Dare to Share: Risks and Rewards of Artifact Sharing in Computer Science

ACSAC 2017

Christian Collberg Todd Proebsting

Department of Computer Science
University of Arizona

http://repeatability.cs.arizona.edu
http://findresearch.org

Supported by *the private foundation that must not be named*

Opening Gambit
The Deception Study
8 Artifact Sharing Proposals
8 Laws of Artifact Sharing
Risks and Rewards

Some Computer Security Paper

Well-known Authors

Abstract

We present a new general technique for protecting clients in distributed systems against *Remote Man-at-the-end* (R-MATE) attacks. Such attacks occur in settings where an adversary has physical access to an untrusted client device and can obtain an advantage from tampering with the hardware itself or the software it contains.

In our system, the trusted server overwhelms the untrusted client's analytical abilities by continuously and automatically generating and pushing to him diverse client code variants. The diversity subsystem employs a set of primitive code transformations that provide an ever-changing attack target for the adversary, making tampering difficult without this being detected by the server.

1. Introduction

Man-at-the-end (MATE) attacks occur in settings where an adversary has physical access to a device and compromises it by tampering with its hardware or software. *Remote* man-at-the-end (R-MATE) attacks occur in distributed systems where *untrusted clients* are in frequent communication with *trusted servers* over a network, and malicious user can get an advantage by compromising an untrusted device.

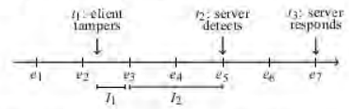
To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, in the *Advanced Metering Infrastructure* (AMI) for controlling the electrical power grid, networked devices ("*smart meters*") are installed at individual house-holds to allow two-way communication with control servers of the utility company. In an R-MATE attack against the AMI, a malicious consumer tampers with the meter to emulate an imminent blackout, or to trick a control server to send disconnect commands to other customers [7, 21]. Second, massive multiplayer online games are susceptible to R-MATE attacks since a malicious player who tampers with the game client can get an advantage over other players [16]. Third, wireless sensors are often deployed in unsecured environments (such as theaters of war) where they are vulnerable to tampering attempts. A compromised sensor could be coached into supplying the wrong observations to a base station, causing real-world damage. Finally, while electronic health records (EHR) are typically protected by encryption while stored in databases and in transit to doctors' offices, they are vulnerable to R-MATE attack if an individual doctor's client machine is compromised.

1.1 Overview

In each of the scenarios above the adversary's goal is to tamper with the client code and data under his control. The trusted server's goal is to *detect* any such integrity violations, after which countermeasures (such as severing connections, legal remedies, etc.) can be launched.

Security mechanisms. In this paper we present a system that achieves protection against R-MATE attacks through the extensive use of code diversity and continuous code replacement. In our system, the trusted server continuously and automatically generates diverse variants of client code, pushes these code updates to the untrusted clients, and installs them as the client is running. The intention is to force the client to constantly analyze and re-analyze incoming code variants, thereby overwhelming his analytical abilities, and making it difficult for him to tamper with the continuously changing code without this being detected by the trusted server.

Limitations. Our system specifically targets distributed applications which have frequent client-server communication, since client tampering can only be detected at client-server interaction events. Furthermore, while our use of code diversity can *delay* an attack, it cannot completely *prevent* it. Our goal is therefore the rapid *detection* of attacks; applications which need to completely prevent any tampering of client code, for even the shortest length of time, are not suitable targets for our system. To see this, consider the following timeline in the history of a distributed application running under our system:



The e_i's are *interaction events*, points in time when clients communicate with servers either to exchange application data or to perform code updates. At time t_1 the client tampers with the code under his control. Until the next interaction event, during interval I_1, the client runs autonomously, and the server cannot detect the attack. At time t_2, after an interval I_2 consisting of a few interaction events, the client's tampering has caused it to display anomalous behavior, perhaps through the use of an outdated communication protocol, and the server detects this. At time t_3, finally, the server issues a response, perhaps by shutting

tigress.cs.arizona.edu

Some Computer Security Paper

Authors

To: authors@cs.ux.edu

Cool paper! Can you send me your system so I can break it? 😀

Thanks!
Christian

To: authors@cs.ux.edu

Cool paper! Can you send me your system so I can break it? 😀

Thanks!
Christian

Reimplement!

reimplement.hs

```
type operator =
      | A
      | B of operand * value * binop
      | C of operand * value * operand * binop
      | D of operand * value * operand * binop
      | E of operand * operand
```

```
U:---  reimplement.hs   All L1     (Lisp Interaction)
```

Technical Report

PhD Thesis

Conference Paper

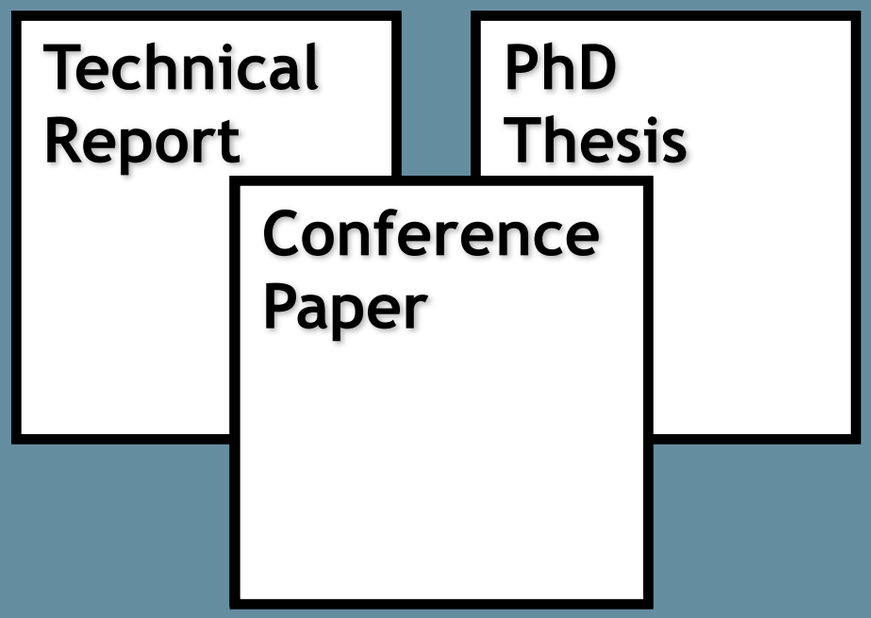
To: authors@cs.ux.edu

f: never used!

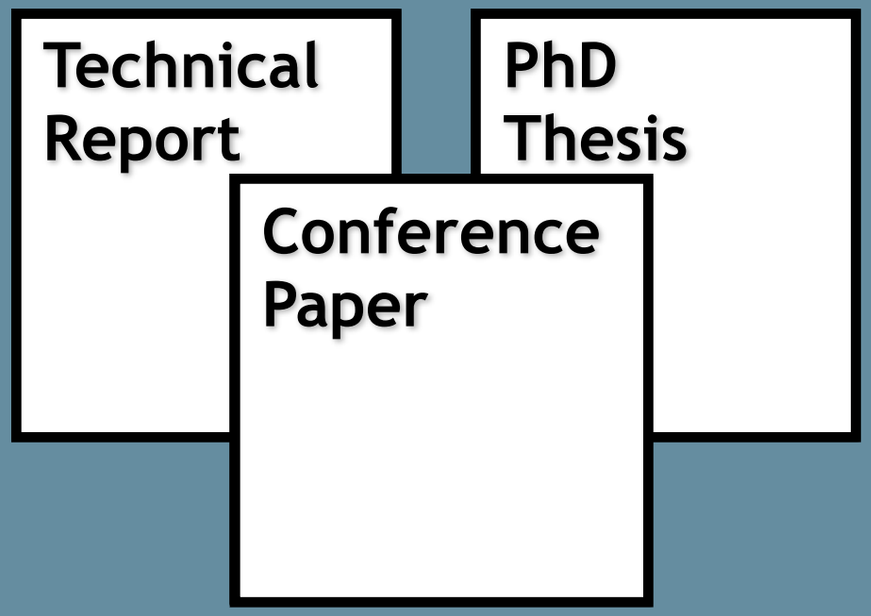
g: not defined!

h: doesn't type check!

i: different in TR and paper!

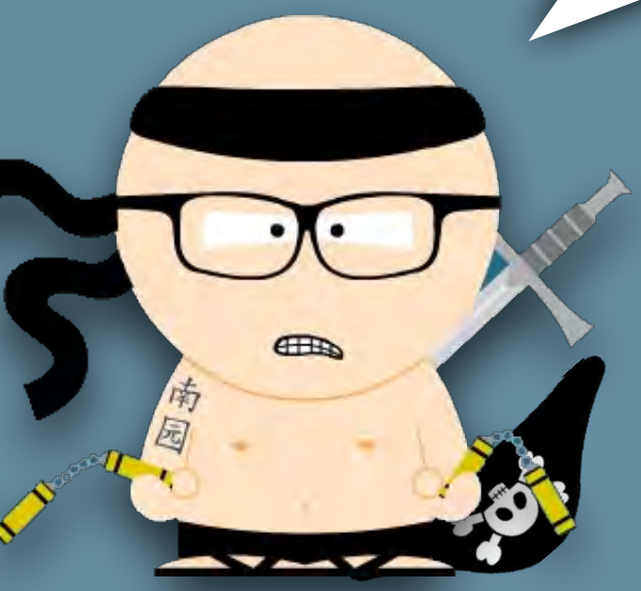


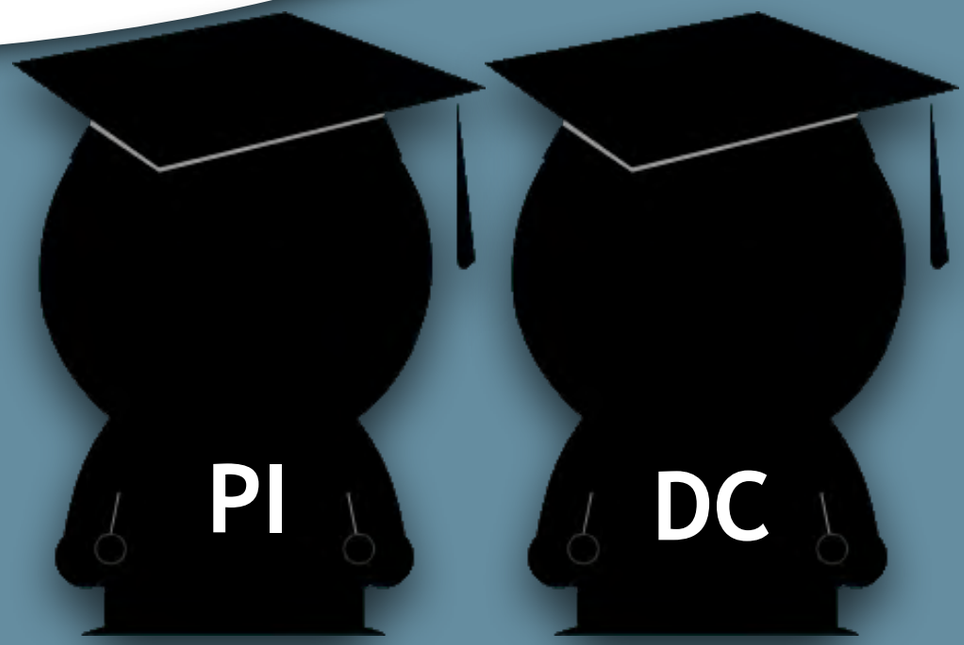
I … have **few recollections of the work**. [It was] like seeing a new paper for the first time.



To: PI,DC@cs.ux.edu

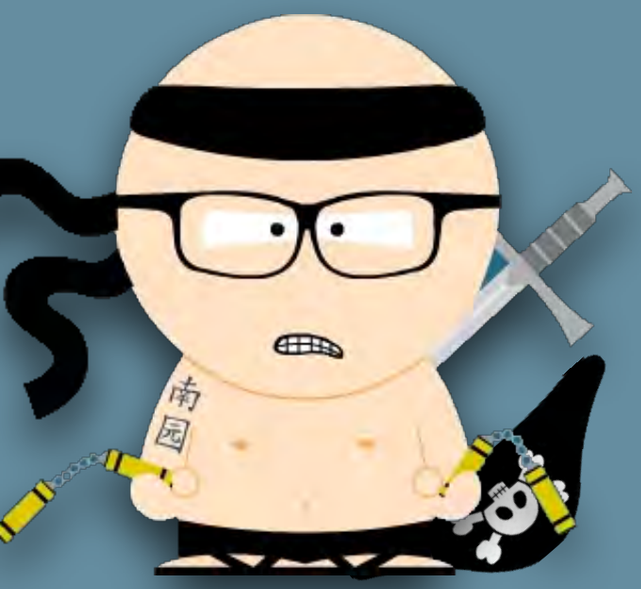
Request under the OPEN RECORDS ACT … ALL RESEARCH ARTIFACTS …





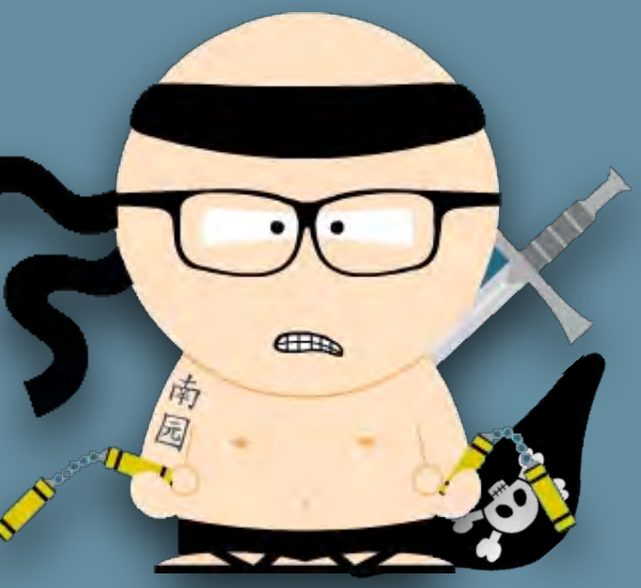
From: legal@cs.ux.edu

… to the extent such records may exist, **they will not be produced pursuant to ORA**.

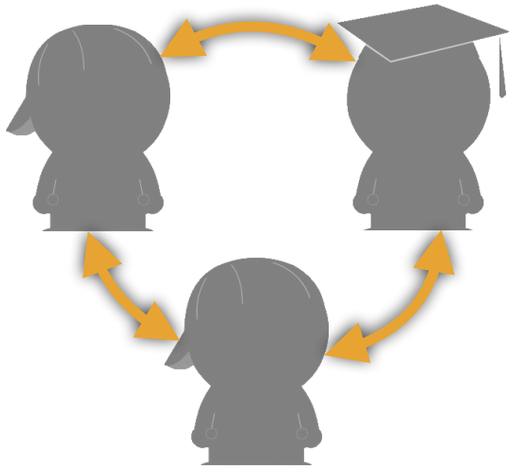


From: legal@cs.ux.edu

… and no, they don’t exist…



Really?



PhD
Thesis

Pursuant to ORA, I request copies of **all electronic mail…**

… a total cost of **$2,263.66** to search for, retrieve, redact and produce such records.

Grant application #: [redacted]

We will also make our data and software available to the research community when appropriate.

Consequences

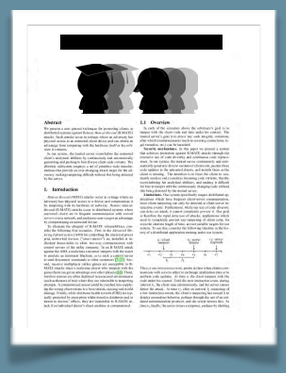
By not sharing their artifacts, and by (perhaps unintentionally) leaving holes in their publications, the authors have effectively guaranteed that their claims can never be refuted.

Consequences

By not sharing their artifacts, and by (perhaps unintentionally) leaving holes in their publications, the authors have effectively guaranteed that their claims can never be refuted.

8th Law of Artifact Sharing
(Pretschner's Law)

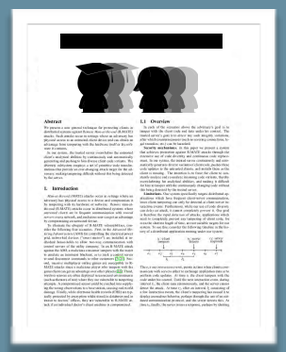
The probability of getting code out of someone is inversely proportional to the outrageousness of the claims in the paper.

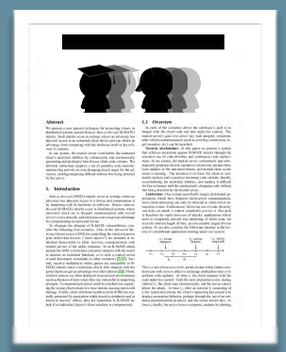


Research Artifacts

- Code
- Data
- Experiments
- …

Repeatability



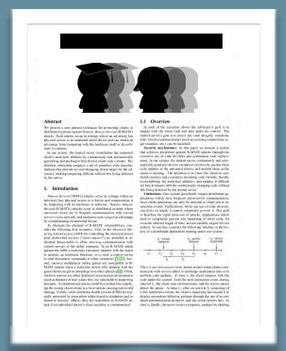


Verify results

Research Artifacts

Research Artifacts

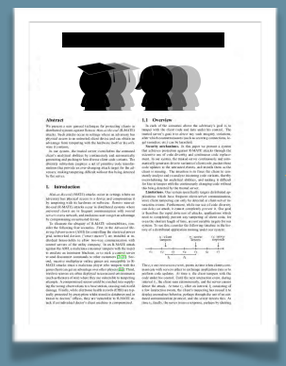
- **Code**
- **Data**
- **Experiments**
- **…**



Research Artifacts

- Code
- Data
- Experiments
- …

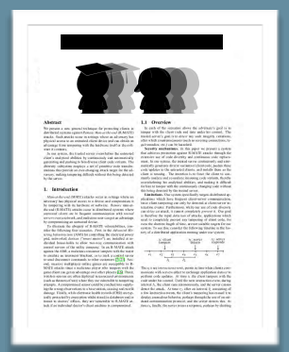
Repeatability



Verify results

Research Artifacts

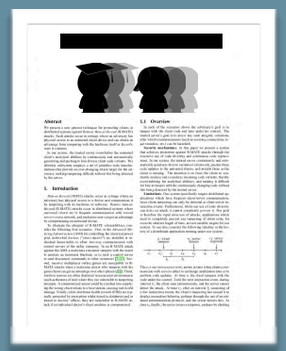
Reproducibility



\+

New Experiment

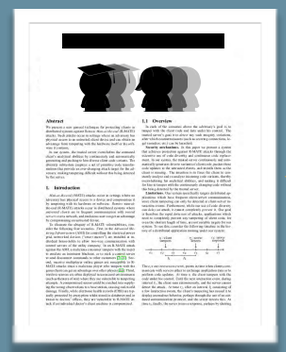
→ **Confirm Hypothesis**



Research Artifacts

- **Code**
- **Data**
- **Experiments**
- **…**

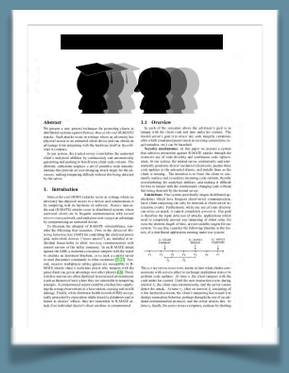
Repeatability



Verify results

Research Artifacts

Reproducibility



+

New Experiment

Confirm Hypothesis

Benefaction

Research Artifacts → **Build upon** → **New Artifacts**

The Deception Study

601 Research Papers

601 Research Papers

Has code?

601 Research Papers

Has code?

Can we find it?

1. Article?
2. Web?
3. Email?

601 Research Papers

Has code?

Can we find it?

1. Article?
2. Web?
3. Email?

Does it “work”?

1. ≤30 mins?
2. >30 mins?
3. Author?

601 Research Papers

Has code?

Can we find it?

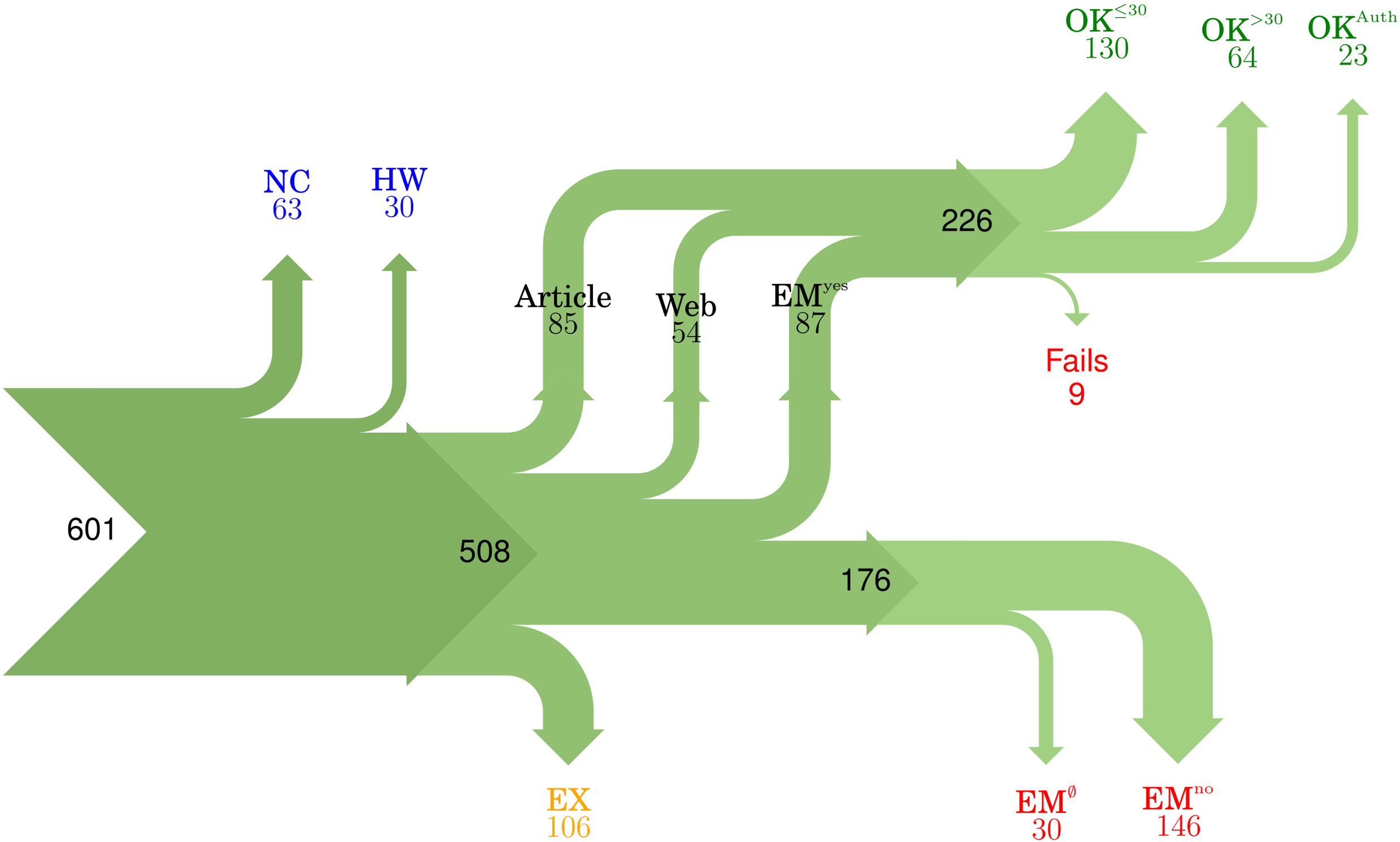
1. Article?
2. Web?
3. Email?

