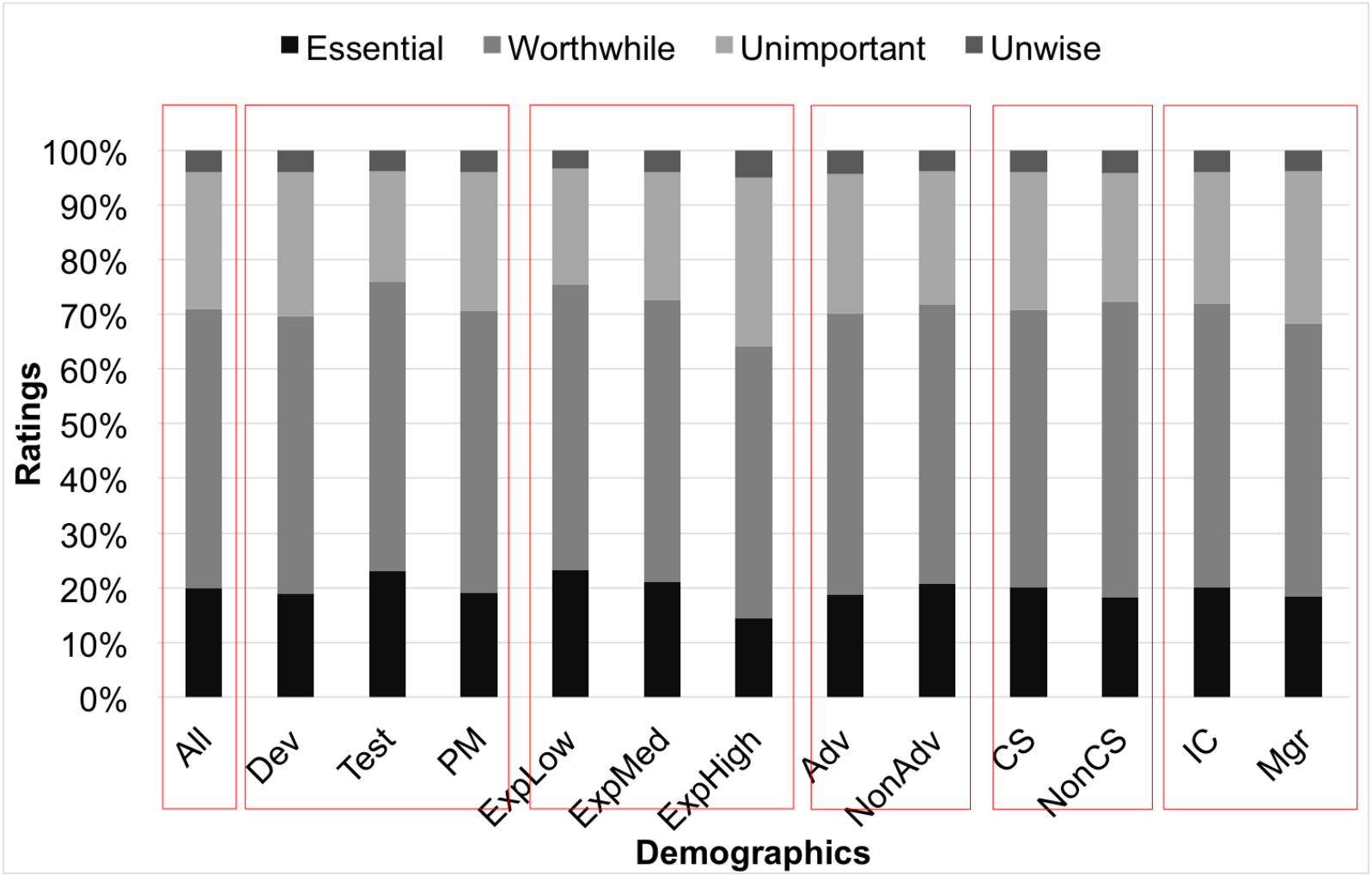
Adapted from author ESEC-FSE presentation slides with permission from authors

Study Methodology

- Use practitioners as a sounding board of high-level research ideas
- Get practitioners feedback on the relevancy of software engineering studies from their perspectives
- Assess the degree-of-disconnect between researcher and practitioners *Health of software engineering research!*

Adapted from author ESEC-FSE presentation slides with permission from authors

PRACTITIONER PERCEPTION



Adapted from author ESEC-FSE presentation slides with permission from authors

Why Unwise?

- A tool not needed. *...would not be something I would use...*
- An empirical study is not actionable. *...since enough is known about common fallacies of this type...*
- Generalizability issue. *...lessons learned...can be very specific...*
- Scalability issue. *...I dont see this being used for large-scale systems...*
- Cost outweighs benefit. *...I believe the cost of implementing and maintain such a solution would be greater...*

Why Unwise? Cont...