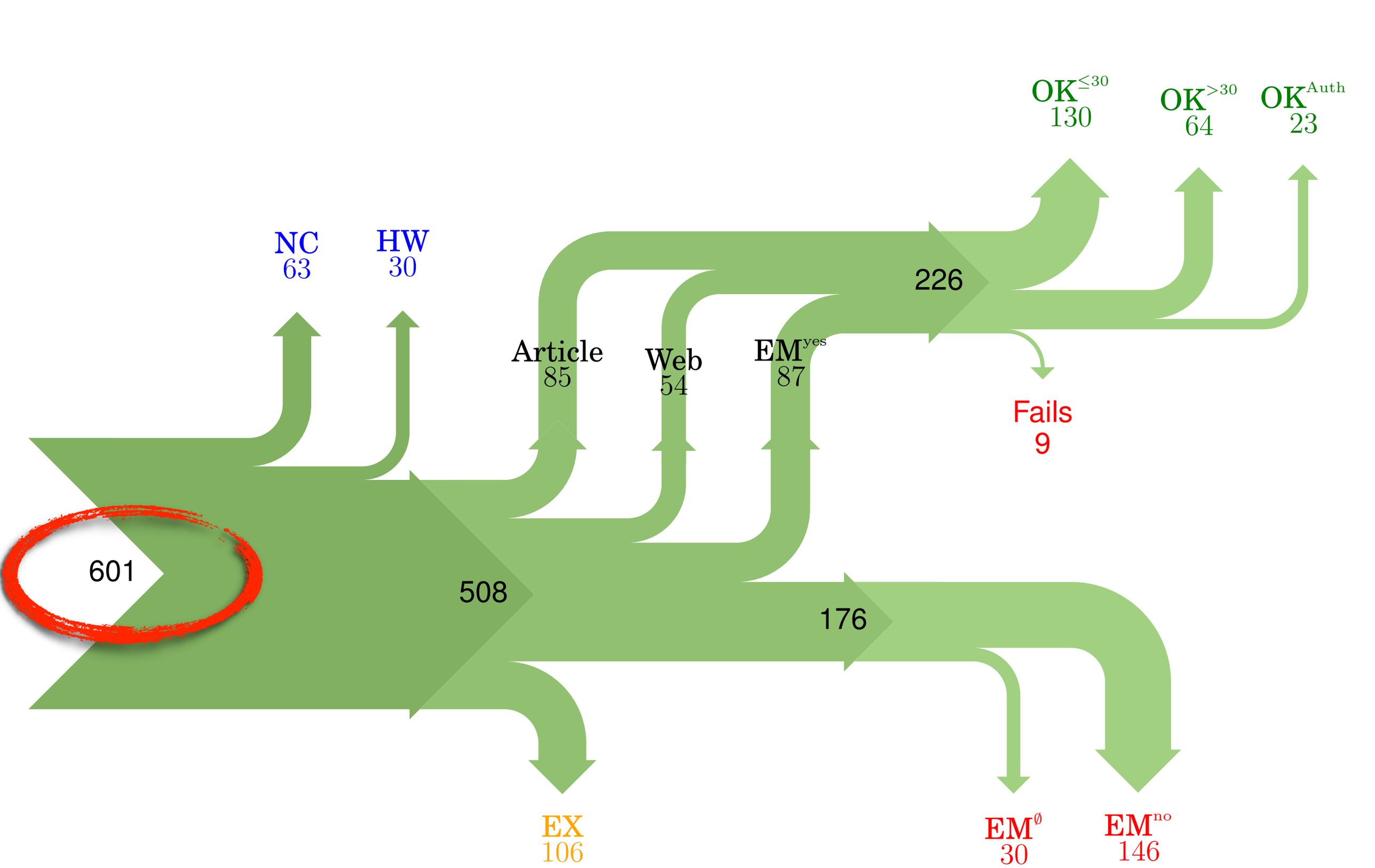
Does it “work”?

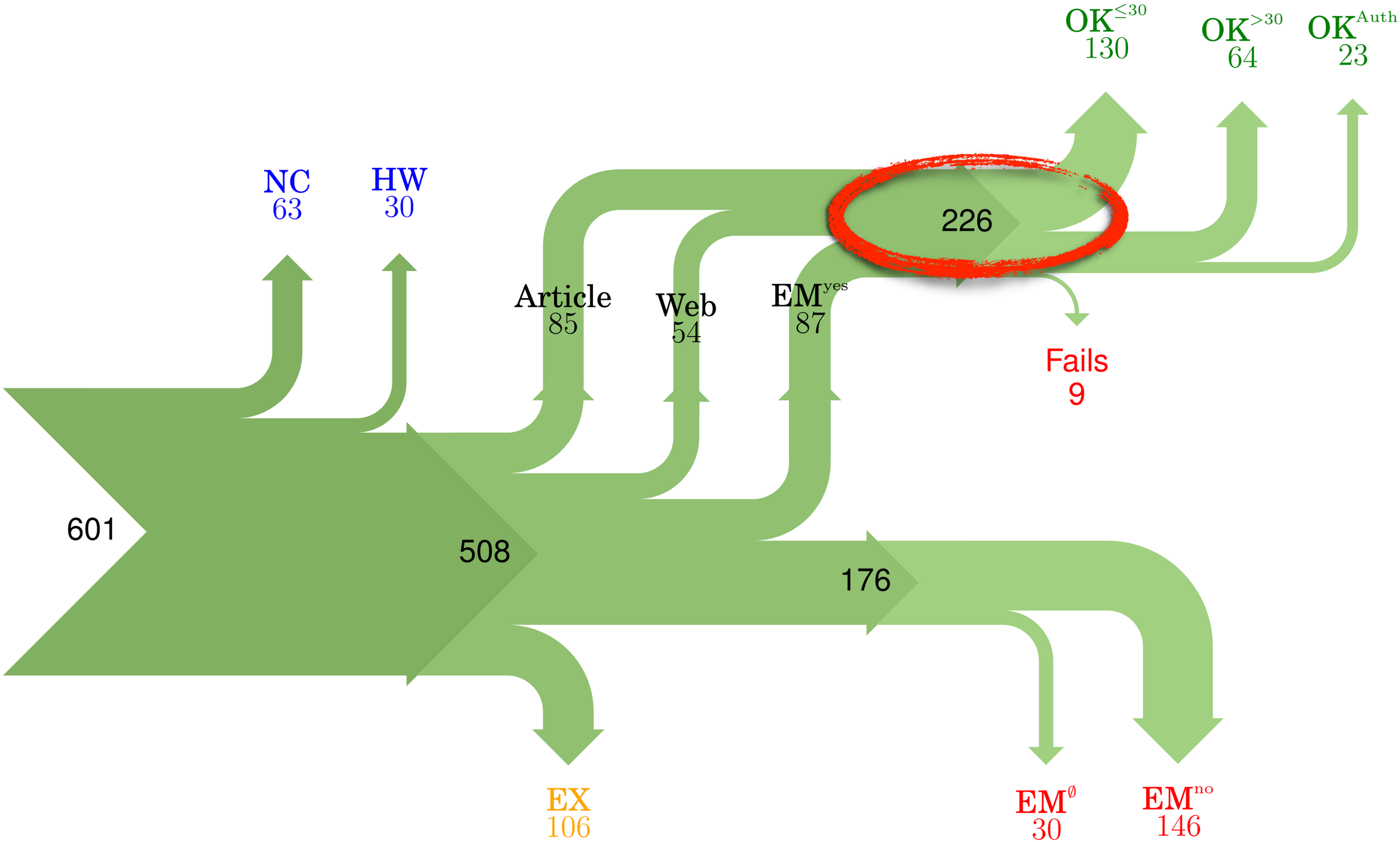
1. ≤30 mins?
2. >30 mins?
3. Author?

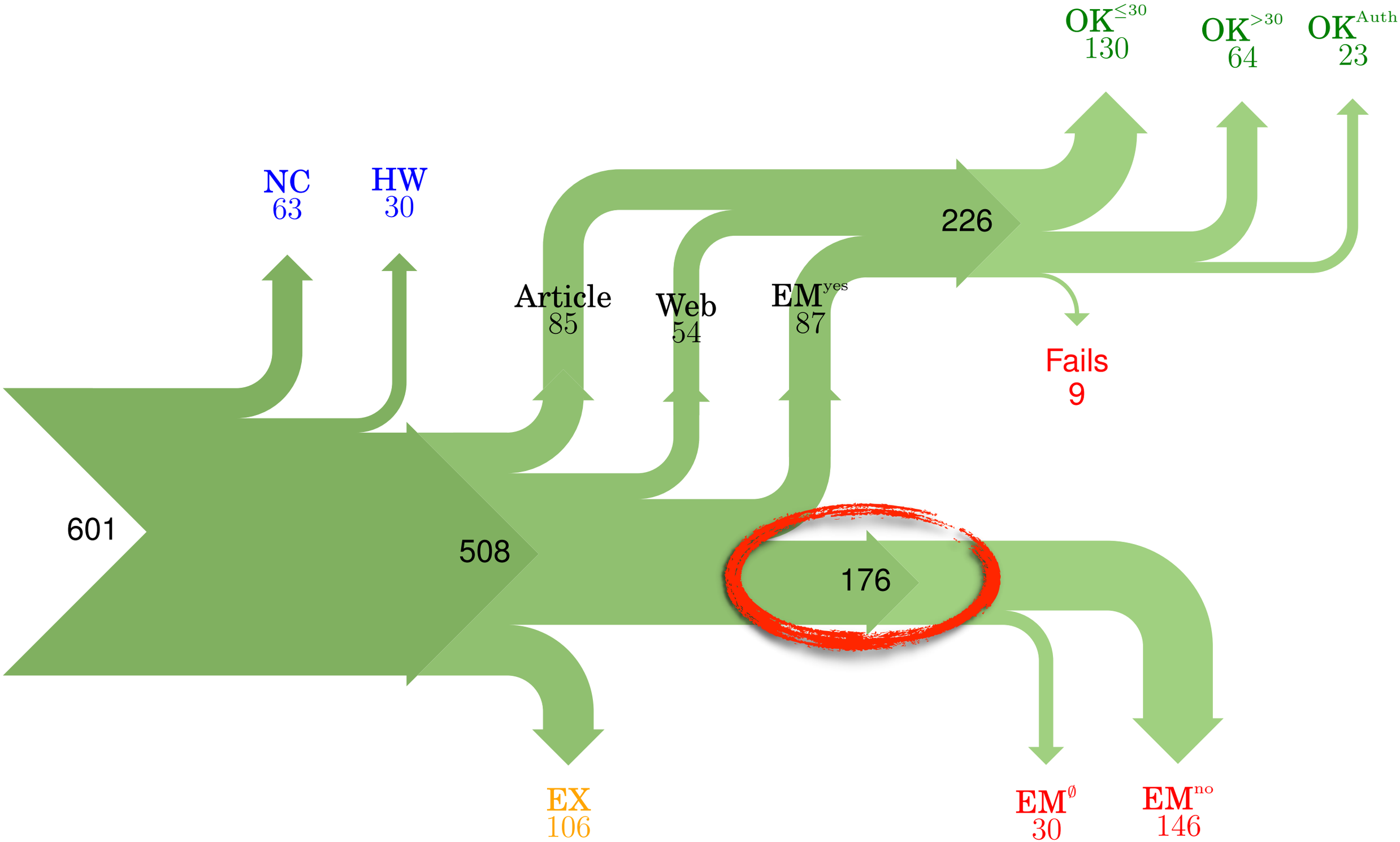
Weakly Repeatable

Authors share their code, and it builds.


NC
63
HW
30
601
Article
85
Web
54
EM^yes
87
508
226
OK^≤30
130
OK^>30
64
OK^Auth
23
Fails
9
176
EX
106
EM^∅
30
EM^no
146


NC
63
HW
30
Article
85
Web
54
EM^yes
87
226
OK^≤30
130
OK^>30
64
OK^Auth
23
Fails
9
601
508
176
EX
106
EM^∅
30
EM^no
146


NC
63
HW
30
Article
85
Web
54
EM^yes
87
226
OK^≤30
130
OK^>30
64
OK^Auth
23
Fails
9
601
508
176
EX
106
EM^∅
30
EM^no
146


NC
63
HW
30
Article
85
Web
54
EM^yes
87
226
OK^≤30
130
OK^>30
64
OK^Auth
23
Fails
9
601
508
176
EX
106
EM^∅
30
EM^no
146

The good news … I was able to find some code. I am just **hoping** that it … **matches the implementation** we … used for the paper.

Versioning

Unfortunately the **current system is not mature** … We are actively working on a number of extensions … Soon …

Available Soon

[Our] prototype … included many moving pieces that only [student] knew how to operate … **he left**.

Personnel Issues

[Our] prototype … included many moving pieces that only [student] knew how to operate … **he left**.

Personnel Issues

… the server in which my implementation was stored had a **disk crash** … three disks crashed … Sorry for that.

Lost Code

… the server in which my implementation was stored had a **disk crash** … three disks crashed … Sorry for that.

Lost Code

The code … is … **hardly usable by anyone** other than the authors … due to our decision to use [obscure variant of obscure language]

Design Issues

7th Law of Artifact Sharing

Unless a project starts with the express goal of post-publication artifact sharing, getting the right code, in a timely fashion, out of the project is virtually impossible.

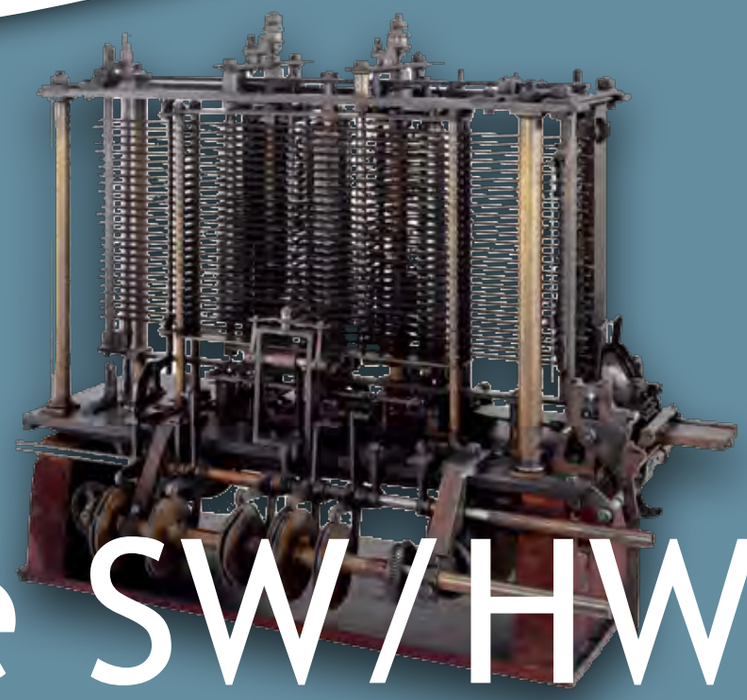
We will not provide the software … [because we spent] **more time getting outsiders up to speed than on our own research**.

Academic Tradeoffs

… **we can't share what did for this paper**. … this is not in the academic tradition, but this is a hazard in an industrial lab.

Industrial Lab Tradeoffs

We have no plans to make the scheduler's source code publicly available … because [ancient OS] as such **does not exist anymore**.



Obsolete SW/HW

Based on bad experiences, we don't want our system **used in situations that it wasn't meant for.** Tell us if you use it to publish comparisons with other techniques.

Controlled Usage

We have an agreement with the [business], and we cannot release the code because of the potential **privacy risks** …

Privacy/Security

Available Soon…

Versioning

Personnel

Obsolete SW/HW

Academic Pressure

Licensing

Don’t want

Fear

Poor Design

Industrial Lab Issues

Privacy/ Security

Sharing Proposal
— #1 —

Artifact Curation

Sharing Proposal
— #1 —

Artifact Curation

ACM Artifact Curation

Refactoring Java Generics by Inferring Wildcards, In Practice

John Altidor
University of Massachusetts
jaltidor@cs.umass.edu

Yannis Smaragdakis
University of Athens
smaragd@di.uoa.gr

ACM DL DIGITAL LIBRARY

APPENDICES and SUPPLEMENTS

- artifact_overview.pdf (100 KB) Artifact Overview for Paper #35 of OOPSLA 2014
- oopsla035.zip (95.77 MB) Please, email questions to jaltidor@cs.umass.edu
- VarJ.zip (95.77 MB) Please, email questions to jaltidor@cs.umass.edu

Home | re3data.org

Secure | https://www.re3data.org

Search Browse Suggest Resources Contact DataCite

re3data.org

REGISTRY OF RESEARCH DATA REPOSITORIES

Search... Search

Search

Register | Sign In

runmycode

Home | Our goals | Researchers | Journal Editors | FAQ | Our partners | News

RunMyCode enables scientists

to openly share the code and data that underlie their research publications

This service is based on the innovative concept of a companion website associated with a scientific publication.

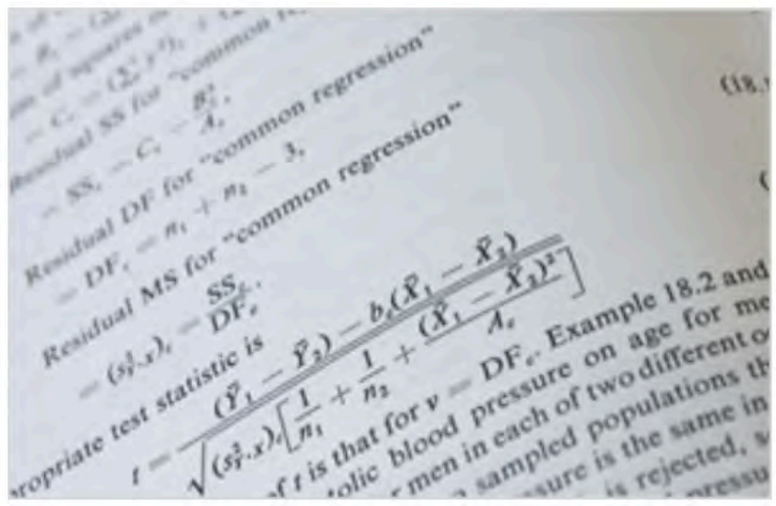
Your email

Create a password

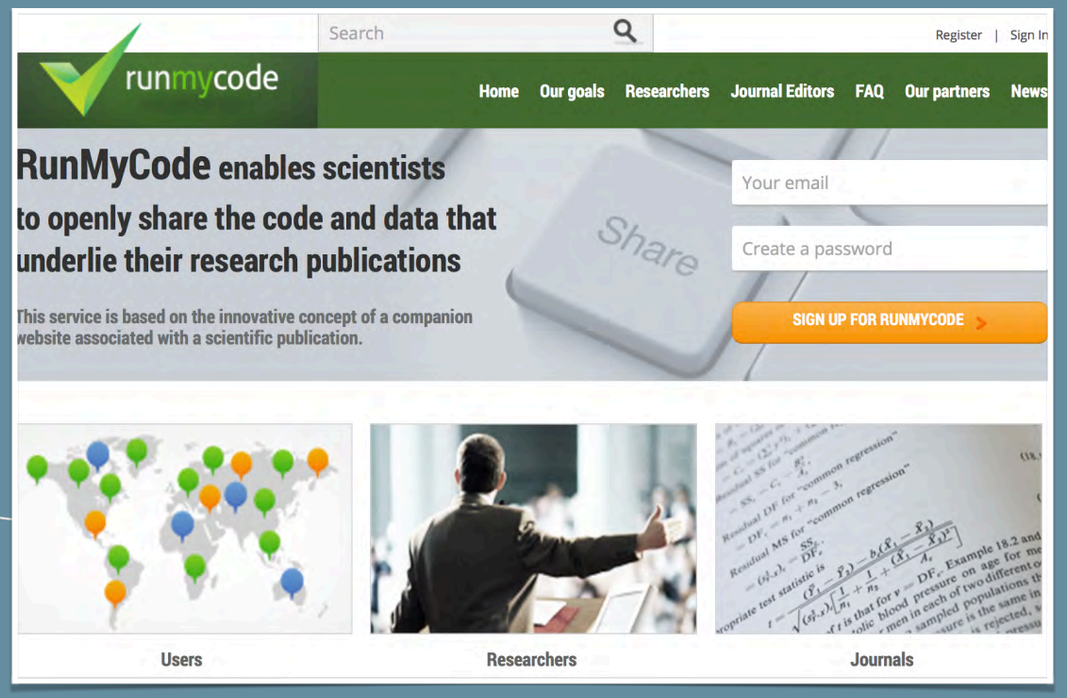
SIGN UP FOR RUNMYCODE >

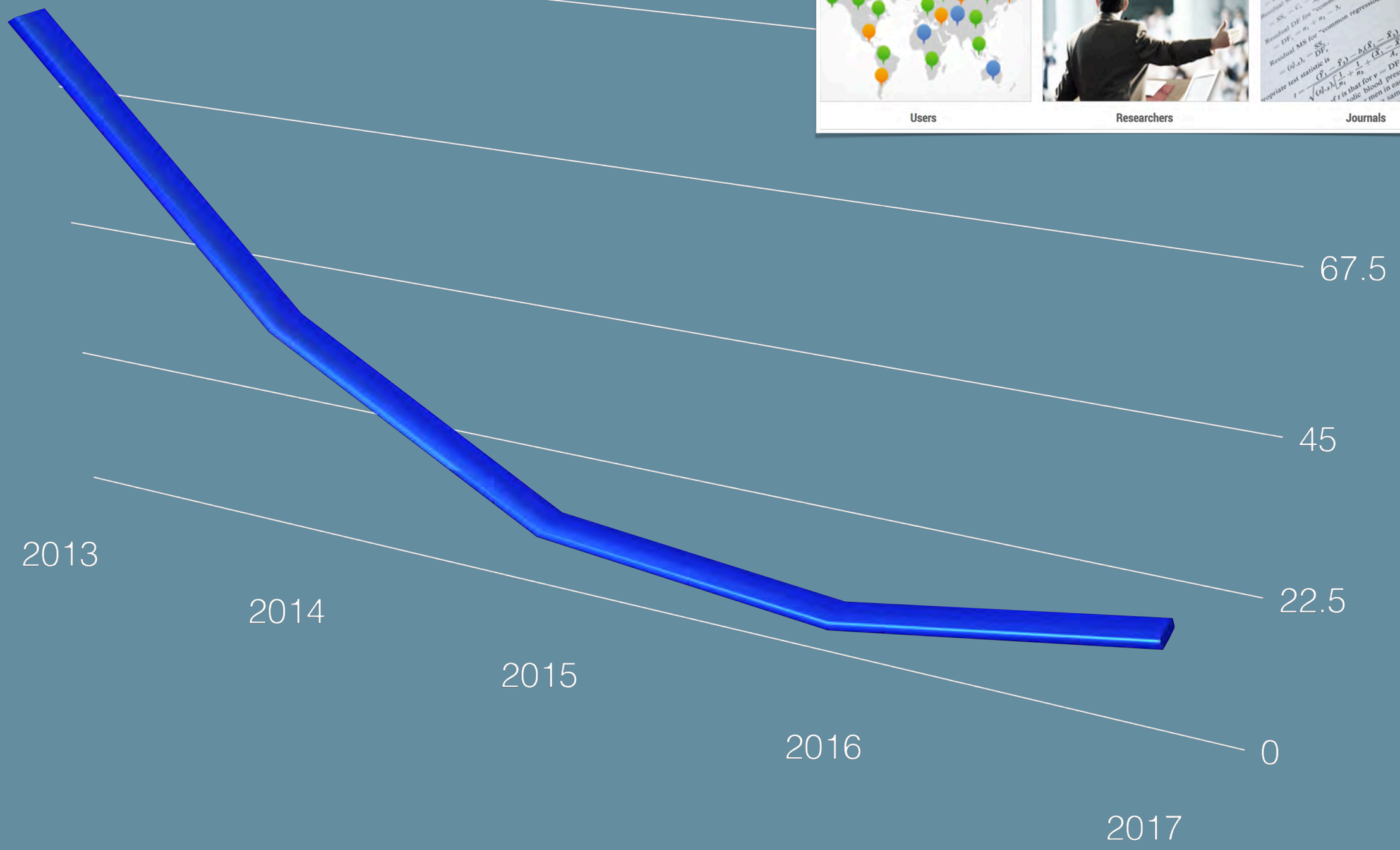
Users

Researchers

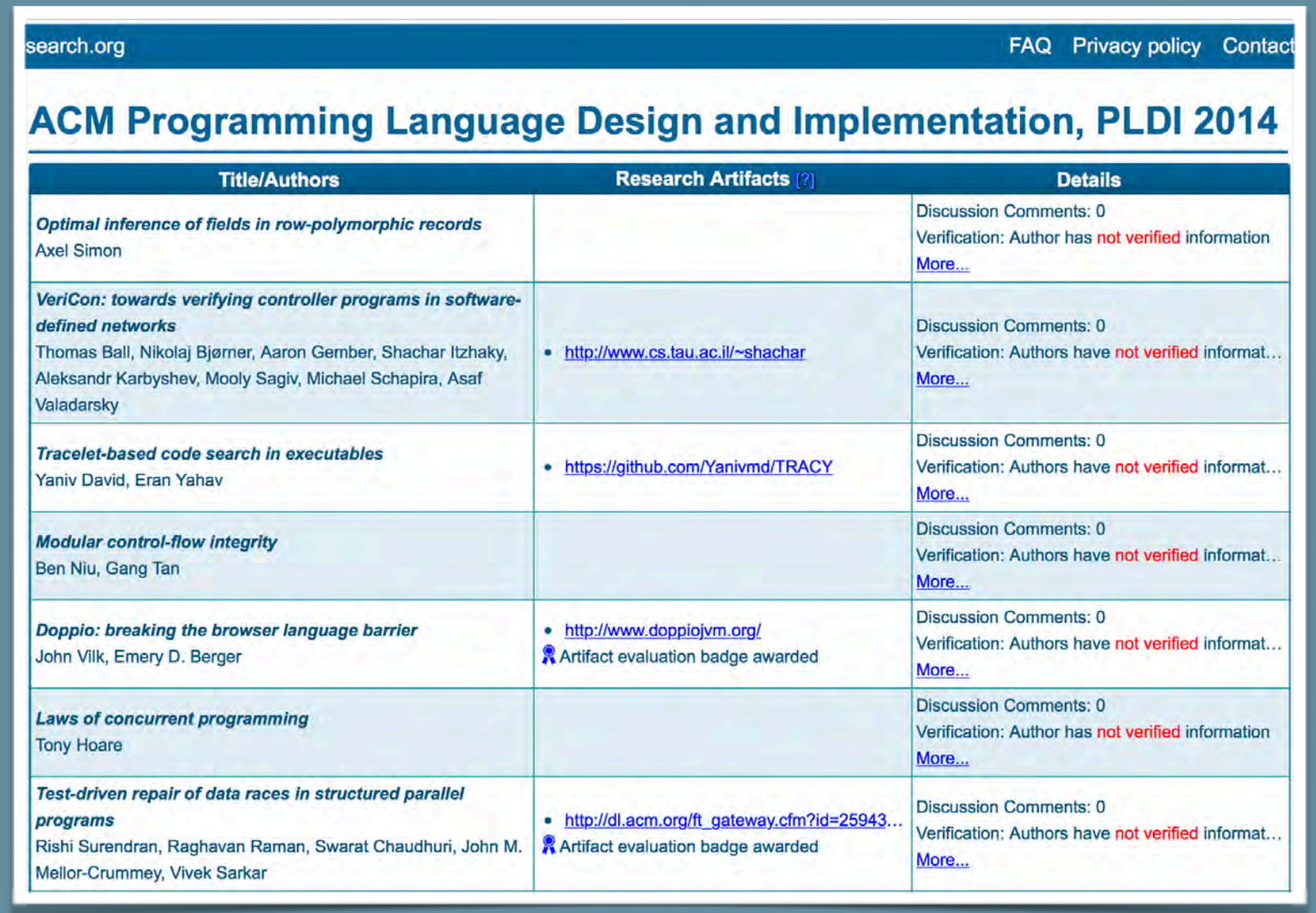


Journals


Search
Register | Sign In
runmycode
Home
Our goals
Researchers
Journal Editors
FAQ
Our partners
News
RunMyCode enables scientists
to openly share the code and data that
underlie their research publications
This service is based on the innovative concept of a companion
website associated with a scientific publication.
Share
Your email
Create a password
SIGN UP FOR RUNMYCODE
Users
Researchers
Journals


67.5
45
22.5
0
2013
2014
2015
2016
2017

FindResearch.org

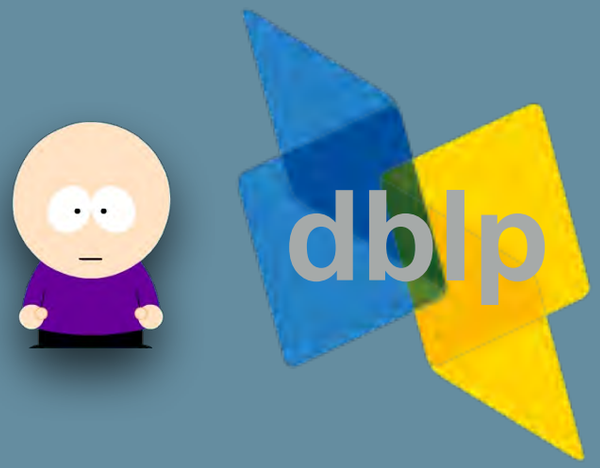


search.org FAQ Privacy policy Contact

ACM Programming Language Design and Implementation, PLDI 2014

Title/Authors	Research Artifacts [?]	Details
Optimal inference of fields in row-polymorphic records Axel Simon		Discussion Comments: 0 Verification: Author has not verified information More...
VeriCon: towards verifying controller programs in software-defined networks Thomas Ball, Nikolaj Bjørner, Aaron Gember, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky	• http://www.cs.tau.ac.il/~shachar	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Tracelet-based code search in executables Yaniv David, Eran Yahav	• https://github.com/Yanivmd/TRACY	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Modular control-flow integrity Ben Niu, Gang Tan		Discussion Comments: 0 Verification: Authors have not verified informat… More...
Doppio: breaking the browser language barrier John Vilk, Emery D. Berger	• http://www.doppiojvm.org/ Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Laws of concurrent programming Tony Hoare		Discussion Comments: 0 Verification: Author has not verified information More...
Test-driven repair of data races in structured parallel programs Rishi Surendran, Raghavan Raman, Swarat Chaudhuri, John M. Mellor-Crummey, Vivek Sarkar	• http://dl.acm.org/ft_gateway.cfm?id=25943… Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...

1. Help the public find artifacts
2. Motivate researchers to share

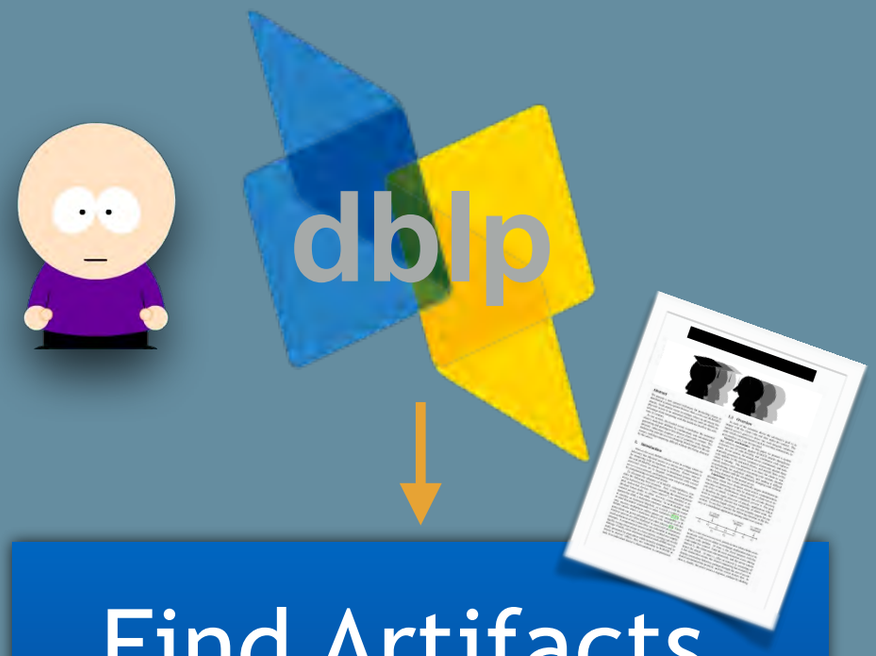


FindResearch.org

search.org FAQ Privacy policy Contact

ACM Programming Language Design and Implementation, PLDI 2014

Title/Authors	Research Artifacts [?]	Details
Optimal inference of fields in row-polymorphic records Axel Simon		Discussion Comments: 0 Verification: Author has not verified information More...
VeriCon: towards verifying controller programs in software-defined networks Thomas Ball, Nikolaj Bjørner, Aaron Gember, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky	• http://www.cs.tau.ac.il/~shachar	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Tracelet-based code search in executables Yaniv David, Eran Yahav	• https://github.com/Yanivmd/TRACY	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Modular control-flow integrity Ben Niu, Gang Tan		Discussion Comments: 0 Verification: Authors have not verified informat… More...
Doppio: breaking the browser language barrier John Vilk, Emery D. Berger	• http://www.doppiojvm.org/ Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Laws of concurrent programming Tony Hoare		Discussion Comments: 0 Verification: Author has not verified information More...
Test-driven repair of data races in structured parallel programs Rishi Surendran, Raghavan Raman, Swarat Chaudhuri, John M. Mellor-Crummey, Vivek Sarkar	• http://dl.acm.org/ft_gateway.cfm?id=25943… Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...



Find Artifacts
Emails, Grants

FindResearch.org

search.org FAQ Privacy policy Contact

ACM Programming Language Design and Implementation, PLDI 2014

Title/Authors	Research Artifacts [?]	Details
Optimal inference of fields in row-polymorphic records Axel Simon		Discussion Comments: 0 Verification: Author has not verified information More...
VeriCon: towards verifying controller programs in software-defined networks Thomas Ball, Nikolaj Bjørner, Aaron Gember, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky	• http://www.cs.tau.ac.il/~shachar	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Tracelet-based code search in executables Yaniv David, Eran Yahav	• https://github.com/Yanivmd/TRACY	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Modular control-flow integrity Ben Niu, Gang Tan		Discussion Comments: 0 Verification: Authors have not verified informat… More...
Doppio: breaking the browser language barrier John Vilk, Emery D. Berger	• http://www.doppiojvm.org/ Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Laws of concurrent programming Tony Hoare		Discussion Comments: 0 Verification: Author has not verified information More...
Test-driven repair of data races in structured parallel programs Rishi Surendran, Raghavan Raman, Swarat Chaudhuri, John M. Mellor-Crummey, Vivek Sarkar	• http://dl.acm.org/ft_gateway.cfm?id=25943… Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...

FindResearch.org

search.org FAQ Privacy policy Contact

ACM Programming Language Design and Implementation, PLDI 2014

Title/Authors	Research Artifacts [?]	Details
Optimal inference of fields in row-polymorphic records Axel Simon		Discussion Comments: 0 Verification: Author has not verified information More...
VeriCon: towards verifying controller programs in software-defined networks Thomas Ball, Nikolaj Bjørner, Aaron Gember, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky	• http://www.cs.tau.ac.il/~shachar	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Tracelet-based code search in executables Yaniv David, Eran Yahav	• https://github.com/Yanivmd/TRACY	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Modular control-flow integrity Ben Niu, Gang Tan		Discussion Comments: 0 Verification: Authors have not verified informat… More...
Doppio: breaking the browser language barrier John Vilk, Emery D. Berger	• http://www.doppiojvm.org/ Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Laws of concurrent programming Tony Hoare		Discussion Comments: 0 Verification: Author has not verified information More...
Test-driven repair of data races in structured parallel programs Rishi Surendran, Raghavan Raman, Swarat Chaudhuri, John M. Mellor-Crummey, Vivek Sarkar	• http://dl.acm.org/ft_gateway.cfm?id=25943… Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...

FindResearch.org

search.org FAQ Privacy policy Contact

ACM Programming Language Design and Implementation, PLDI 2014

Title/Authors	Research Artifacts [?]	Details
Optimal inference of fields in row-polymorphic records Axel Simon		Discussion Comments: 0 Verification: Author has not verified information More...
VeriCon: towards verifying controller programs in software-defined networks Thomas Ball, Nikolaj Bjørner, Aaron Gember, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky	• http://www.cs.tau.ac.il/~shachar	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Tracelet-based code search in executables Yaniv David, Eran Yahav	• https://github.com/Yanivmd/TRACY	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Modular control-flow integrity Ben Niu, Gang Tan		Discussion Comments: 0 Verification: Authors have not verified informat… More...
Doppio: breaking the browser language barrier John Vilk, Emery D. Berger	• http://www.doppiojvm.org/ Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Laws of concurrent programming Tony Hoare		Discussion Comments: 0 Verification: Author has not verified information More...
Test-driven repair of data races in structured parallel programs Rishi Surendran, Raghavan Raman, Swarat Chaudhuri, John M. Mellor-Crummey, Vivek Sarkar	• http://dl.acm.org/ft_gateway.cfm?id=25943… Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...

Find Artifacts
Emails, Grants

Verify
Information

Publish

FindResearch.org

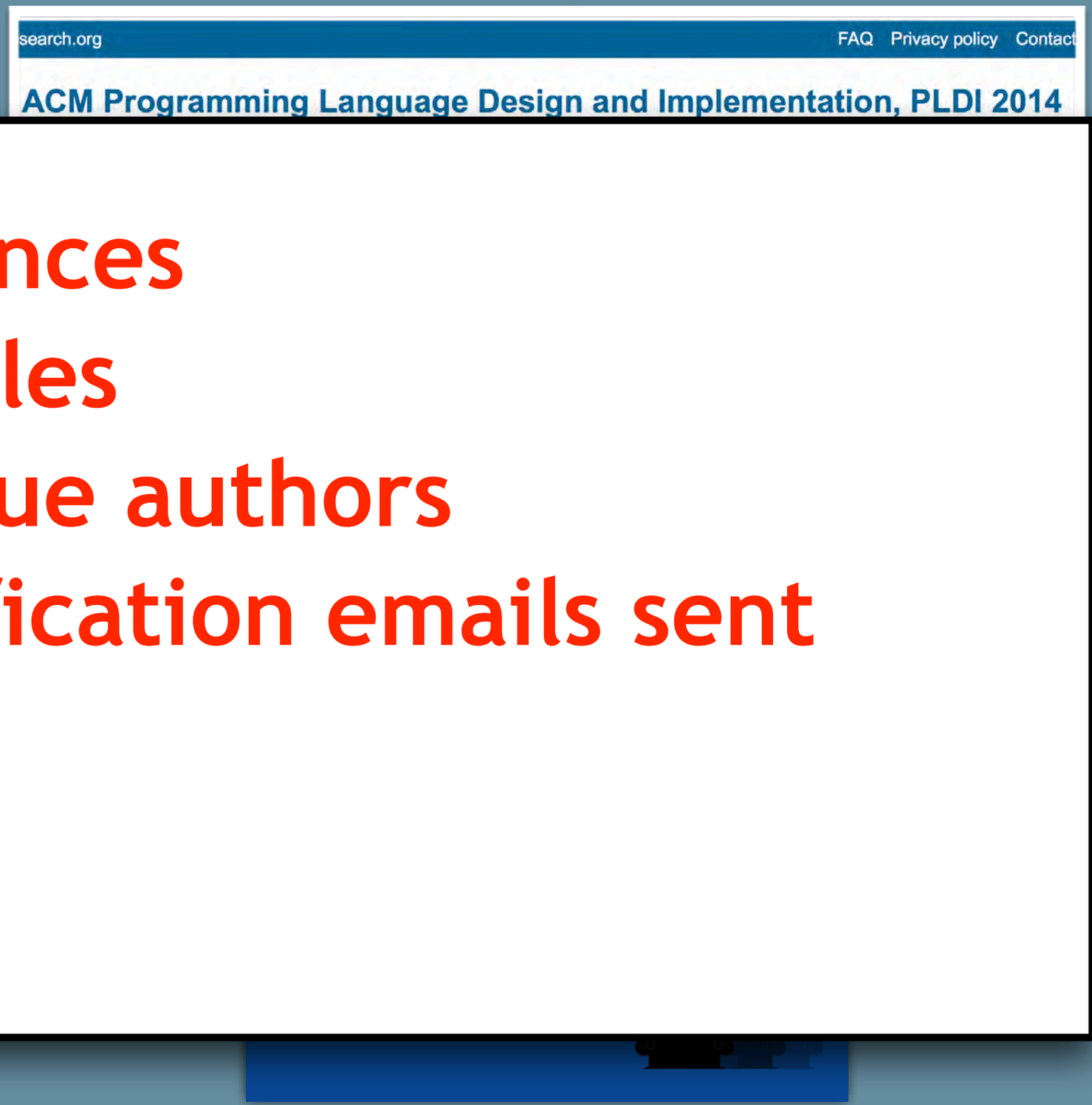
search.org FAQ Privacy policy Contact

ACM Programming Language Design and Implementation, PLDI 2014

Title/Authors	Research Artifacts [?]	Details
Optimal inference of fields in row-polymorphic records Axel Simon		Discussion Comments: 0 Verification: Author has not verified information More...
VeriCon: towards verifying controller programs in software-defined networks Thomas Ball, Nikolaj Bjørner, Aaron Gember, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky	• http://www.cs.tau.ac.il/~shachar	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Tracelet-based code search in executables Yaniv David, Eran Yahav	• https://github.com/Yanivmd/TRACY	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Modular control-flow integrity Ben Niu, Gang Tan		Discussion Comments: 0 Verification: Authors have not verified informat… More...
Doppio: breaking the browser language barrier John Vilk, Emery D. Berger	• http://www.doppiojvm.org/ Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...
Laws of concurrent programming Tony Hoare		Discussion Comments: 0 Verification: Author has not verified information More...
Test-driven repair of data races in structured parallel programs Rishi Surendran, Raghavan Raman, Swarat Chaudhuri, John M. Mellor-Crummey, Vivek Sarkar	• http://dl.acm.org/ft_gateway.cfm?id=25943… Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat… More...

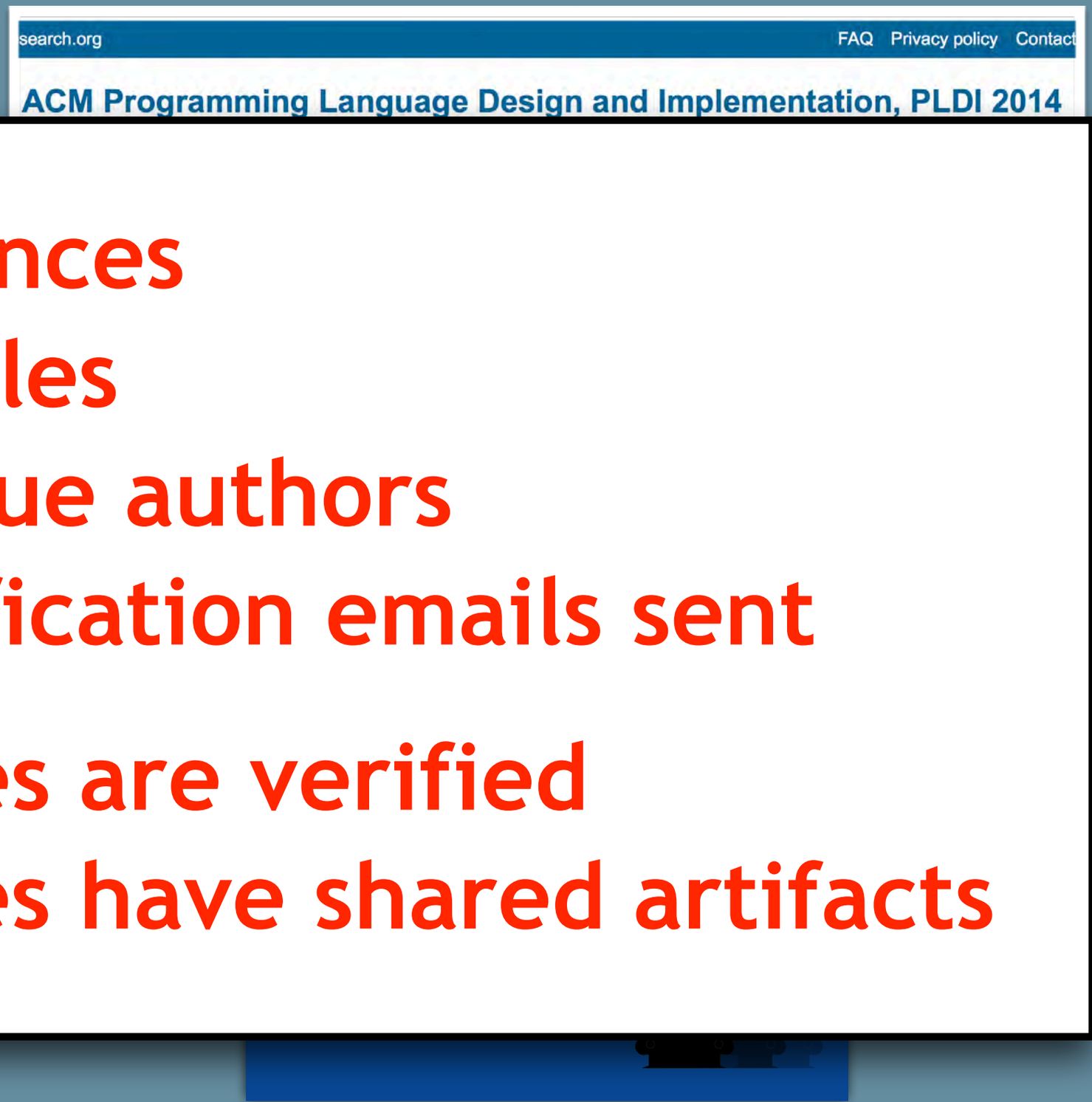
Discuss!

FindResearch.org



- 120 conferences
- 12,000 articles
- 47,000 unique authors
- 43,000 verification emails sent

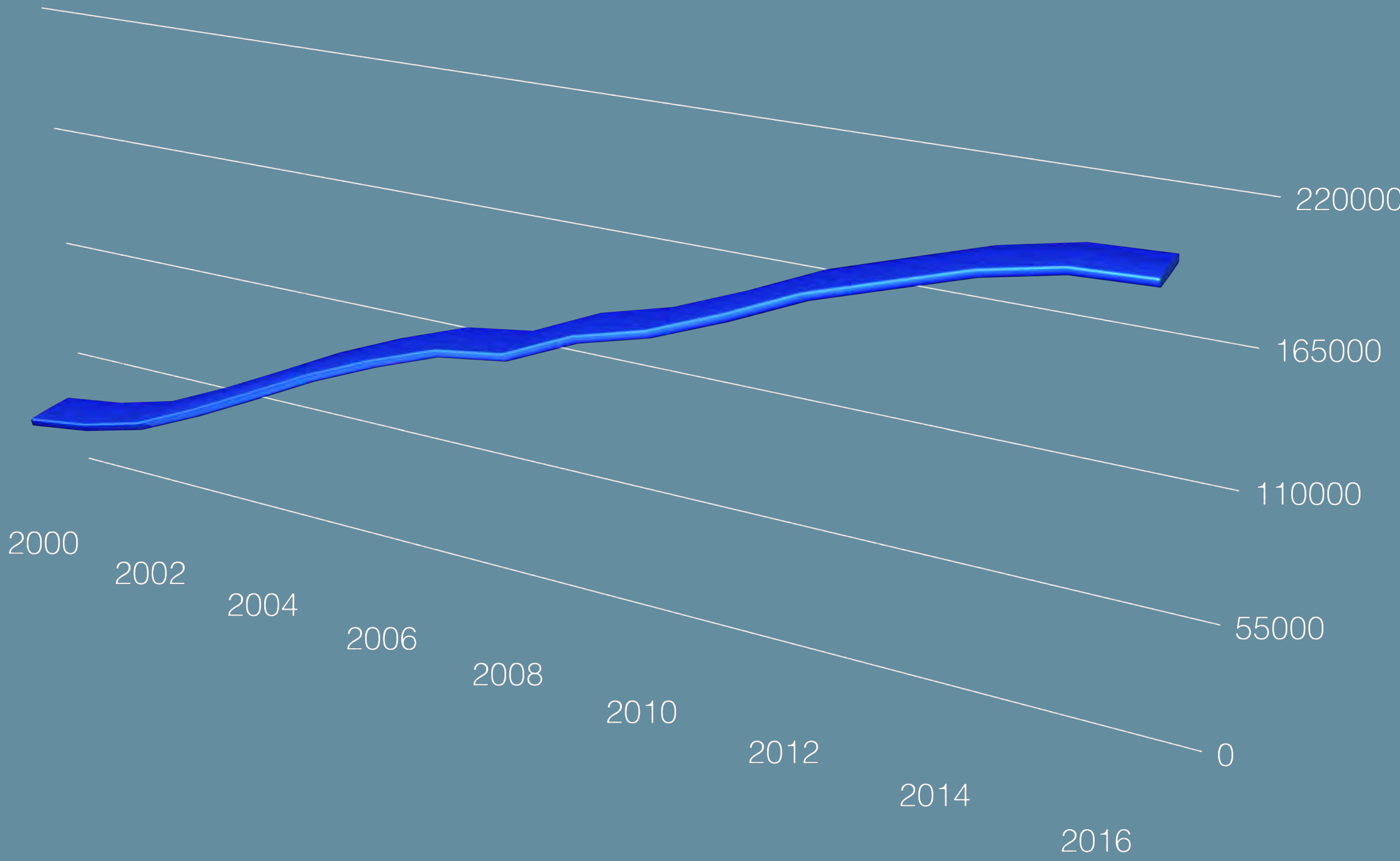
FindResearch.org



- 120 conferences
- 12,000 articles
- 47,000 unique authors
- 43,000 verification emails sent

- 9% of articles are verified
- 6% of articles have shared artifacts

Papers/Year (Dblp)

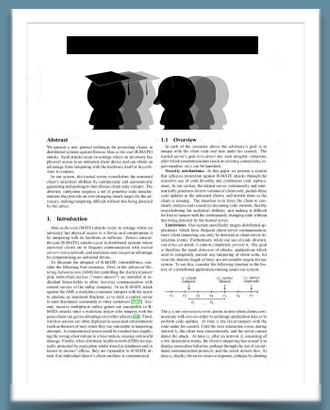


6th Law of Artifact Sharing

Even if you built it,
they still won't come.

Sharing Proposal
— #2 —

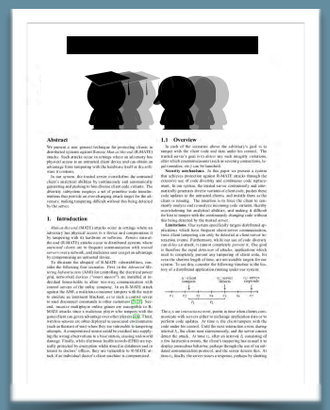
Checklists



Does it work?
(Repeatability)

ARTIFACT

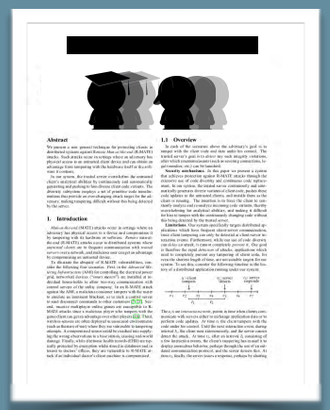
Why do you
want my code?



Will the code help me understand the paper?

ARTIFACT

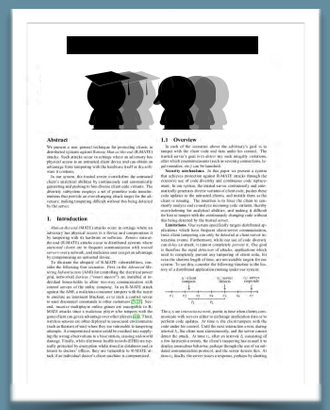
Why do you want my code?