- Questionable assumptions about inputs or conditions.
  ...Description is often not filled correctly. hence it is unwise to rely on it...
- Another solution seems better. *...I dont think natural language is that important. Instead helping users find the keywords or tags is should be the focus...*
- Proposed solution has side effects. *...Drag and drop solutions have always seemed to me as a quick and easy way to write inefficient code...*
- Disbelief in a particular technology or methodology. *...I dont believe in design patterns, force fitting something into a pattern is not wise...*

Why Unwise? Cont...

- Questionable assumptions about inputs or conditions.
  ...Description is often not filled correctly. hence it is unwise to rely on it...
- Another solution seems better. *...I dont think natural language is that important. Instead helping users find the keywords or tags is should be the focus...*
- Proposed solution has side effects. *...Drag and drop solutions have always seemed to me as a quick and easy way to write inefficient code...*
- **Disbelief in a particular technology or methodology.** ***...I dont believe in design patterns, force fitting something into a pattern is not wise...***

Developer Beliefs

Belief & Evidence in Empirical Software Engineering

ICSE 2016

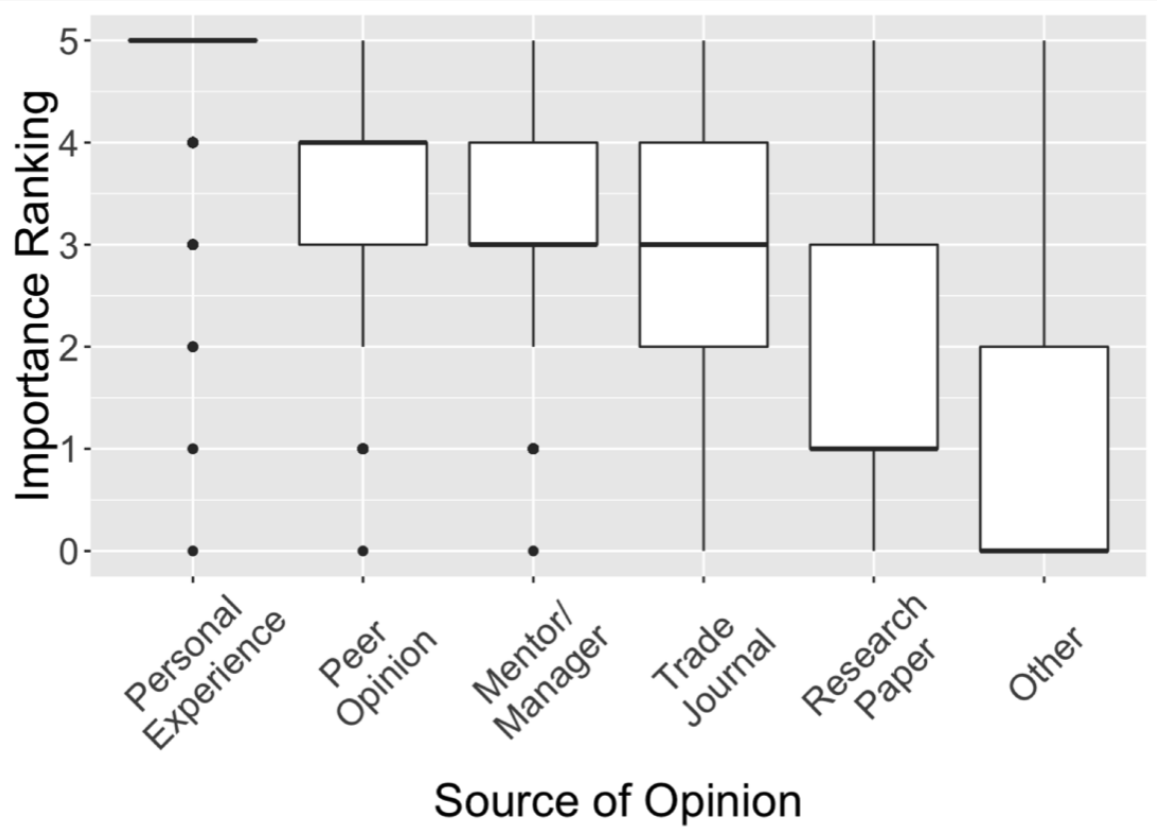
- Engineers
  - Highly Trained, Opinionated, Professionals
  - Increasing evidence on important SE Issues (but no such thing as goodprogramming.gov)
  - Do software engineers pay attention to evidence? To research?

Adapted from author's ICSE presentation slides with permission

Belief & Evidence in Empirical Software Engineering

Adapted from author's ICSE presentation slides with permission

Opinion Formation



Adapted from author's ICSE presentation slides with permission

Belief & Evidence in Empirical Software Engineering

ICSE 2016

- Source of opinions: NOT necessarily Scientific Evidence.
- Developers beliefs vs. Evidence disparity.
- Emphasizes importance of Evidence-based Software Engineering..

Adapted from author's ICSE presentation slides with permission

Discussion

Borrowing Ideas for Security Research

- How can we apply these ideas to measure the impact of security research as well as the perception of practitioners?
  - What sort of results do practitioners look for in security research?
  - Does it align with the types of studies academic researchers are comfortable doing?