Can I build on it?
(Benefaction)

ARTIFACT

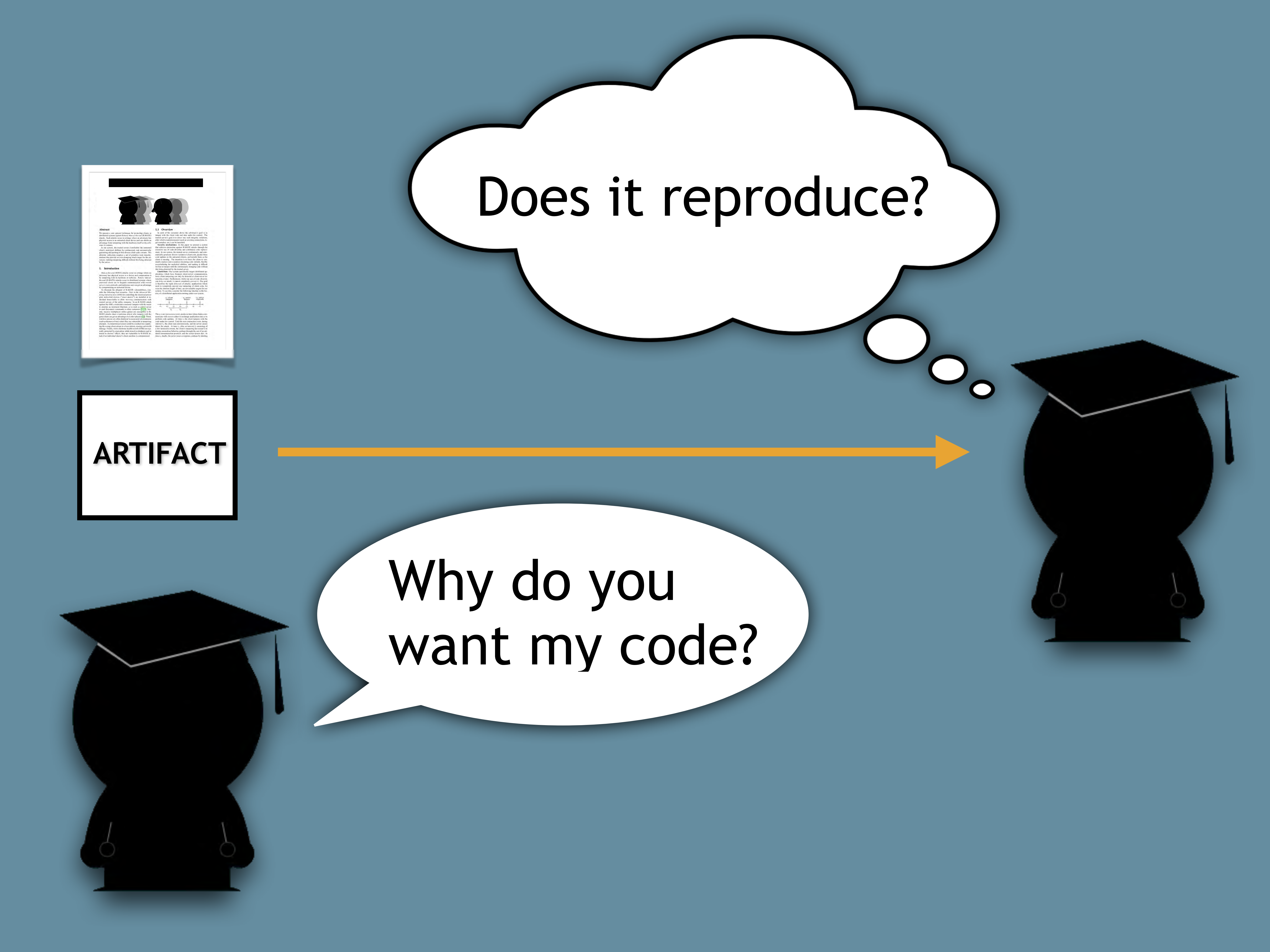
Why do you
want my code?

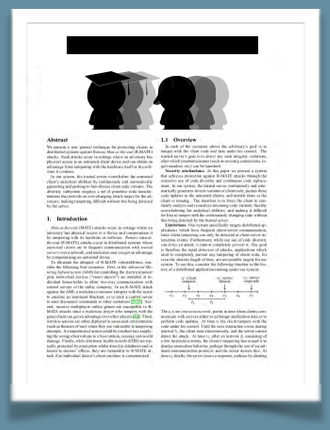


How does it compare to my work?

ARTIFACT

Why do you want my code?

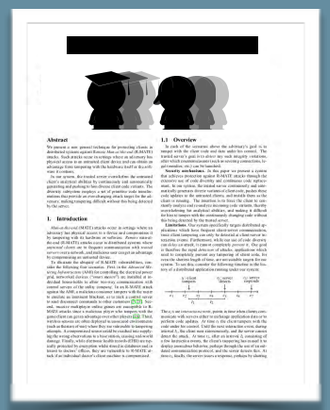

Does it reproduce?
ARTIFACT
Why do you want my code?



ARTIFACT

It’s on GitHub!
I’m done!

☑ Share everything



Scripts to run Experiments

README

Libraries

Makefile

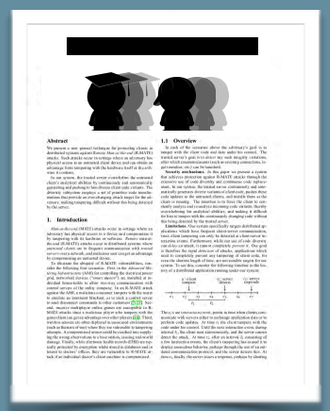
ARTIFACT

Sources

COQ Proof

Data Sets

☑ Share everything



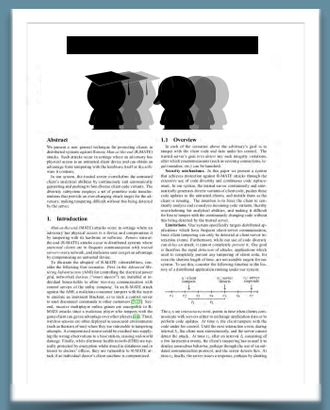
ARTIFACT

Libraries

Where's abclib.so?

Just **apt-get** it!
Works for me!

☑ Ensure *longevity*:
include all external code



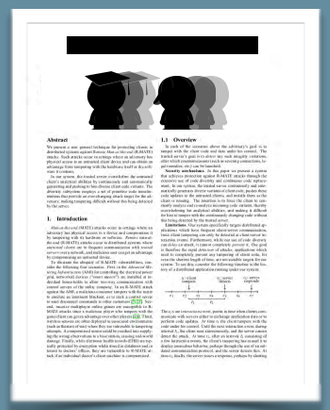
README

ARTIFACT

Which gcc version???

Uhm, 4.2 or higher?

☑ Document software you can’t include



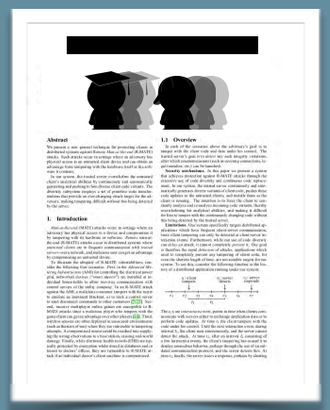
Paper
⇔
artifact?

ARTIFACT

ARTIFACT
V1.2

Uhm, I think this one?

☑ Clearly link paper to artifact



FindResearch.org

Paper⇒github!

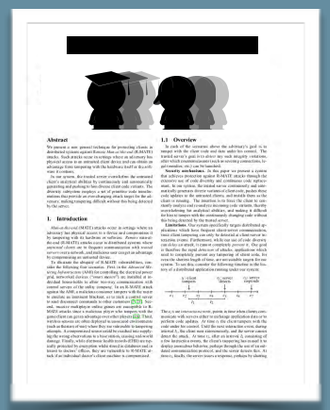
Paper
⇔
artifact?

ARTIFACT

ARTIFACT
V1.2

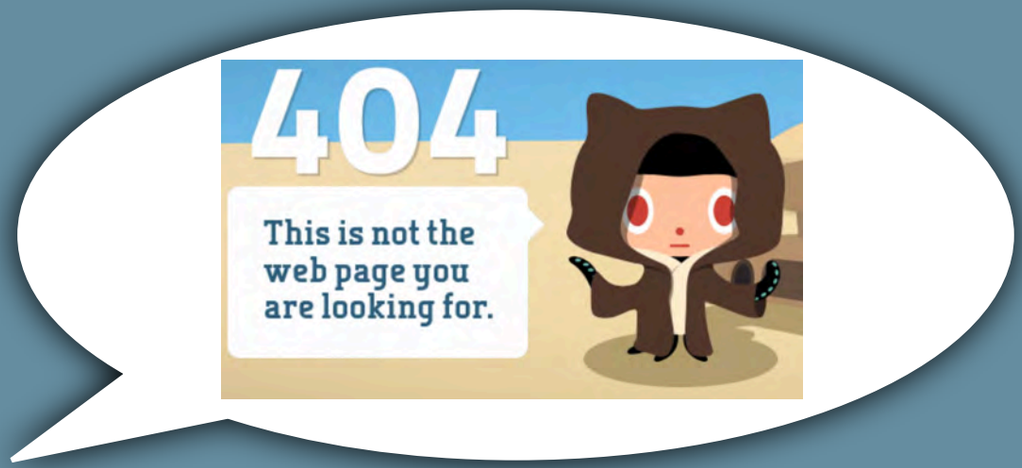
Uhm, I think this one?

☑ Clearly link paper to artifact

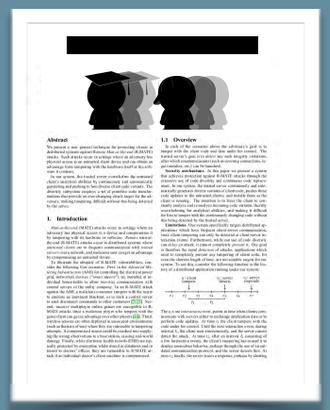


Where is your artifact?

ARTIFACT



☑ Ensure availability: find permanent storage

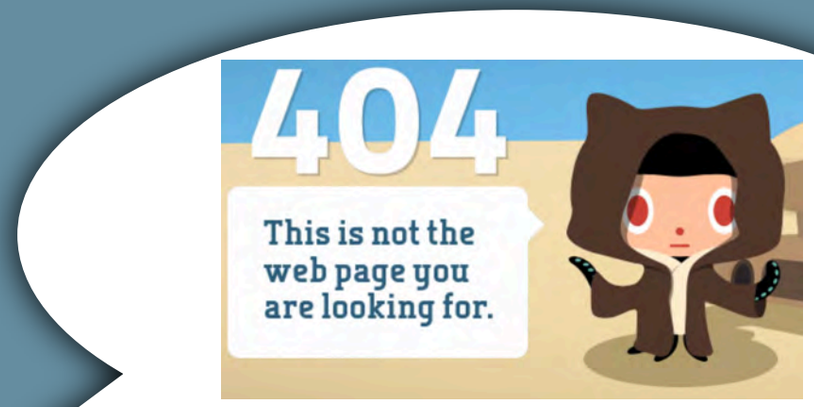


ARTIFACT

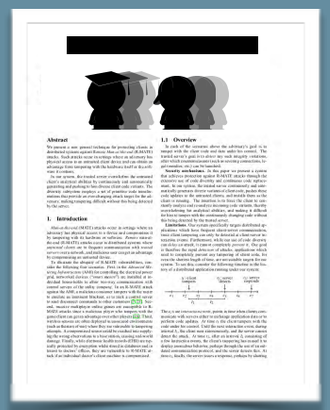
FindResearch.org

3% of verified papers with shared artifacts have broken links

Where is your artifact?



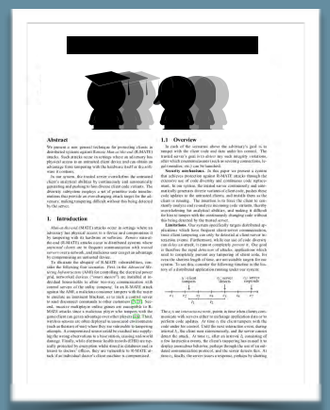
☑ Ensure availability: find permanent storage



Can you help me?

ARTIFACT

- [x] Use permanent email addresses



ARTIFACT

FindResearch.org

- 9% of emails bounced
- many articles without any email address

Can you help me?

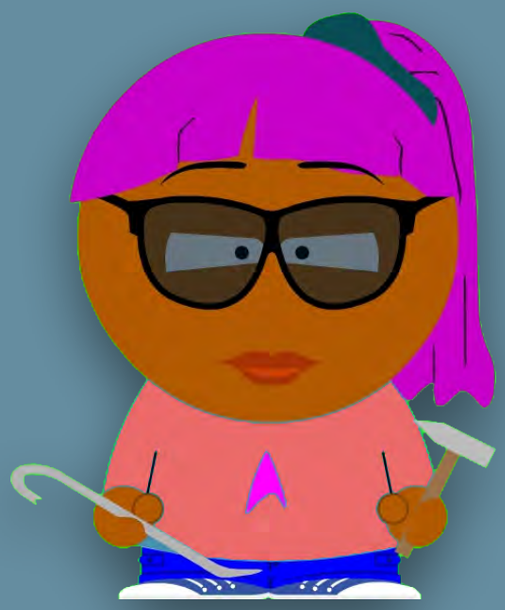
☑ Use permanent email addresses

5th Law of Artifact Sharing

To ensure repeatability of your results by others, you must

1. share everything
2. assume nothing
3. remain reachable

Sharing Proposal
— #3 —



Tool Support

Executable Paper 1

Abstract

We present a new general technique for protecting clients in distributed systems against *Remote Man-at-the-end* (R-MATE) attacks. Such attacks occur in settings where an adversary has physical access to an untrusted client device and can obtain an advantage from tampering with the hardware itself or the software it contains.

In our system, the trusted server overwhelms the untrusted client's analytical abilities by continuously and automatically generating and pushing to him diverse client code variants. The diversity subsystem employs a set of primitive code transformations that provide an ever-changing attack target for the adversary, making tampering difficult without this being detected by the server.

1. Introduction

Man-at-the-end (MATE) attacks occur in settings where an adversary has physical access to a device and compromises it by tampering with its hardware or software. *Remote* man-at-the-end (R-MATE) attacks occur in distributed systems where *untrusted clients* are in frequent communication with *trusted servers* over a network, and malicious user can get an advantage by compromising an untrusted device.

To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, in the *Advanced Metering Infrastructure* (AMI) for controlling the electrical power grid, networked devices ("*smart meters*") are installed at individual house-holds to allow two-way communication with control servers of the utility company. In an R-MATE attack against the AMI, a malicious consumer tampers with the meter to emulate an imminent blackout, or to trick a control server to send disconnect commands to other customers [7, 21]. Second, massive multiplayer online games are susceptible to R-MATE attacks since a malicious player who tampers with the game client can get an advantage over other players [16]. Third, wireless sensors are often deployed in unsecured environments (such as theaters of war) where they are vulnerable to tampering attempts. A compromised sensor could be coached into supplying the wrong observations to a base station, causing real-world damage. Finally, while electronic health records (EHR) are typically protected by encryption while stored in databases and in transit to doctors' offices, they are vulnerable to R-MATE attack if an individual doctor's client machine is compromised.

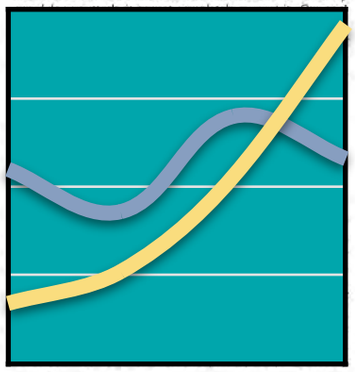
1.1 Overview

In each of the scenarios above the adversary's goal is to tamper with the client code and data under his control. The trusted server's goal is to *detect* any such integrity violations, after which countermeasures (such as severing connections, legal remedies, etc.) can be launched.

tryme

stantly analyze and re-analyze incoming code variants, thereby overwhelming his analytical abilities, and making it difficult for him to tamper with the continuously changing code without this being detected by the trusted server.

Limitations. Our system specifically targets distributed applications which have frequent client-server communication, since client tampering can only be detected at client-server interaction events. Furthermore, while our use of code diversity



is.ieis.tue.nl/staff/pvgorp/share

SHARE

Paper1.vm

Experiments

Code

Data

Paper2.vm

Executable Paper 1

Abstract

We present a new general technique for protecting clients in distributed systems against *Remote Man-at-the-end* (R-MATE) attacks. Such attacks occur in settings where an adversary has physical access to an untrusted client device and can obtain an advantage from tampering with the hardware itself or the software it contains.

In our system, the trusted server overwhelms the untrusted client's analytical abilities by continuously and automatically generating and pushing to him diverse client code variants. The diversity subsystem employs a set of primitive code transformations that provide an ever-changing attack target for the adversary, making tampering difficult without this being detected by the server.

1. Introduction

Man-at-the-end (MATE) attacks occur in settings where an adversary has physical access to a device and compromises it by tampering with its hardware or software. *Remote* man-at-the-end (R-MATE) attacks occur in distributed systems where *untrusted clients* are in frequent communication with *trusted servers* over a network, and malicious user can get an advantage by compromising an untrusted device.

To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, in the *Advanced Metering Infrastructure* (AMI) for controlling the electrical power grid, networked devices ("*smart meters*") are installed at individual house-holds to allow two-way communication with control servers of the utility company. In an R-MATE attack against the AMI, a malicious consumer tampers with the meter to emulate an imminent blackout, or to trick a control server to send disconnect commands to other customers [7, 21]. Second, massive multiplayer online games are susceptible to R-MATE attacks since a malicious player who tampers with the game client can get an advantage over other players [16]. Third, wireless sensors are often deployed in unsecured environments (such as theaters of war) where they are vulnerable to tampering attempts. A compromised sensor could be coached into supplying the wrong observations to a base station, causing real-world damage. Finally, while electronic health records (EHR) are typically protected by encryption while stored in databases and in transit to doctors' offices, they are vulnerable to R-MATE attack if an individual doctor's client machine is compromised.

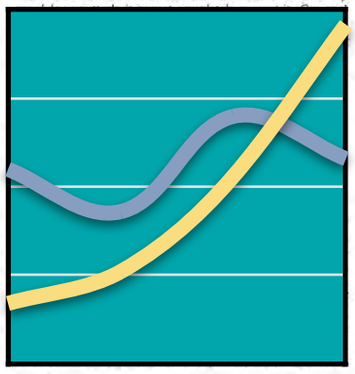
1.1 Overview

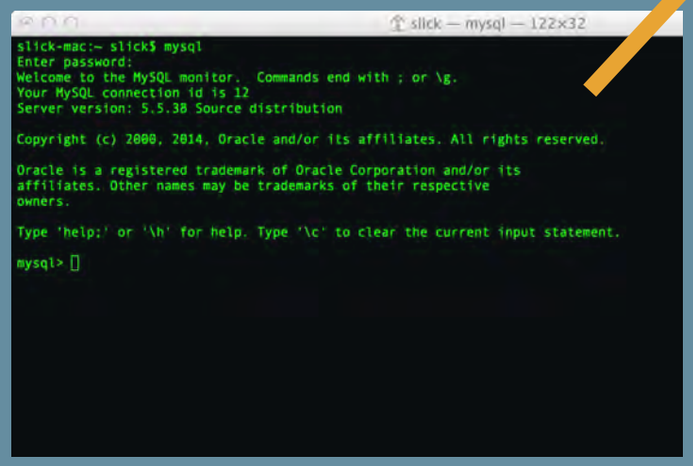
In each of the scenarios above the adversary's goal is to tamper with the client code and data under his control. The trusted server's goal is to *detect* any such integrity violations, after which countermeasures (such as severing connections, legal remedies, etc.) can be launched.

tryme

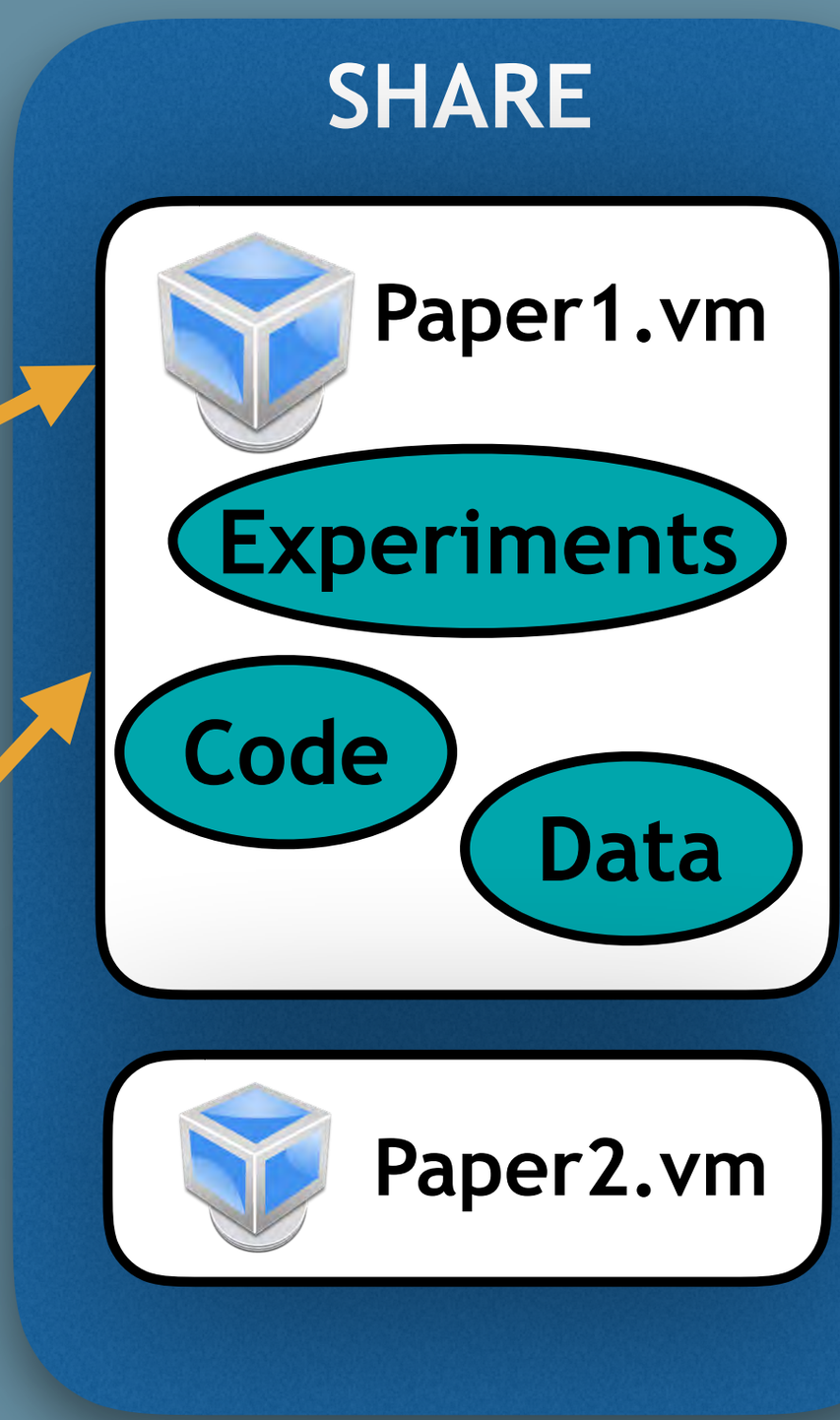
stantly analyze and re-analyze incoming code variants, thereby overwhelming his analytical abilities, and making it difficult for him to tamper with the continuously changing code without this being detected by the trusted server.

Limitations. Our system specifically targets distributed applications which have frequent client-server communication, since client tampering can only be detected at client-server interaction events. Furthermore, while our use of code diversity





is.ieis.tue.nl/staff/pvgorp/share



Executable Paper 1

Abstract

We present a new general technique for protecting clients in distributed systems against *Remote Man-at-the-end* (R-MATE) attacks. Such attacks occur in settings where an adversary has physical access to an untrusted client device and can obtain an advantage from tampering with the hardware itself or the software it contains.

In our system, the trusted server overwhelms the untrusted client's analytical abilities by continuously and automatically generating and pushing to him diverse client code variants. The diversity subsystem employs a set of primitive code transformations that provide an ever-changing attack target for the adversary, making tampering difficult without this being detected by the server.

1. Introduction

Man-at-the-end (MATE) attacks occur in settings where an adversary has physical access to a device and compromises it by tampering with its hardware or software. *Remote* man-at-the-end (R-MATE) attacks occur in distributed systems where *untrusted clients* are in frequent communication with *trusted servers* over a network, and malicious user can obtain an advantage by compromising an untrusted device.

To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, the *Advanced Metering Infrastructure* (AMI) for controlling the electrical power grid, networked devices ("*smart meters*") are installed at individual house-holds to allow two-way communication with control servers of the utility company. In an R-MATE attack against the AMI, a malicious consumer tampers with the meter to emulate an imminent blackout, thereby trick a control server to send disconnect commands to other customers [7, 21]. Second, massive multiplayer online games are susceptible to R-MATE attacks since a malicious player who tampers with the game client can get an advantage over other players [16]. Third, wireless sensors are often deployed in unsecured environments (such as theaters of war) where they are vulnerable to tampering attempts. A compromised sensor could be coached into supplying the wrong observations to a base station causing real-world damage. Finally, while electronic health records (EHR) are typically protected by encryption while stored in databases and in transit to doctors' offices, they are vulnerable to R-MATE attack if an individual doctor's client machine is compromised.

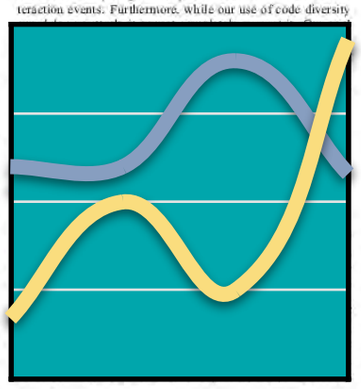
1.1 Overview

In each of the scenarios above the adversary's goal is to tamper with the client code and data under his control. The trusted server's goal is to *detect* any such integrity violations, after which countermeasures (such as severing connections, legal remedies, etc.) can be launched.

tryme

stantly analyze and test [illegible] code variants, thereby overwhelming the client's analytical abilities, and making it difficult for him to tamper with the continuously changing code without this being detected by the trusted server.

Limitations. Our system specifically targets distributed applications which have frequent client-server communication, since client tampering can only be detected at client-server interaction events. Furthermore, while our use of code diversity



```
slick — mysql — 122×32
slick-mac:~ slick$ mysql
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.38 Source distribution

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

is.ieis.tue.nl/staff/pvgorp/share

SHARE

Paper1.vm

Experiments

Code

Data

Paper2.vm

VisTrails

Workflow v1.0

www.vistrails.org

Data

Paper

Abstract

We present a new general technique for protecting clients in distributed systems against *Remote Man-at-the-end* (R-MATE) attacks. Such attacks occur in settings where an adversary has physical access to an untrusted client device and can obtain an advantage from tampering with the hardware itself or the software it contains.

In our system, the trusted server overwhelms the untrusted client's analytical abilities by continuously and automatically generating and pushing to him diverse client code variants. The diversity subsystem employs a set of primitive code transformations that provide an ever-changing attack target for the adversary, making tampering difficult without this being detected by the server.

1. Introduction

Man-at-the-end (MATE) attacks occur in settings where an adversary has physical access to a device and compromises it by tampering with its hardware or software. *Remote* man-at-the-end (R-MATE) attacks occur in distributed systems where *untrusted clients* are in frequent communication with *trusted servers* over a network, and malicious user can get an advantage by compromising an untrusted device.

To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, in the *Advanced Metering Infrastructure* (AMI) for controlling the electrical power grid, networked devices ("*smart meters*") are installed at individual house-holds to allow two-way communication with control servers of the utility company. In an R-MATE attack against the AMI, a malicious consumer tampers with the meter to emulate an imminent blackout, or to trick a control server to send disconnect commands to other customers [2, 23]. Second, massive multiplayer online games are susceptible to R-MATE attacks since a malicious player who tampers with the game client can get an advantage over other players [16]. Third, wireless sensors are often deployed in unsecured environments (such as theaters of war) where they are vulnerable to tampering attempts. A compromised sensor could be coached into supplying the wrong observations to a base station, causing real-world damage. Finally, while electronic health records (EHR) are typically protected by encryption while stored in databases and in transit to doctors' offices, they are vulnerable to R-MATE attack if an individual doctor's client machine is compromised.

1.1 Overview

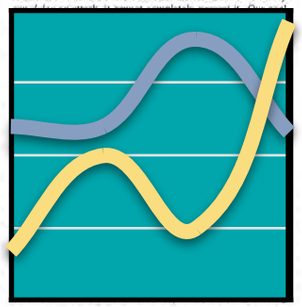
In each of the scenarios above the adversary's goal is to tamper with the client code and data under his control. The trusted server's goal is to *detect* any such integrity violations, after which countermeasures (such as severing connections, legal remedies, etc.) can be launched.

Security mechanisms. In this paper we present a system

tryme

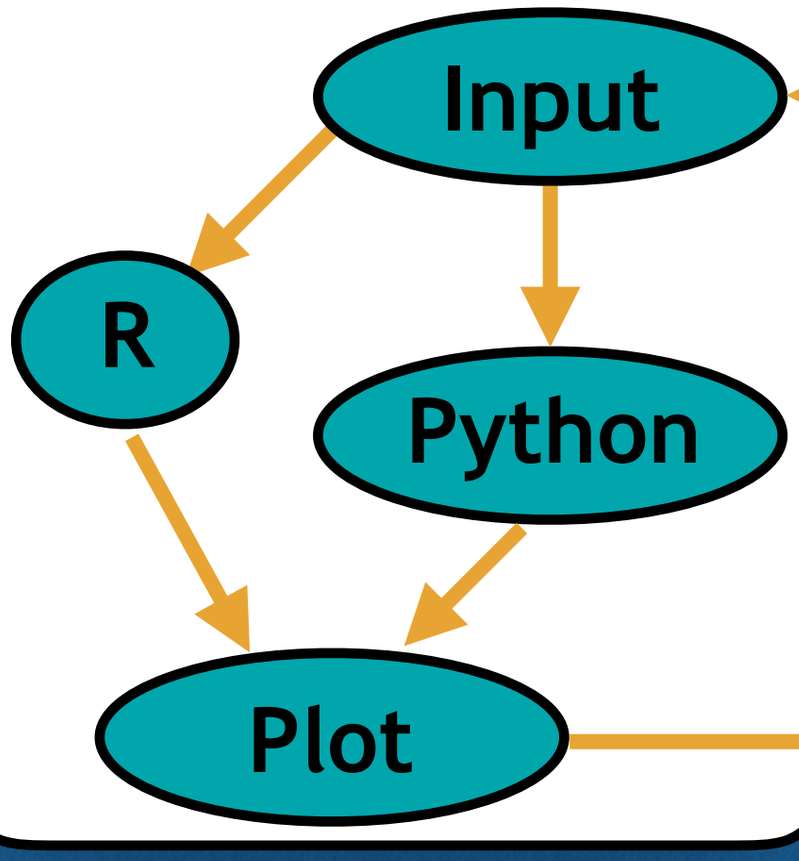
this being detected by the trusted server.

Limitations. Our system specifically targets distributed applications which have frequent client-server communication, since client tampering can only be detected at client-server interaction events. Furthermore, while our use of code diversity

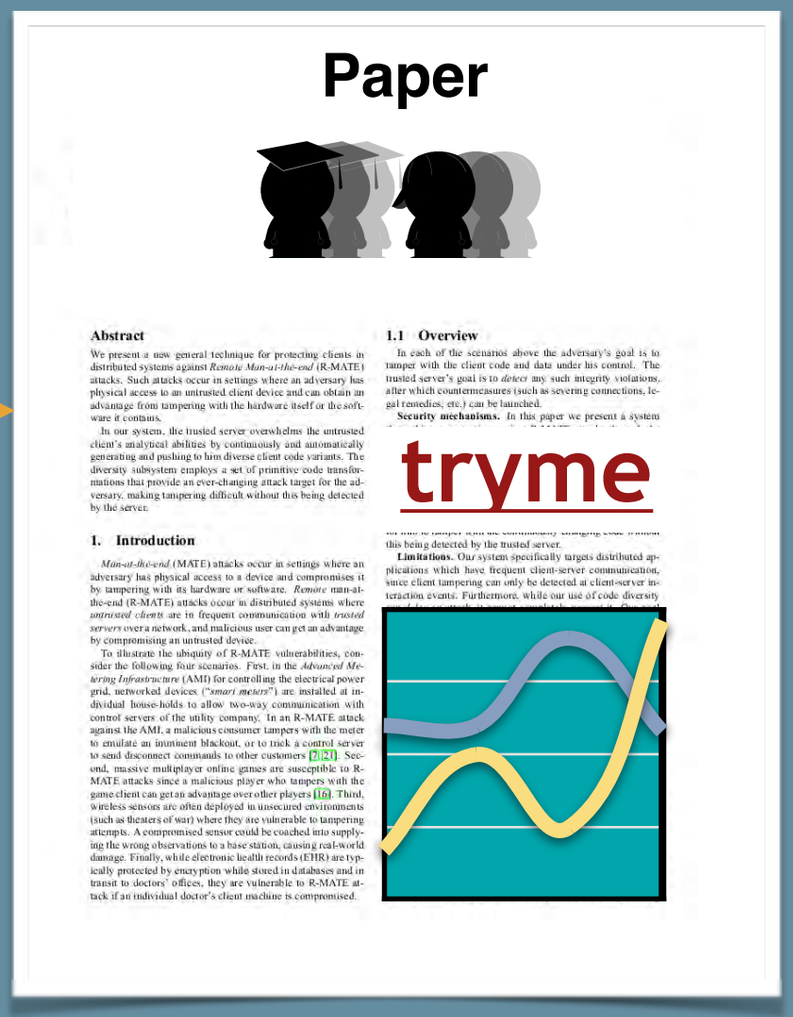


VisTrails

Workflow v1.0



www.vistrails.org



VisTrails

Workflow v1.1

Workflow v1.0

Input

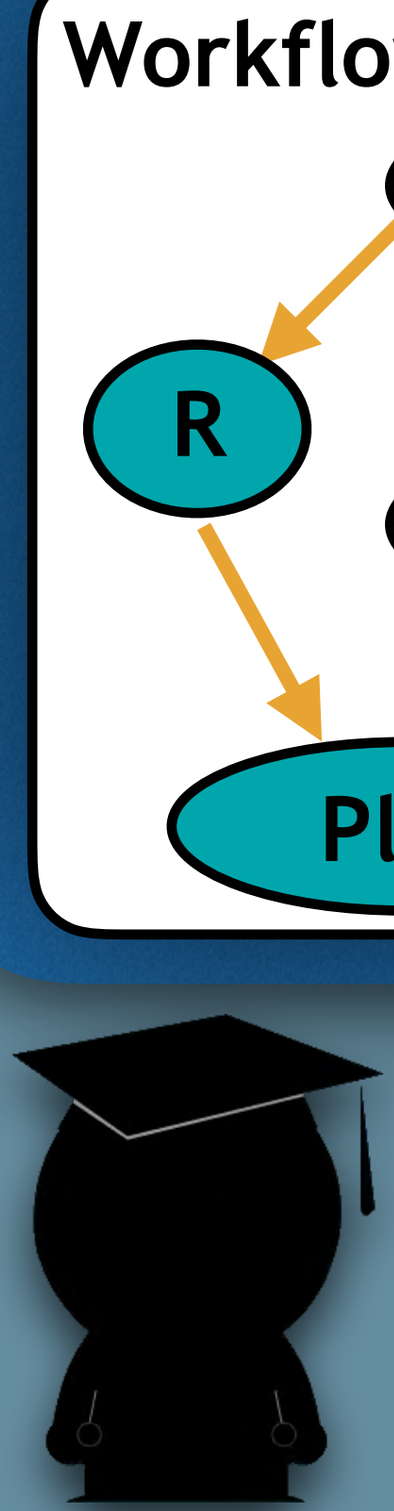
R

Python

Plot

Data

www.vistrails.org



Paper

Abstract

We present a new general technique for protecting clients in distributed systems against *Remote Man-at-the-end* (R-MATE) attacks. Such attacks occur in settings where an adversary has physical access to an untrusted client device and can obtain an advantage from tampering with the hardware itself or the software it contains.

In our system, the trusted server overwhelms the untrusted client's analytical abilities by continuously and automatically generating and pushing to him diverse client code variants. The diversity subsystem employs a set of primitive code transformations that provide an ever-changing attack target for the adversary, making tampering difficult without this being detected by the server.

1. Introduction

Man-at-the-end (MATE) attacks occur in settings where an adversary has physical access to a device and compromises it by tampering with its hardware or software. *Remote* man-at-the-end (R-MATE) attacks occur in distributed systems where *untrusted clients* are in frequent communication with *trusted servers* over a network, and malicious user can get an advantage by compromising an untrusted device.

To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, in the *Advanced Metering Infrastructure* (AMI) for controlling the electrical power grid, networked devices ("*smart meters*") are installed at individual house-holds to allow two-way communication with control servers of the utility company. In an R-MATE attack against the AMI, a malicious consumer tampers with the meter to emulate an imminent blackout, or to trick a control server to send disconnect commands to other customers [2, 21]. Second, massive multiplayer online games are susceptible to R-MATE attacks since a malicious player who tampers with the game client can get an advantage over other players [16]. Third, wireless sensors are often deployed in unsecured environments (such as theaters of war) where they are vulnerable to tampering attempts. A compromised sensor could be coached into supplying the wrong observations to a base station, causing real-world damage. Finally, while electronic health records (EHR) are typically protected by encryption while stored in databases and in transit to doctors' offices, they are vulnerable to R-MATE attack if an individual doctor's client machine is compromised.

1.1 Overview

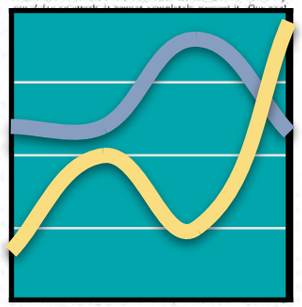
In each of the scenarios above the adversary's goal is to tamper with the client code and data under his control. The trusted server's goal is to *detect* any such integrity violations, after which countermeasures (such as severing connections, legal remedies, etc.) can be launched.

Security mechanisms. In this paper we present a system

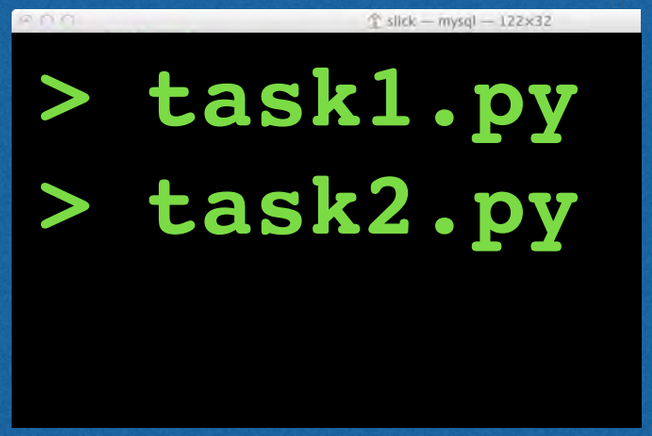
tryme

this being detected by the trusted server.

Limitations. Our system specifically targets distributed applications which have frequent client-server communication, since client tampering can only be detected at client-server interaction events. Furthermore, while our use of code diversity



ReproZip-Pack



TRACE

```
open()
exec()
```

www.reprozip.org

www.reprozip.org

ReproZip-Pack

```
> task1.py
> task2.py
```

TRACE

```
open()
exec()
```

task.zip

```
python
libc
task1
```

www.reprozip.org

ReproZip-Pack

```
> task1.py
> task2.py
```

TRACE

```
open()
exec()
```

task.zip

```
python
libc
task1
```

ReproZip-Unpack

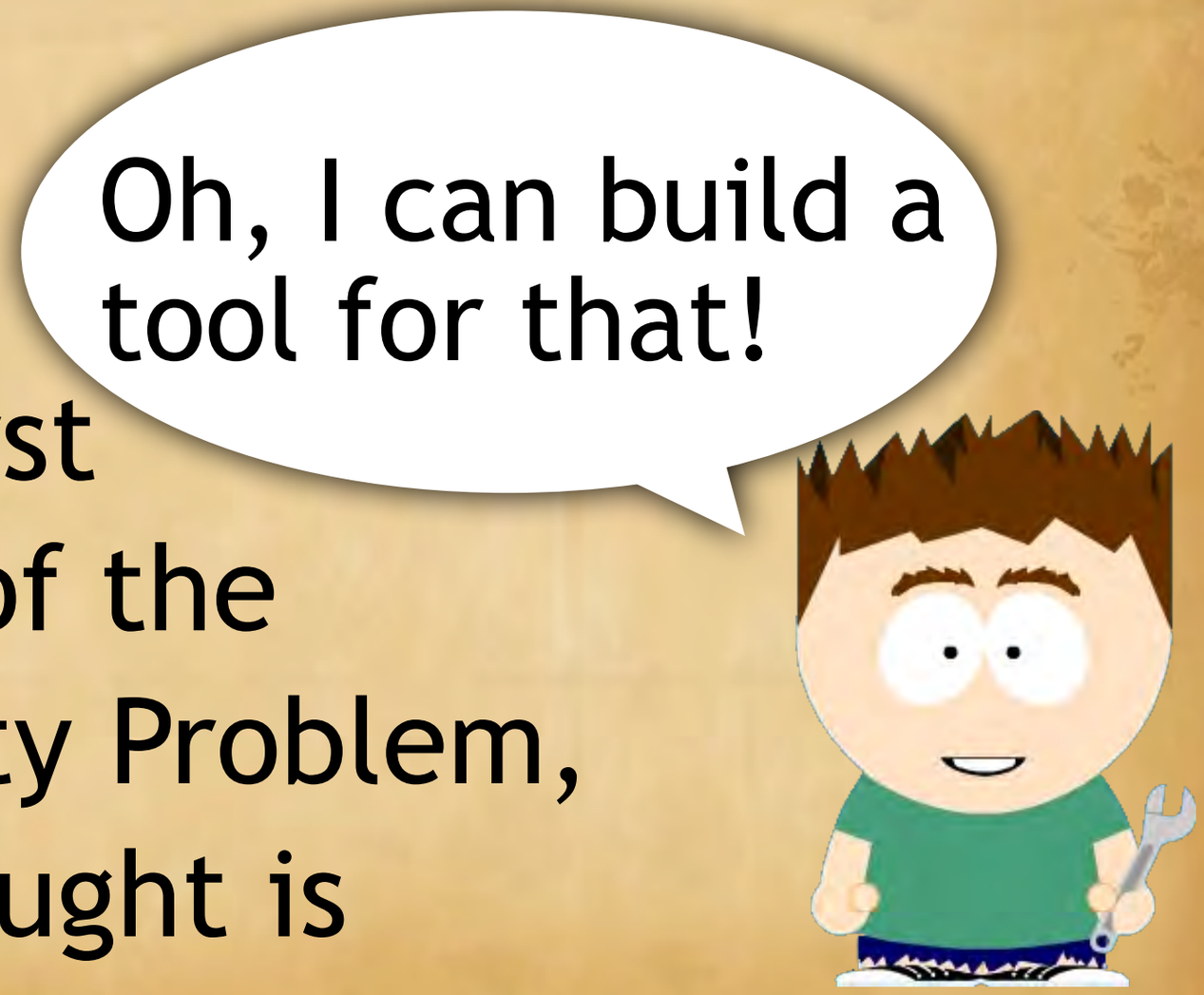
```
> run task1
> run task2
```

4th Law of Artifact Sharing

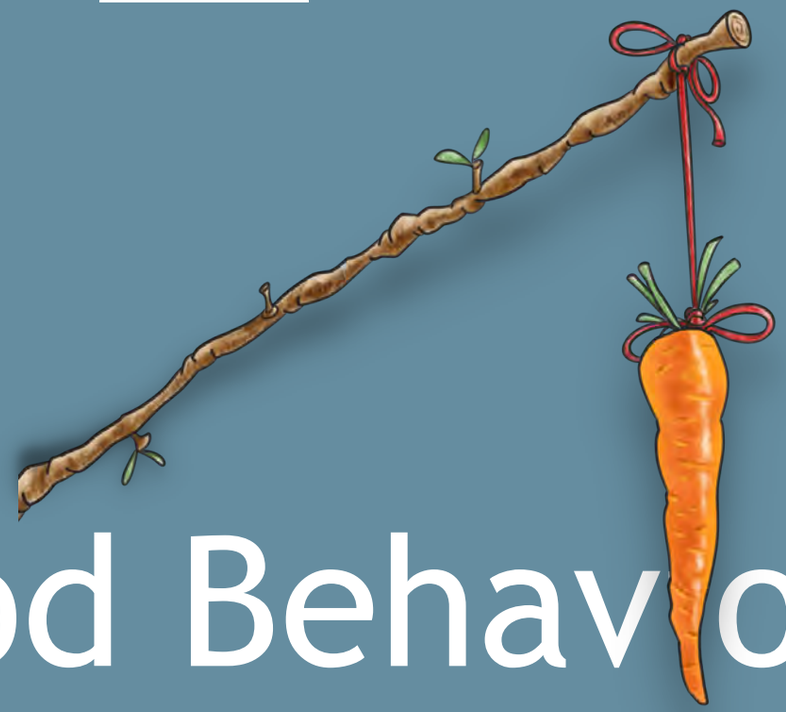
When a
Computer
Scientist is first
made aware of the
Reproducibility Problem,
their first thought is

4th Law of Artifact Sharing

When a Computer Scientist is first made aware of the Reproducibility Problem, their first thought is



Sharing Proposal
— #4 —

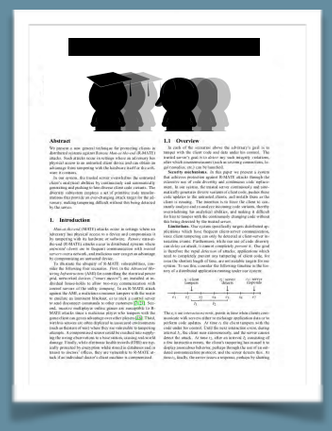


Rewarding Good Behavior

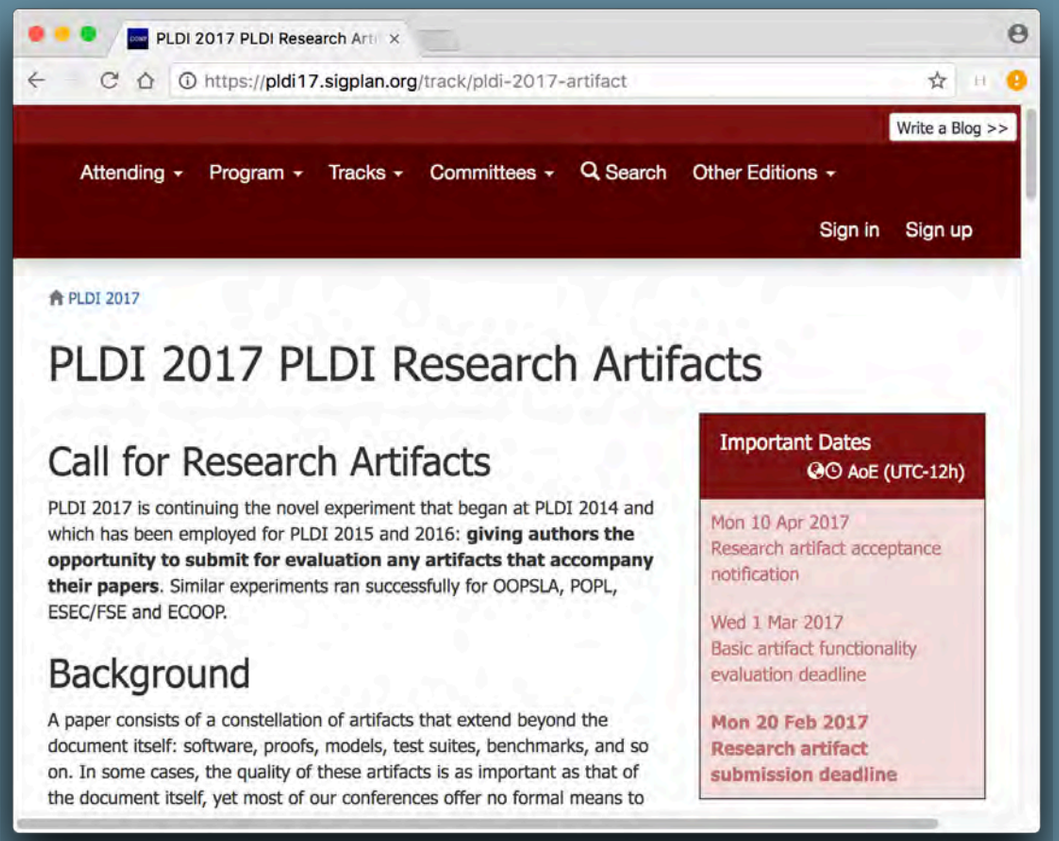
Sharing Proposal

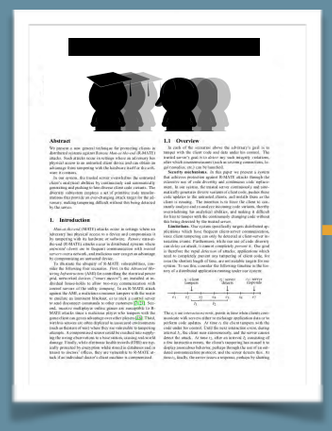
— #4 —

Rewarding Good Behavior



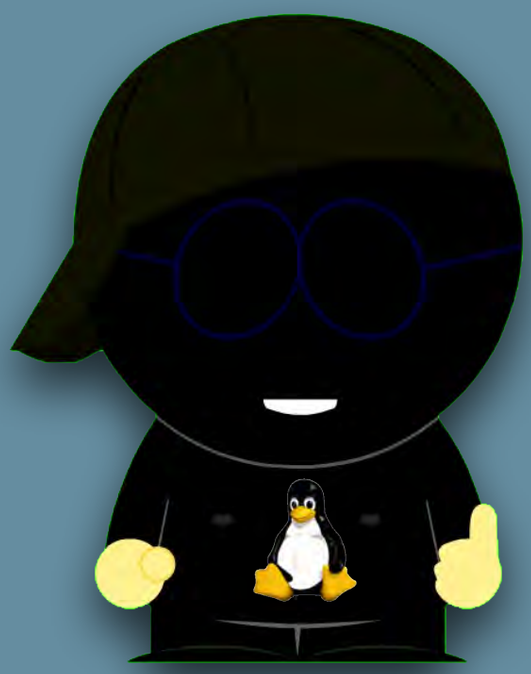
ARTIFACT


PLDI 2017 PLDI Research Arti
https://pldi17.sigplan.org/track/pldi-2017-artifact
Write a Blog >>
Attending
Program
Tracks
Committees
Search
Other Editions
Sign in
Sign up
PLDI 2017
PLDI 2017 PLDI Research Artifacts
Call for Research Artifacts
PLDI 2017 is continuing the novel experiment that began at PLDI 2014 and which has been employed for PLDI 2015 and 2016: giving authors the opportunity to submit for evaluation any artifacts that accompany their papers. Similar experiments ran successfully for OOPSLA, POPL, ESEC/FSE and ECOOP.
Background
A paper consists of a constellation of artifacts that extend beyond the document itself: software, proofs, models, test suites, benchmarks, and so on. In some cases, the quality of these artifacts is as important as that of the document itself, yet most of our conferences offer no formal means to
Important Dates
AoE (UTC-12h)
Mon 10 Apr 2017
Research artifact acceptance notification
Wed 1 Mar 2017
Basic artifact functionality evaluation deadline
Mon 20 Feb 2017
Research artifact submission deadline



Paper accepted?

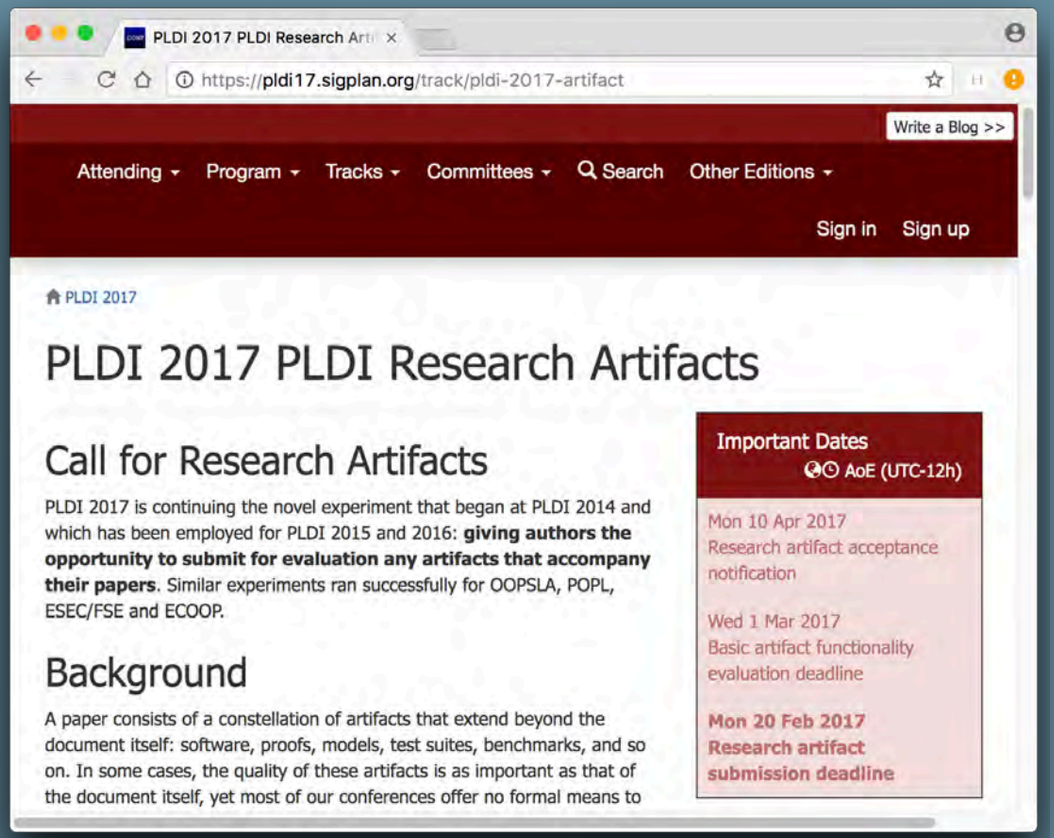
ARTIFACT



Paper accepted?

ARTIFACT

Artifact accepted?



OK!

ARTIFACT

Paper accepted?

Artifact accepted?

- Voluntary
- Does not affect accept/reject
- No expectation of sharing

OK!


Artifact
Consistent
Complete
Well Documented
Easy to Reuse
PLDI
AEC
Evaluated

ARTIFACT

Repeatability

ARTIFACT

ARTIFACT

Repeatability

Reproducibility

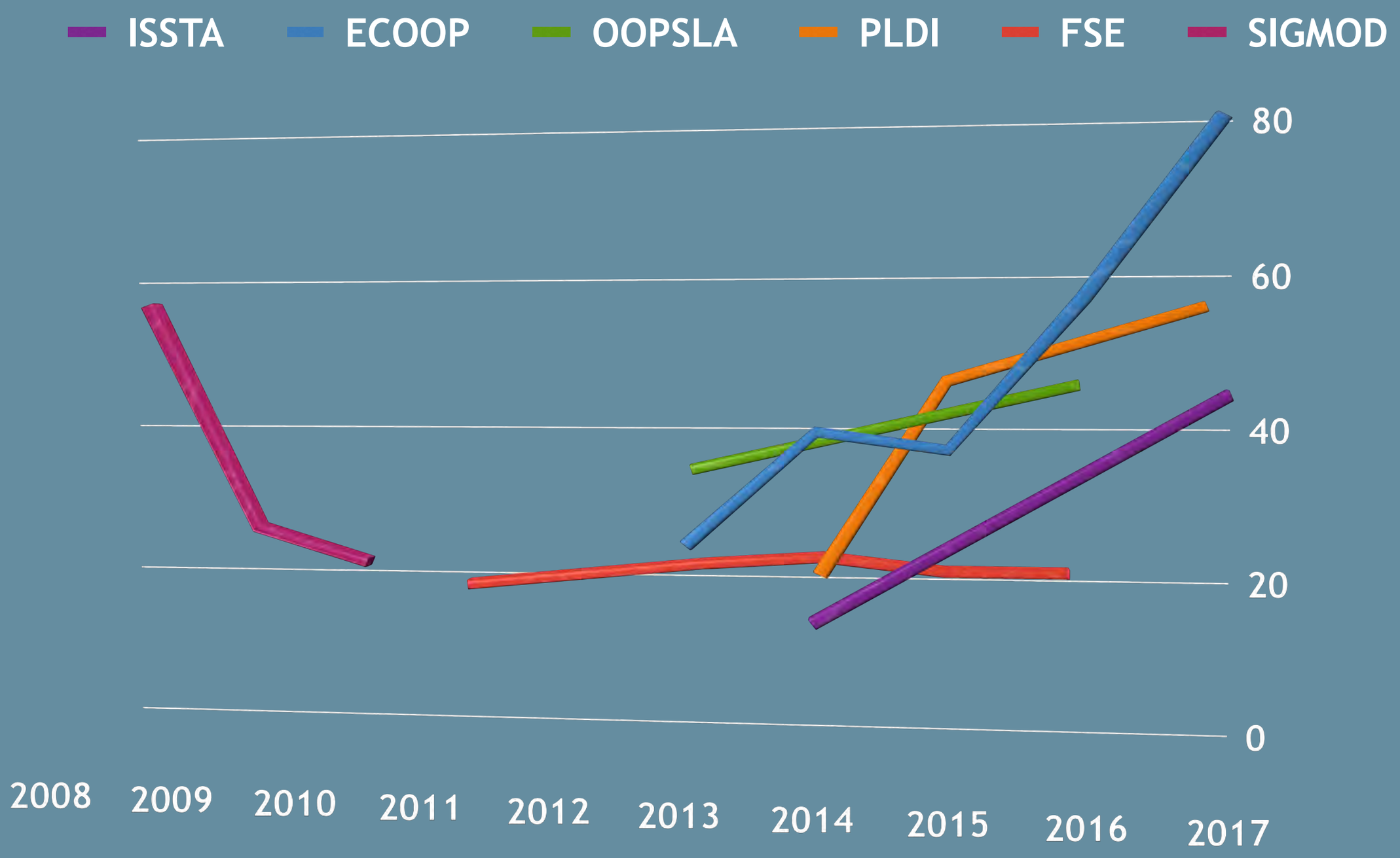
ARTIFACT

Repeatability

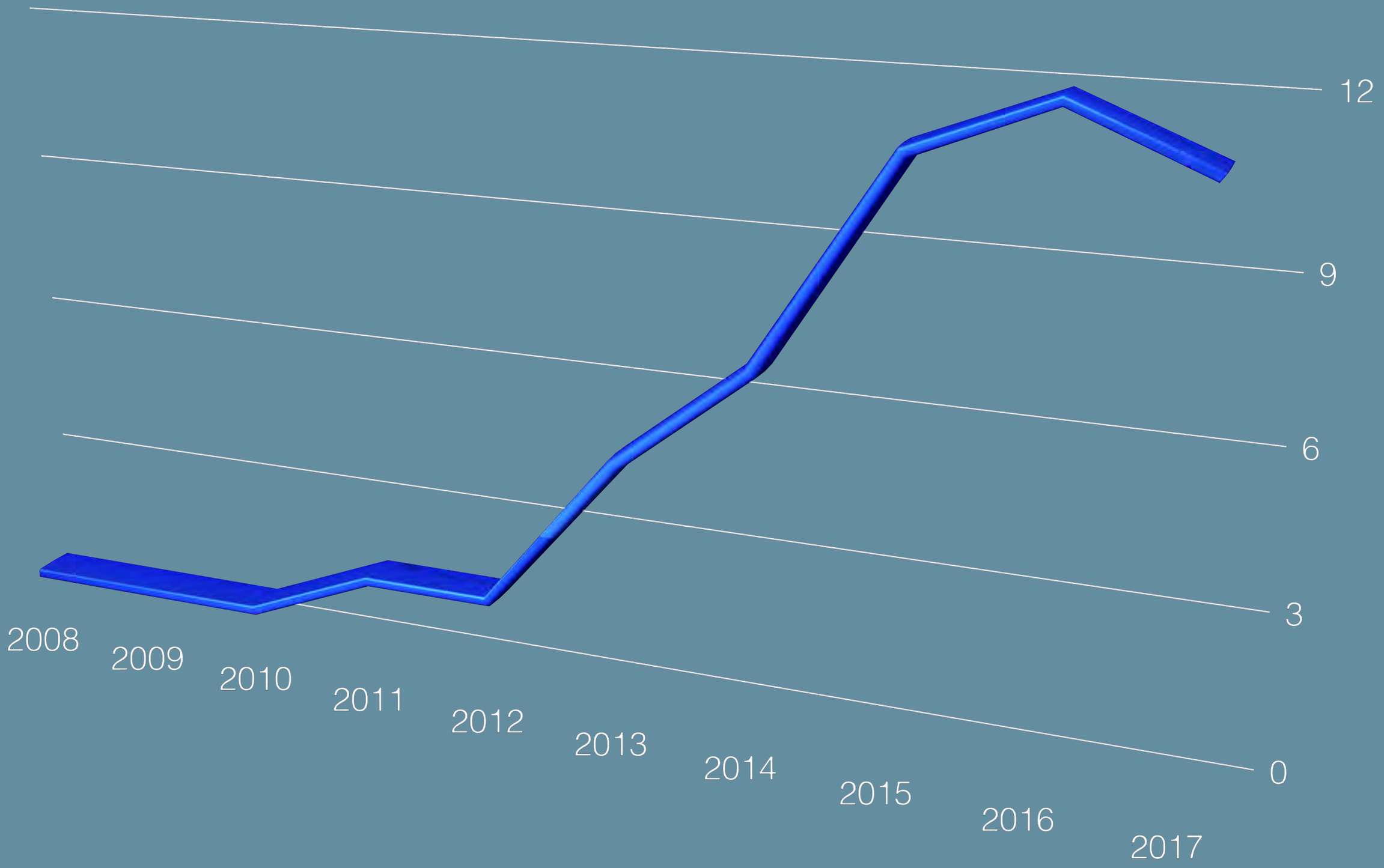
Reproducibility

Benefaction

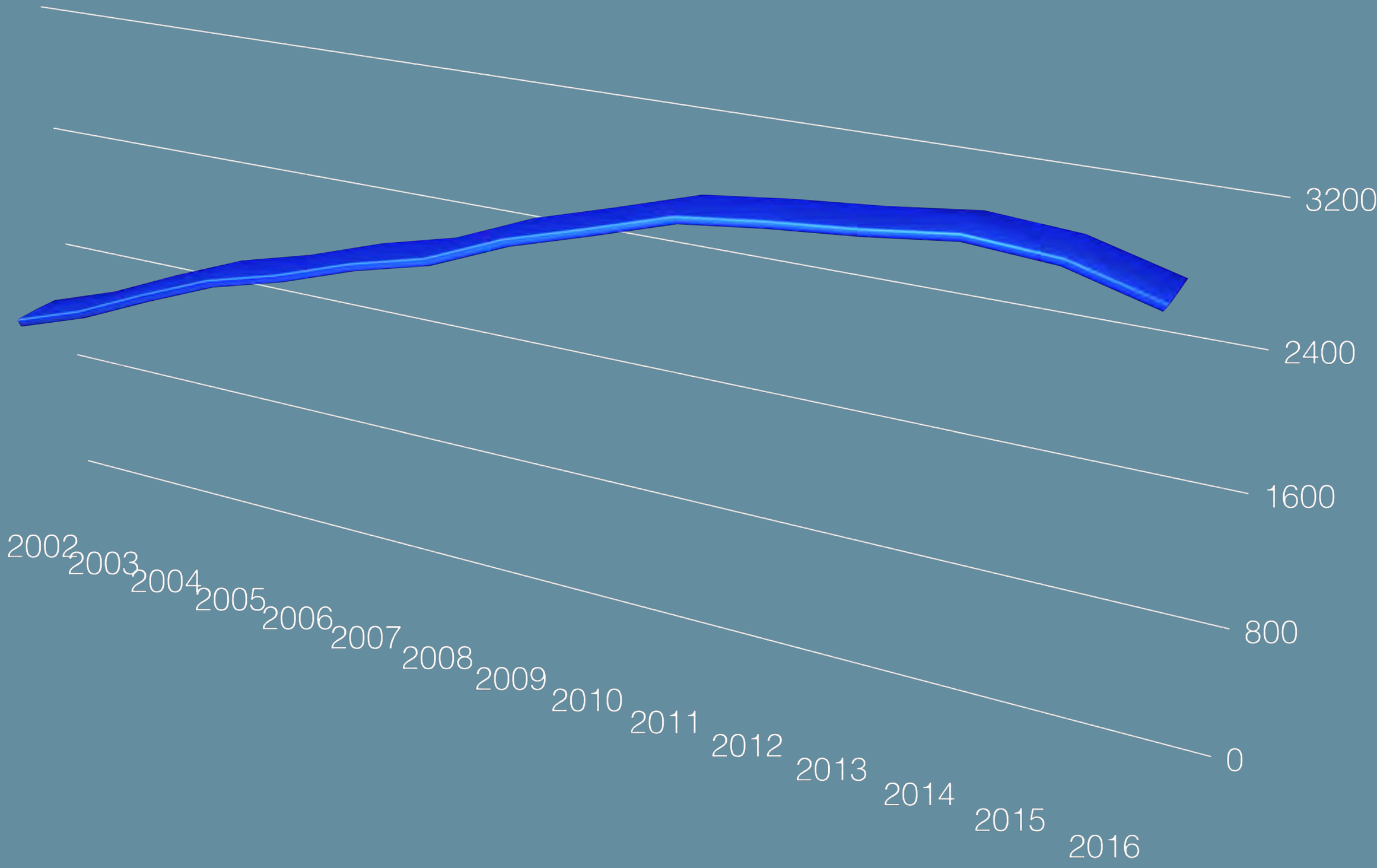
Accepted Artifacts / Accepted Papers (%)



Conferences with AE



Publication Venues (Dblp)



Sharing Proposal
— #5 —

Punishing Bad Behavior

Grant application
#: [redacted]

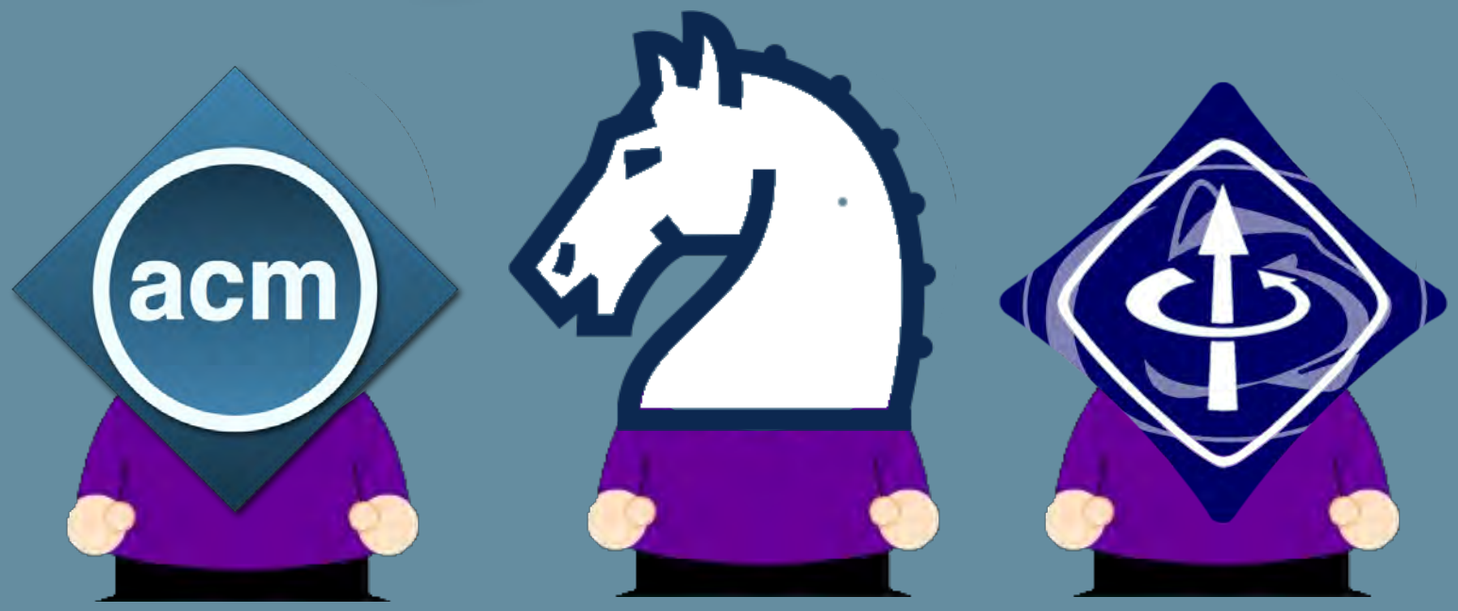
We will make our data and software available.

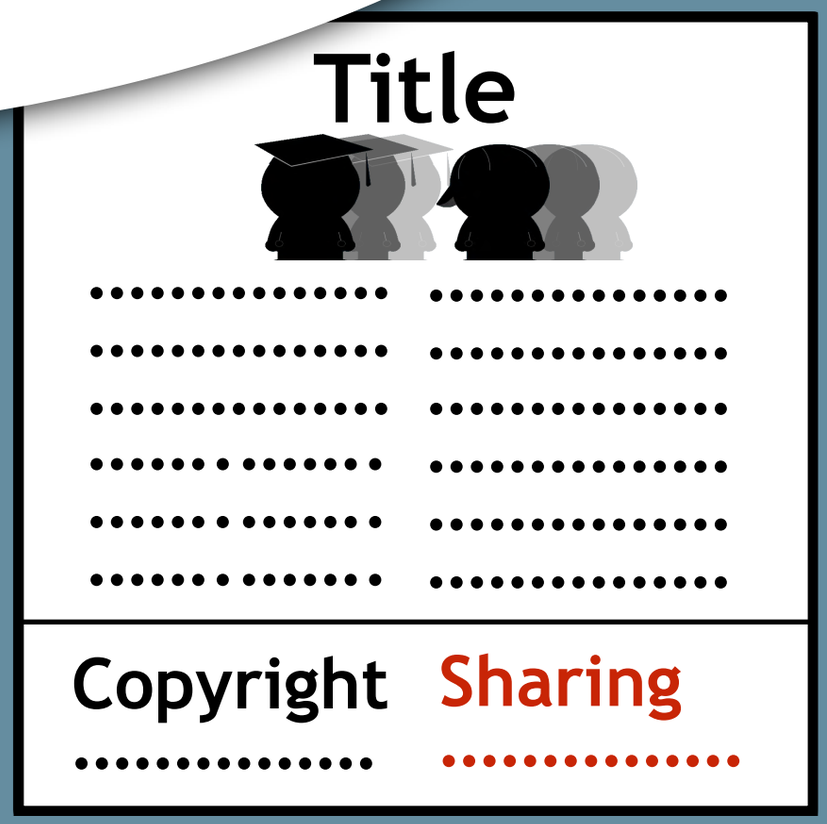
Random Audit!

Are you sharing like you promised in the grant application?

Sharing Contract!

What level of sharing are you committing to?

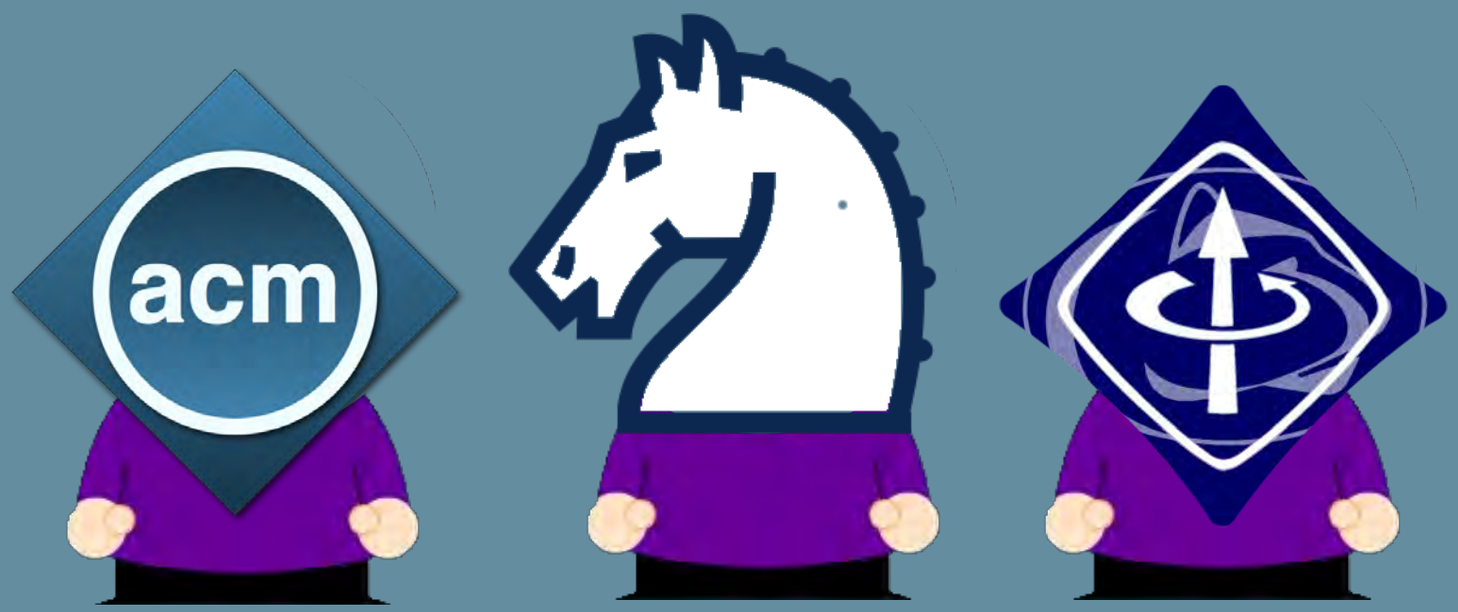




Author

Sharing Contract

- **License:** …
- **Artifacts:** source code, data, …
- **Where:** …
- **Support:** …



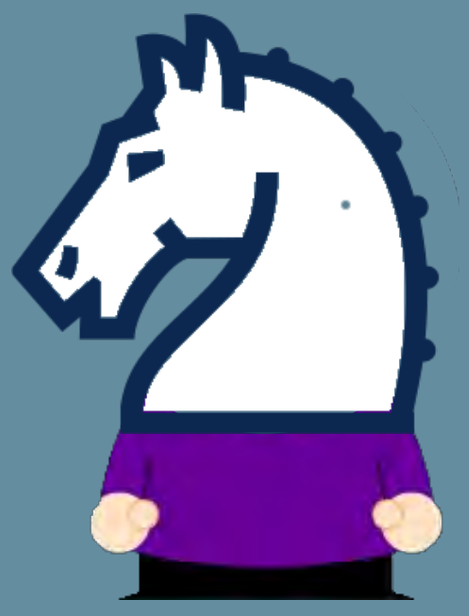
Sharing Contract

- **License:** …
- **Artifacts:** source code, data, …
- **Where:** …
- **Support:** …

Accept/ Reject?

Author

Reviewer



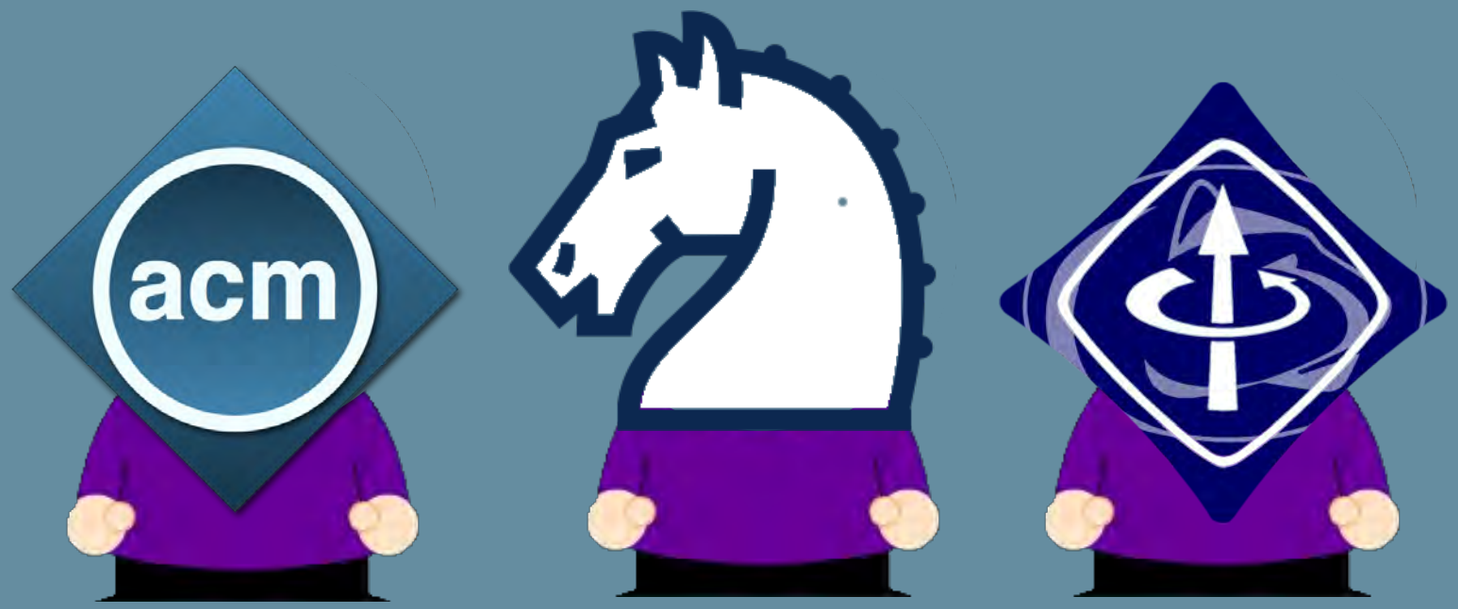
Sharing Contract

- **License:** …
- **Artifacts:** source code, data, …
- **Where:** …
- **Support:** …

You promised!

Author

Reader



Sharing Contract

- **License:** …
- **Artifacts:** source code, data, …
- **Where:** …
- **Support:** …

You promised!

Author

Reader

acm

nature

Sharing Proposal

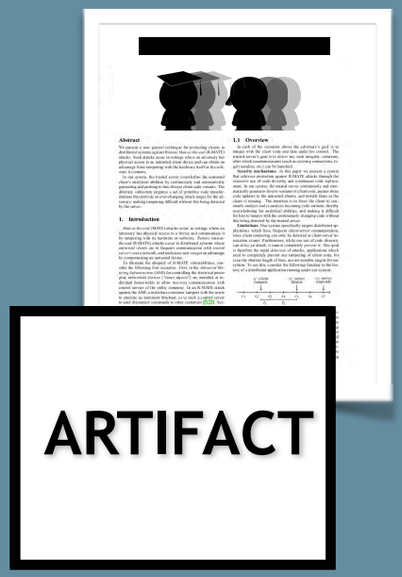
— #6 —

Optional Sharing

Sharing Proposal

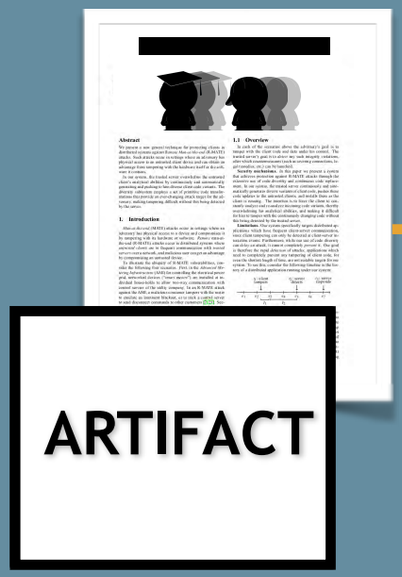
— #6 —

Mandated Sharing



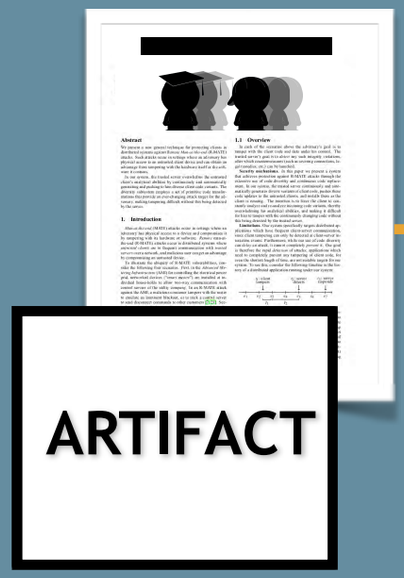
ARTIFACT


Volume 2, No. 1
March 1975
JOURNAL OF
Development
ECONOMICS
Editors:
ALAN S. MANNE
T.N. SRINIVASAN
Review Editor:
PADMA DESAI
Associate Editors:
I. ADELMAN
E.L. BACHA
J.N. BHAGWATI
M. BRUNO
H.J. BRUTON
M. CELASUN
P. DASGUPTA
DIAZ ALEJANDRO
I. ENCARNACION
E. GARCIA
M. GARGOURI
D. GHAI
R. JOLLY
J.M. KATZ
L.B.M. MENNES
Y. MURAKAMI
H.M.A. ONITIRI
G. PAPANDREOU
J. SANDEE
L. SOLIS
L.G. STOLERU
L.J. TAYLOR
J.E. WEISSKOPF
CONTENTS
D.B. Keesing, Economic Lessons from China 1
B.K. Kapur, Money as a Medium of exchange and monetary growth in an underdevelopment context 33
V.E. Tokman, Income distribution, technology and employment in developing countries: An application to Ecuador 49
Book Review by V. Corbo 81
UNIVERSITY OF ARIZONA
JUN 1 1 1975
LIBRARY



ARTIFACT

Artifacts submitted with paper?

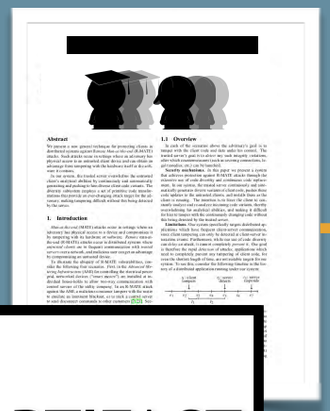


ARTIFACT

Artifacts submitted with paper?

Yes

Artifacts permit replication?



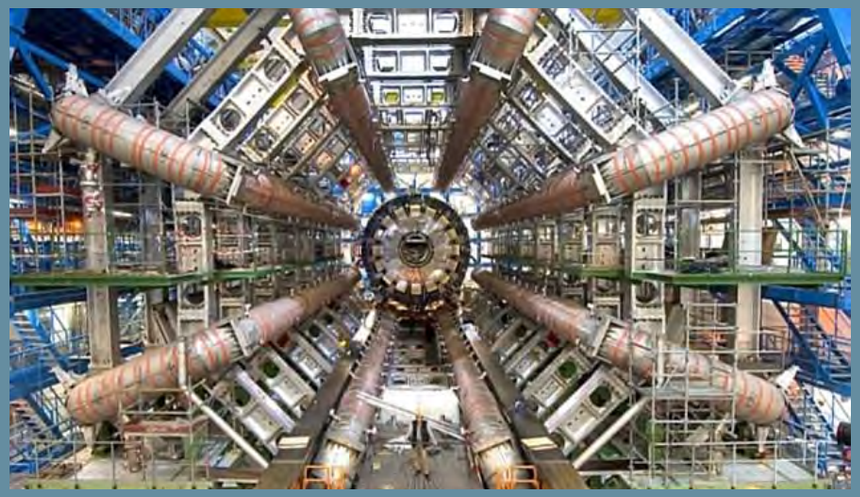
ARTIFACT

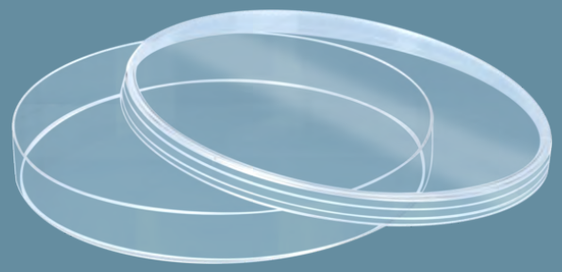
Artifacts submitted with paper?

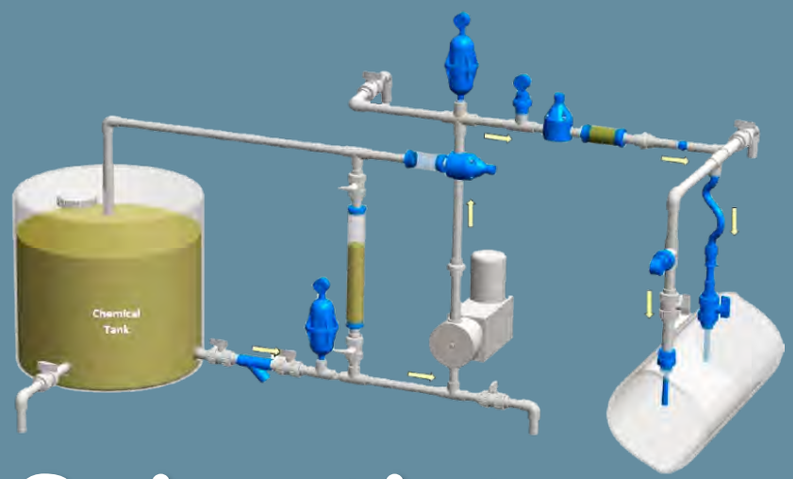
Yes

Artifacts permit replication?

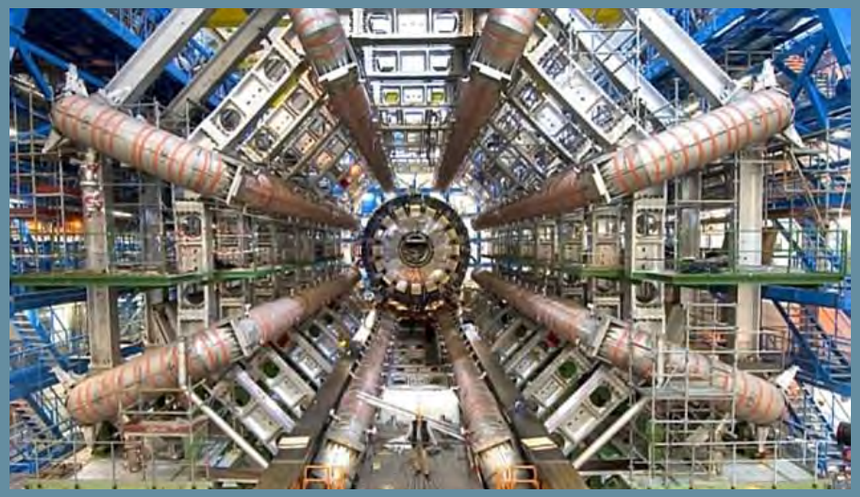
Review paper

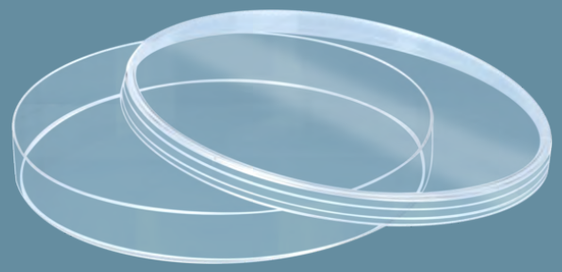




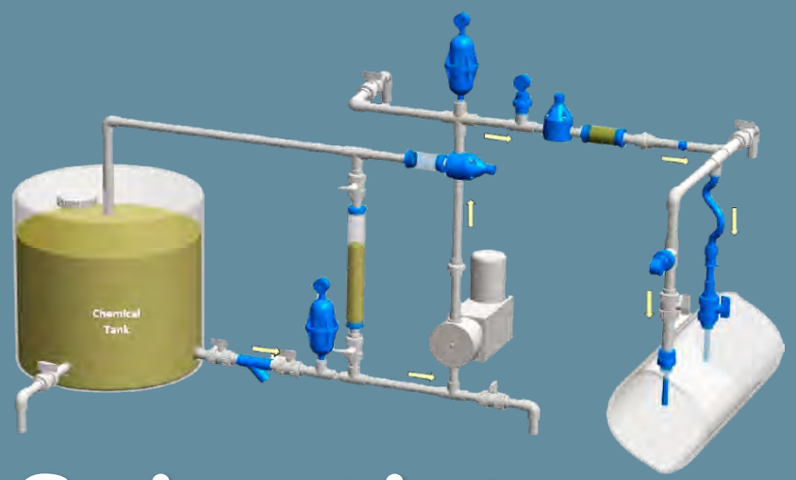


Scientist





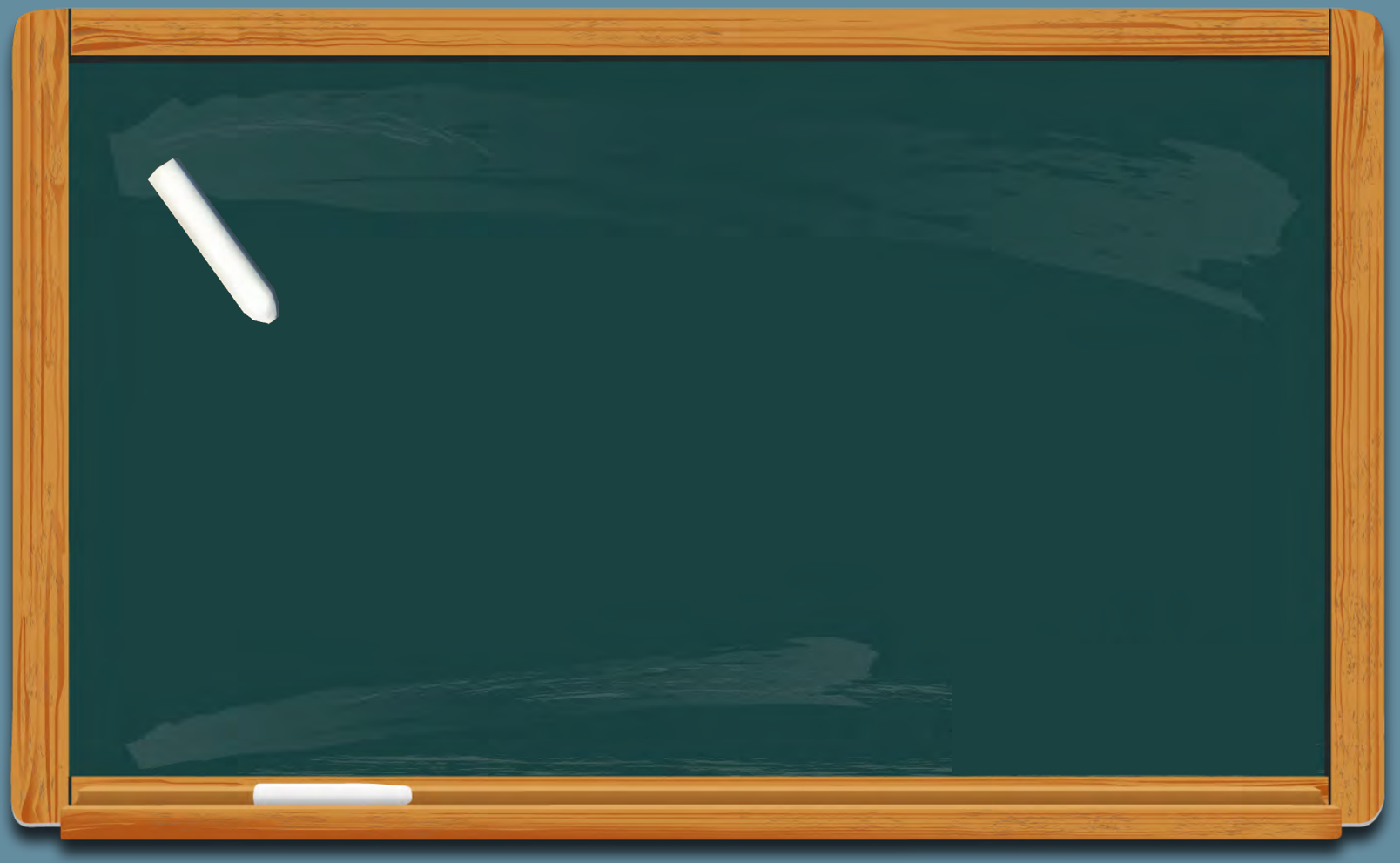
01



Computer Scientist

Scientist

Sharing Proposal
— #7 —



Sharing Proposal
— #7 —

CS Research Methods Courses?

CS Research Methods Courses?

- Reading, writing, presenting, reviewing papers
- Experimental design
- Statistics, data processing, visualization
- Proposal writing, career issues
- Intellectual property, research ethics

CS Research Methods Courses?

- Reading, writing, presenting, reviewing papers
- Experimental design
- Statistics, data processing, visualization
- Proposal writing, career issues
- Intellectual property, research ethics

Reproducibility???

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING

2.671 Measurement and Instrumentation

Keeping a complete and accurate record of experimental methods and data … **could someone else**, … use your notebook to **repeat your work**, and obtain the same results?

Reproducibility PI Manifesto

Lorena Barba

I pledge to

- ☑ teach grad students about reproducibility
- ☑ share artifacts at the time of submission
- ☑ add a *reproducibility statement* to papers

http://lorenabarba.com/gallery/reproducibility-pi-manifesto/

The dissertation proposal should state **if and how** they will provide access to code and data to support reproducibility.

Sharing Proposal

— #8 —

All I Really Need to Know I Learned in Kindergarten

Why do we care about reproducibility and repeatability?

Why do we care about reproducibility and repeatability?

Why do we care about reproducibility and repeatability?

Why do we care about reproducibility and repeatability?

Dear B, I read your nice paper, thanks for sharing the code! However, I'm unable to reproduce your results.

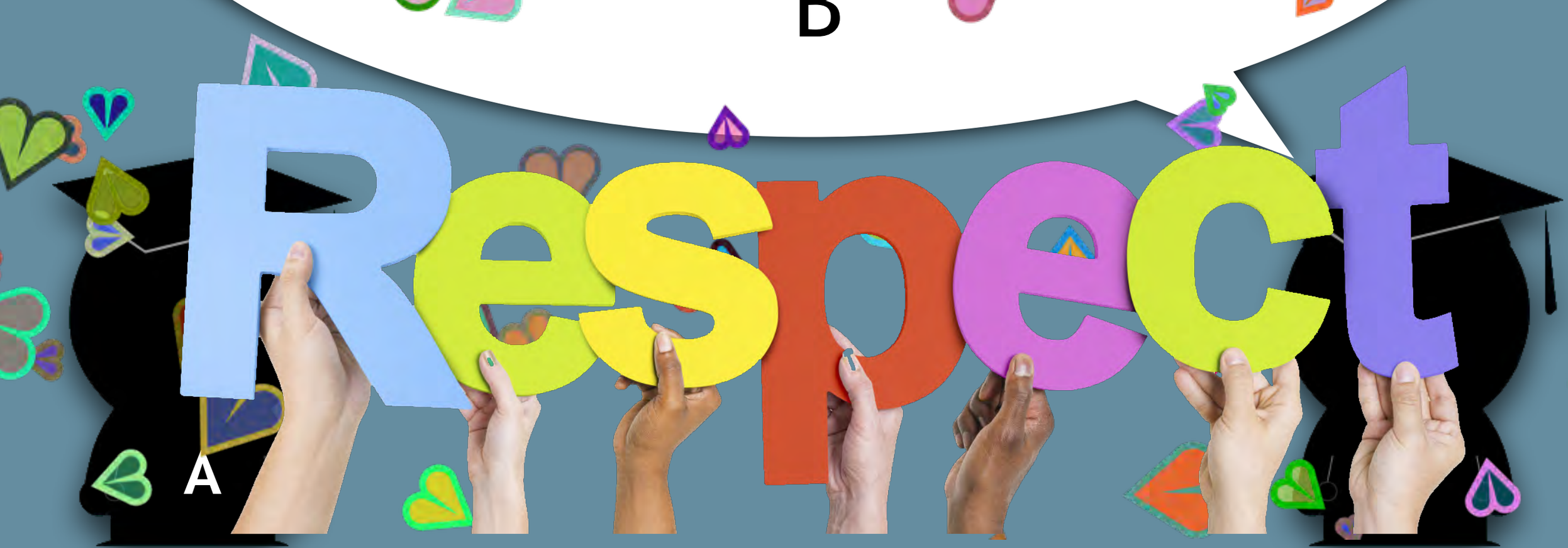
Sincerely,
A

Dear A, thank you for pointing out our errors!

Best wishes,
B

Dear A, thank you for pointing out our errors!

Best wishes,
B



BLOG@CACM

Yes, Computer Scientists Are Hypercritical

By Jeannette M. Wing
October 6, 2011
Comments (15)

VIEW AS: SHARE:

Are computer scientists hypercritical? Are we more critical than scientists and engineers in other disciplines? Bertrand Meyer's August 22, 2011 The Nastiness Problem in Computer Science blog post partially makes the argument referring to secondhand information from the National Science Foundation (NSF). Here are some NSF numbers to back the claim that we are hypercritical.

This graph plots average reviewer ratings of all proposals submitted from 2005 to 2010 to NSF overall (red line), just Computer & Information Science & Engineering (CISE) (green line), and NSF minus CISE (blue line). Proposal ratings are based on a scale of 1 (poor) to 5 (excellent). For instance, in 2010, the average reviewer rating across all CISE programs is 2.96; all NSF directorates including CISE, 3.24; all NSF directorates excluding CISE, 3.30.

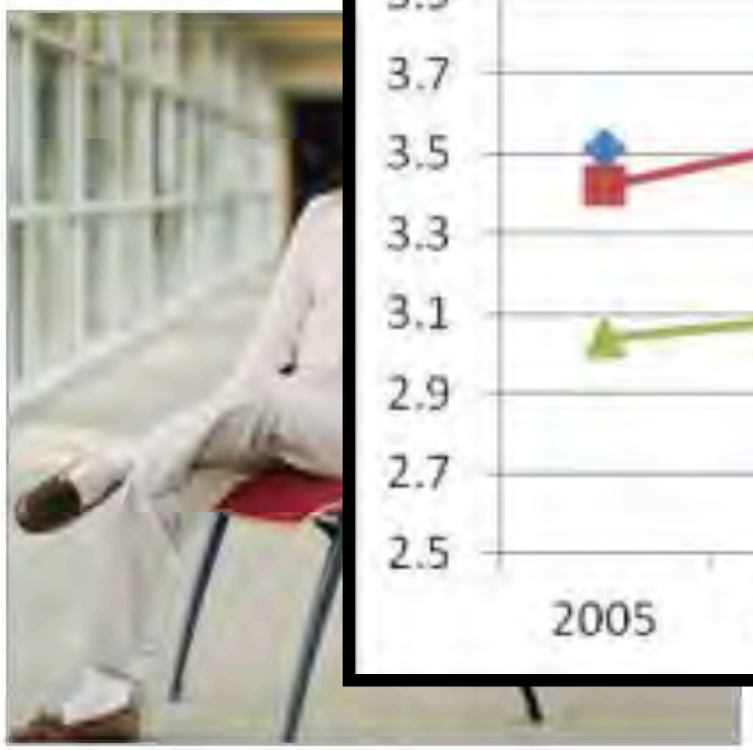
https://cacm.acm.org/blogs/blog-cacm/134743-yes-computer-scientists-are-hypercritical/fulltext

BLOG@CACM

Yes, Computer Scientists Are Hypercritical

By Jeannette M. Win
October 6, 2011
Comments (15)

VIEW AS:



average reviewer rating across all CISE programs is 2.96; all NSF directorates including CISE, 3.24; all NSF directorates excluding CISE, 3.30.

https://cacm.acm.org/blogs/blog-cacm/134743-yes-computer-scientists-are-hypercritical/fulltext

DBMS Research First 50 Years, Next 50 Years

Jeffrey F. Naughton

- SIGMOD 2010
- 350 submissions
- Number of papers with all reviews "accept" or higher:

1

BLOG@CACM

The Nastiness Problem in Computer Science

By Bertrand Meyer
August 22, 2011
Comments (33)

VIEW AS: SHARE:

Are we malevolent grumps? Nothing personal, but as a community computer scientists sometimes seem to succumb to negativism. They admit it themselves. A common complaint in the profession is that instead of taking a cue from our colleagues in more cogently organized fields such as physics, who band together for funds, promotion, and recognition, we are incurably fractious. In committees, for example, we damage everyone's chances by badmouthing colleagues with approaches other than ours. At least this is a widely perceived view (*"Circling the wagons and shooting inward,"* as Greg Andrews put it in a recent discussion). Is it accurate?

One statistic that I have heard cited is that in 1-to-5 evaluations of projects submitted to the U.S. National Science Foundation the

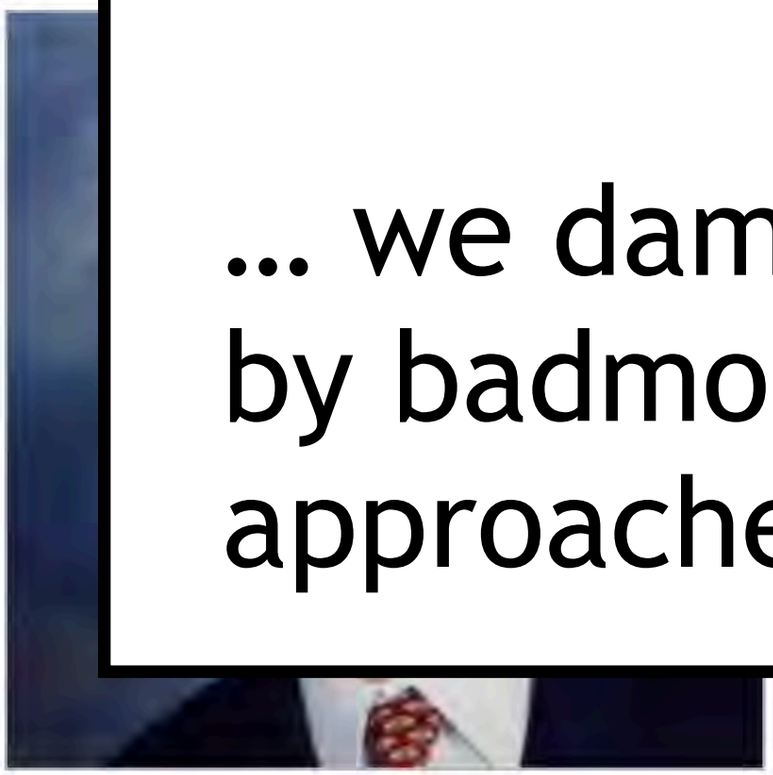
https://cacm.acm.org/blogs/blog-cacm/123611-the-nastiness-problem-in-computer-science/fulltext

BLOG@CACM

The Nastiness Problem in Computer Science

By Bertrand Meyer
August 22, 2011
Comments (33)

VIEW



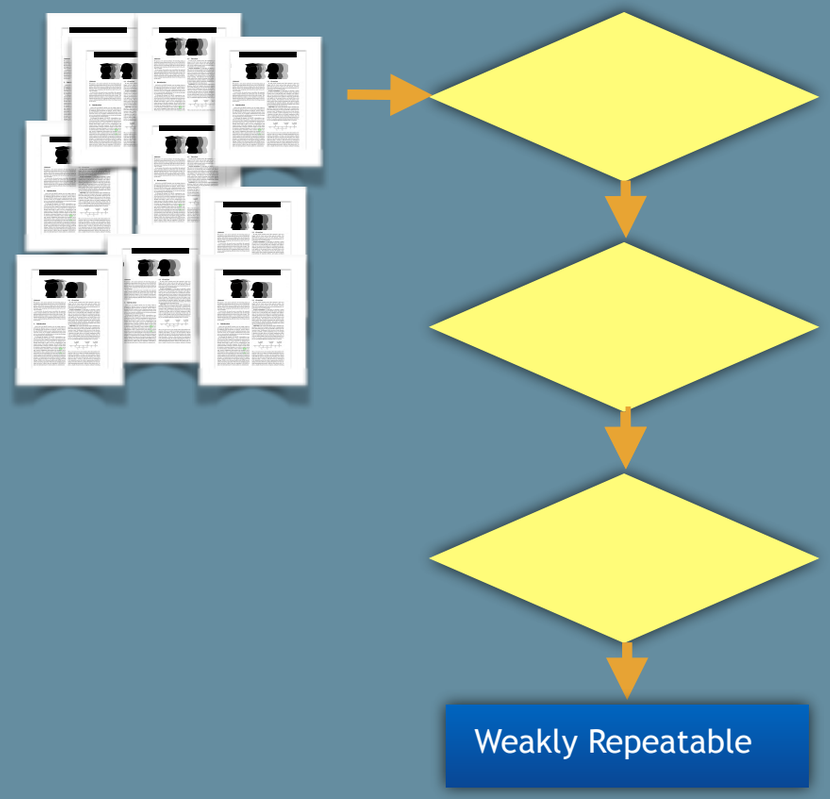
Are we malevolent grumps?

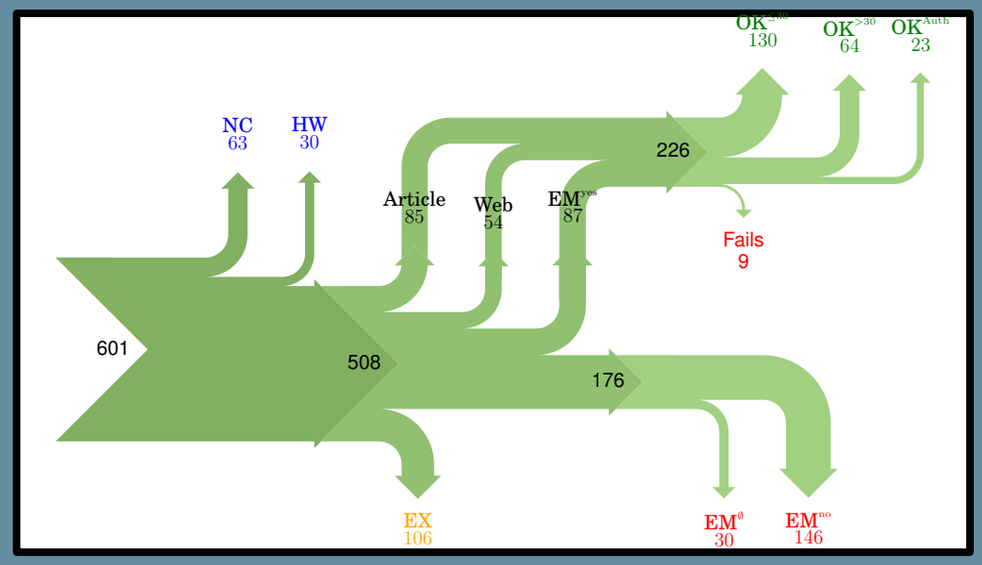
… we damage everyone's chances by badmouthing colleagues with approaches other than ours.

One statistic that I have heard cited is that in 1-to-5 evaluations of projects submitted to the U.S. National Science Foundation the

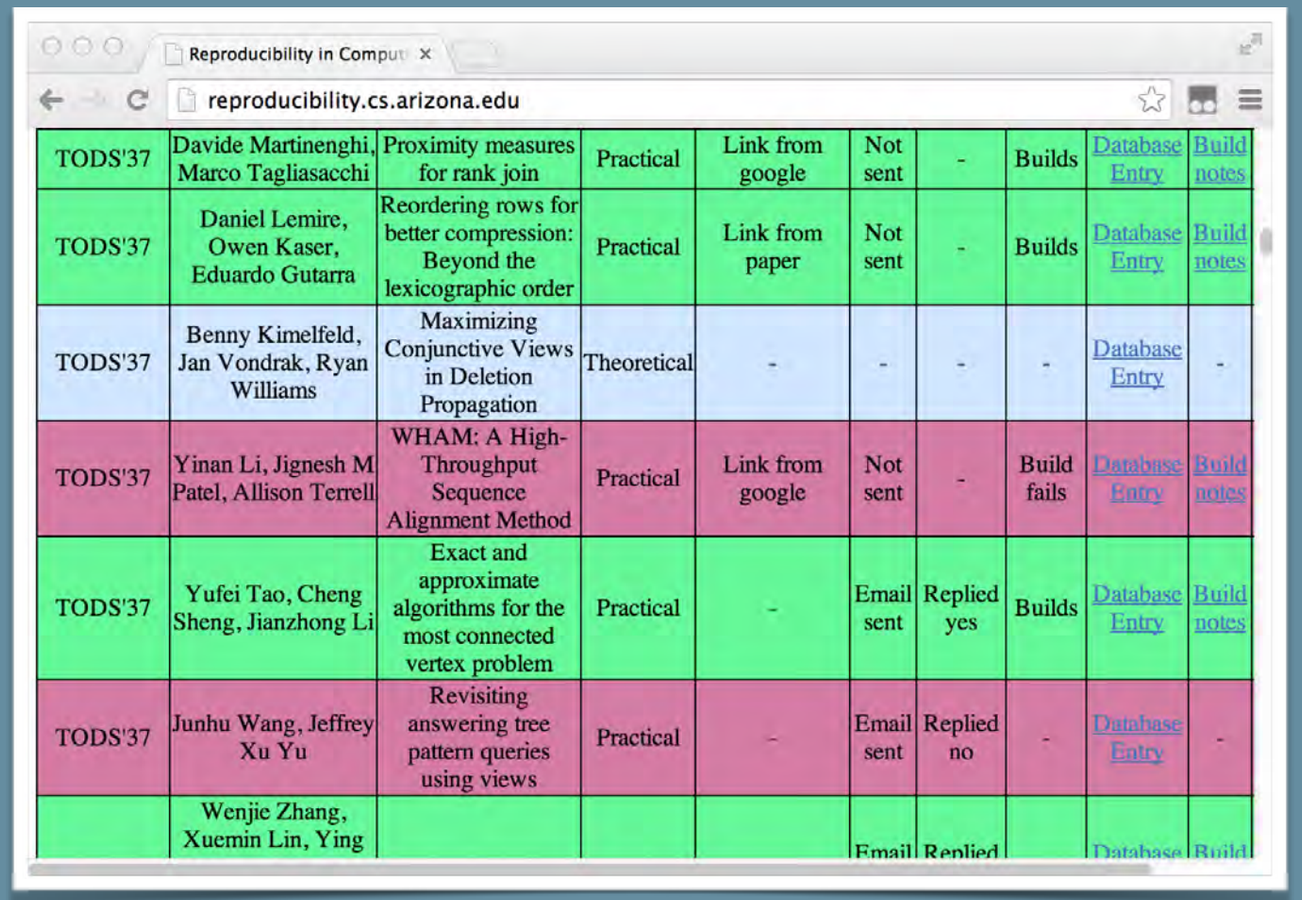
https://cacm.acm.org/blogs/blog-cacm/123611-the-nastiness-problem-in-computer-science/fulltext

What Happened Next?





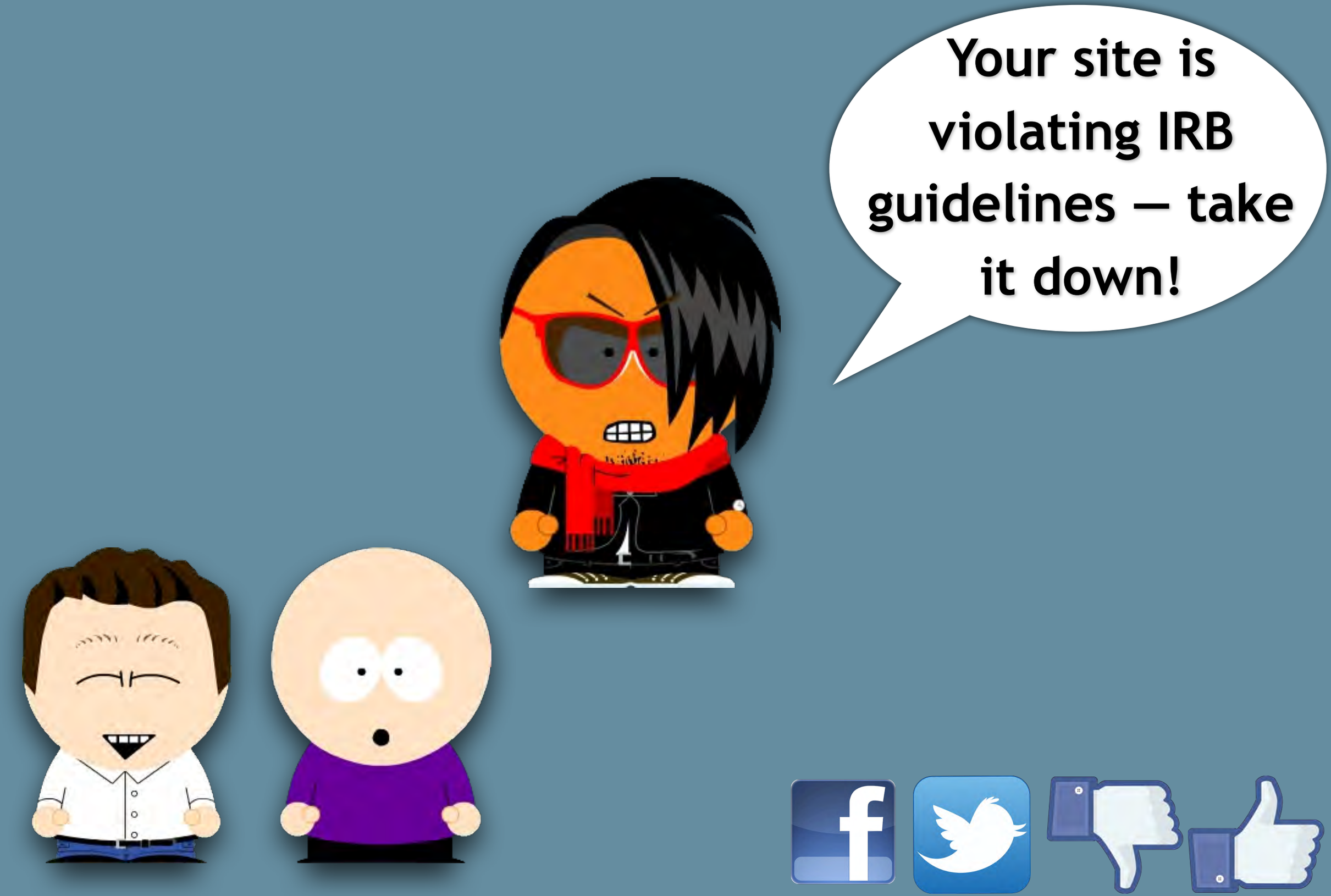
Submitted Paper



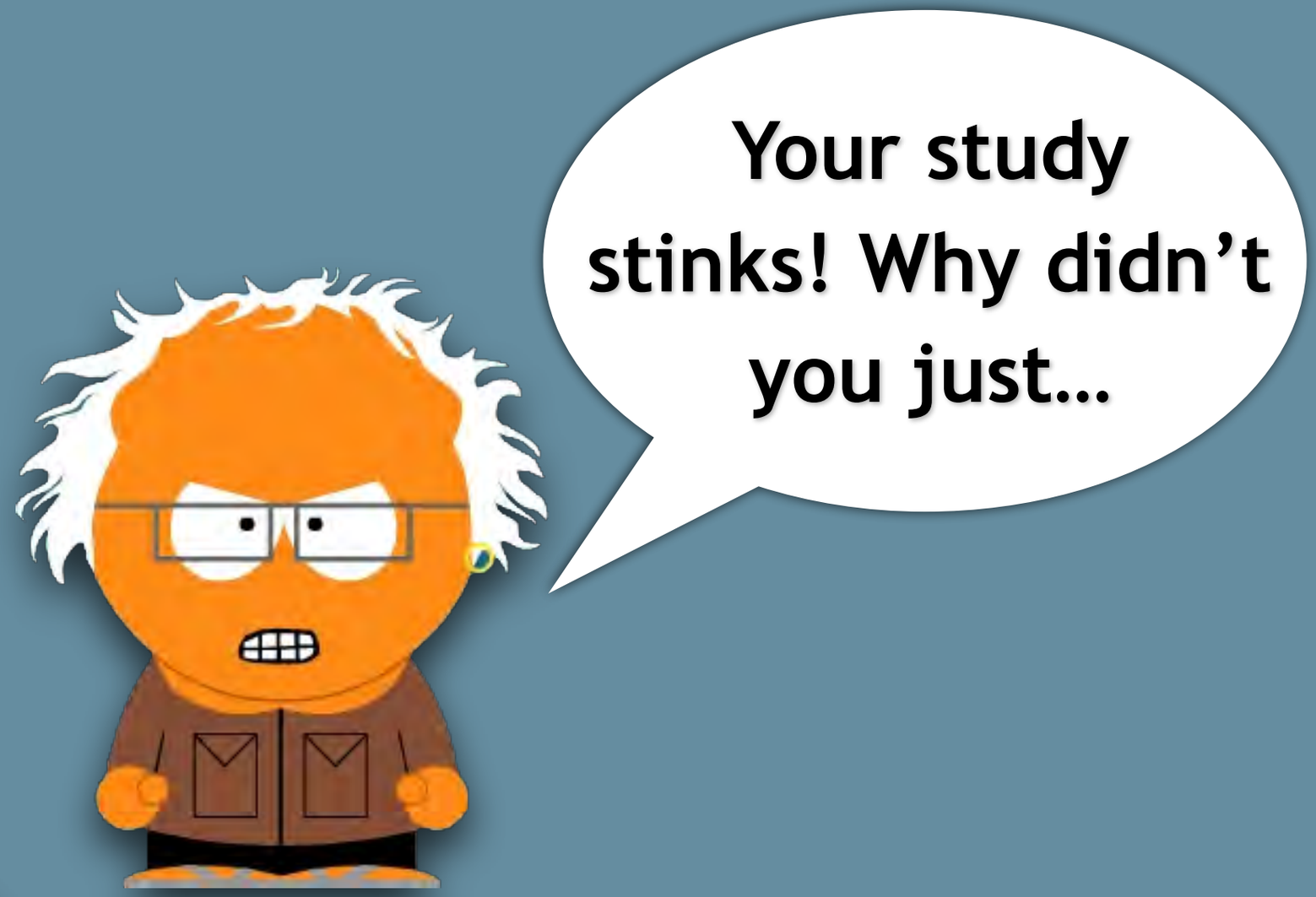
TODS'37	Davide Martinenghi, Marco Tagliasacchi	Proximity measures for rank join	Practical	Link from google	Not sent	-	Builds	Database Entry	Build notes
TODS'37	Daniel Lemire, Owen Kaser, Eduardo Gutarra	Reordering rows for better compression: Beyond the lexicographic order	Practical	Link from paper	Not sent	-	Builds	Database Entry	Build notes
TODS'37	Benny Kimelfeld, Jan Vondrak, Ryan Williams	Maximizing Conjunctive Views in Deletion Propagation	Theoretical	-	-	-	-	Database Entry	-
TODS'37	Yinan Li, Jignesh M Patel, Allison Terrell	WHAM: A High-Throughput Sequence Alignment Method	Practical	Link from google	Not sent	-	Build fails	Database Entry	Build notes
TODS'37	Yufei Tao, Cheng Sheng, Jianzhong Li	Exact and approximate algorithms for the most connected vertex problem	Practical	-	Email sent	Replied yes	Builds	Database Entry	Build notes
TODS'37	Junhu Wang, Jeffrey Xu Yu	Revisiting answering tree pattern queries using views	Practical	-	Email sent	Replied no	-	Database Entry	-
	Wenjie Zhang, Xuemin Lin, Ying				Email	Replied		Database	Build

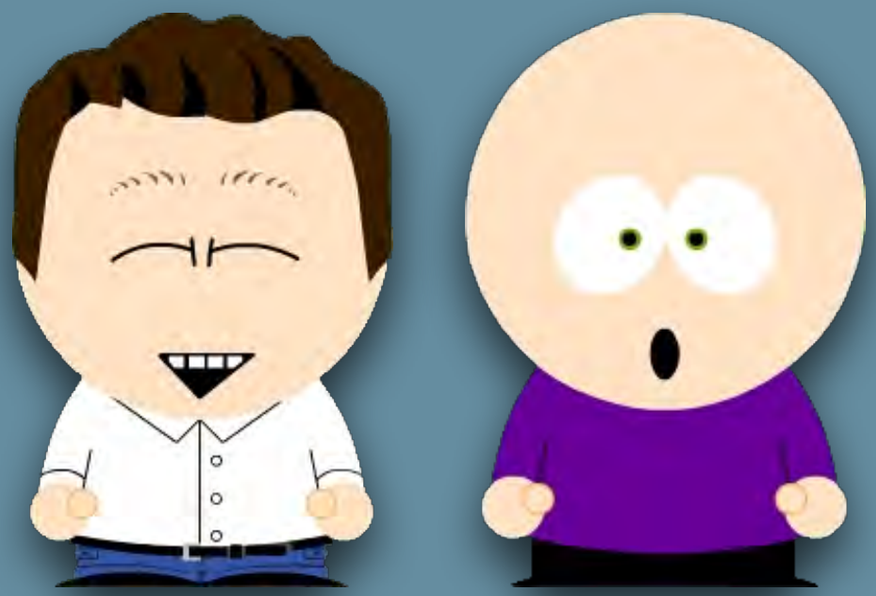
Hate Us on Facebook!

Hate Us on Facebook!



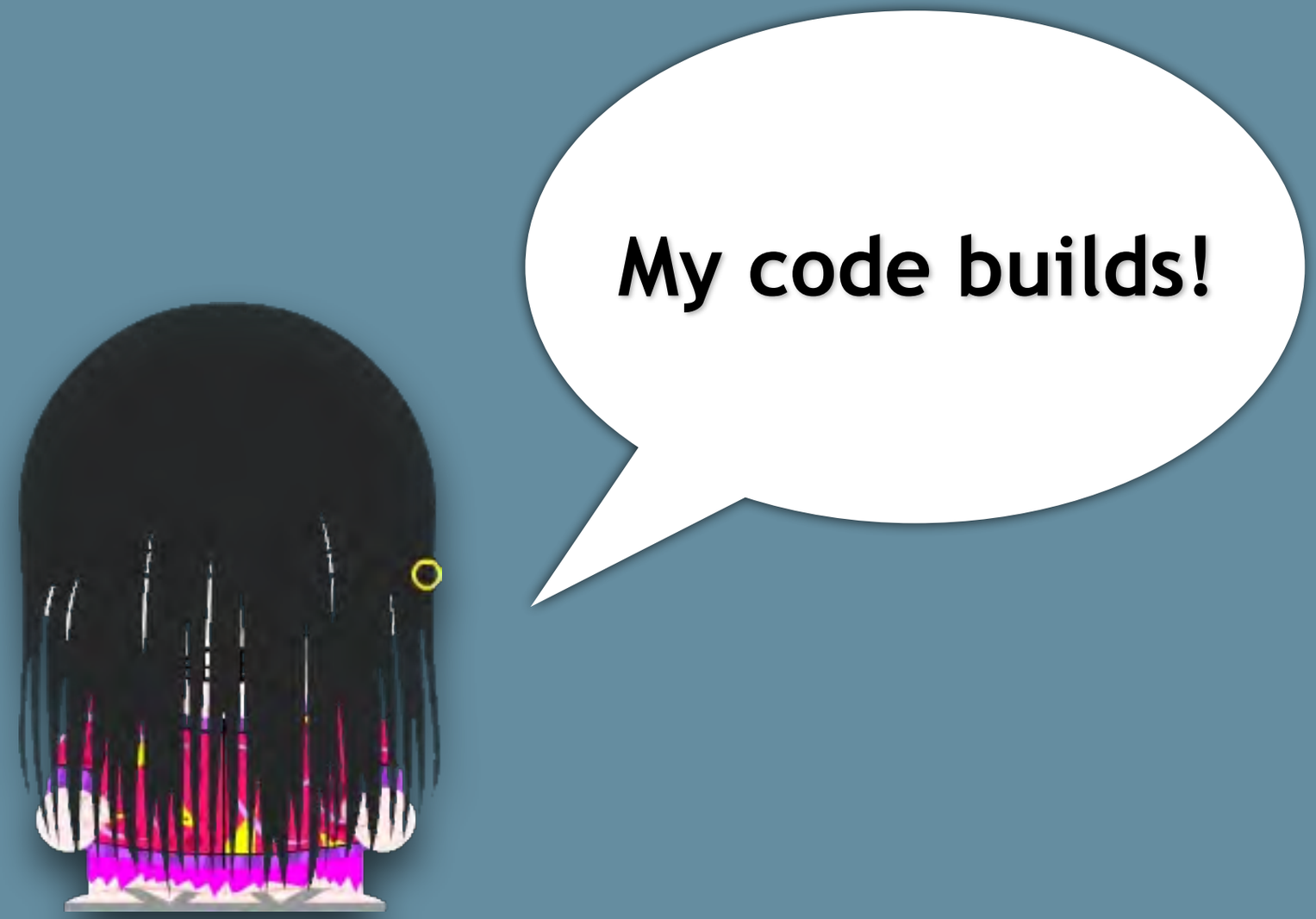
Hate Us on Facebook!

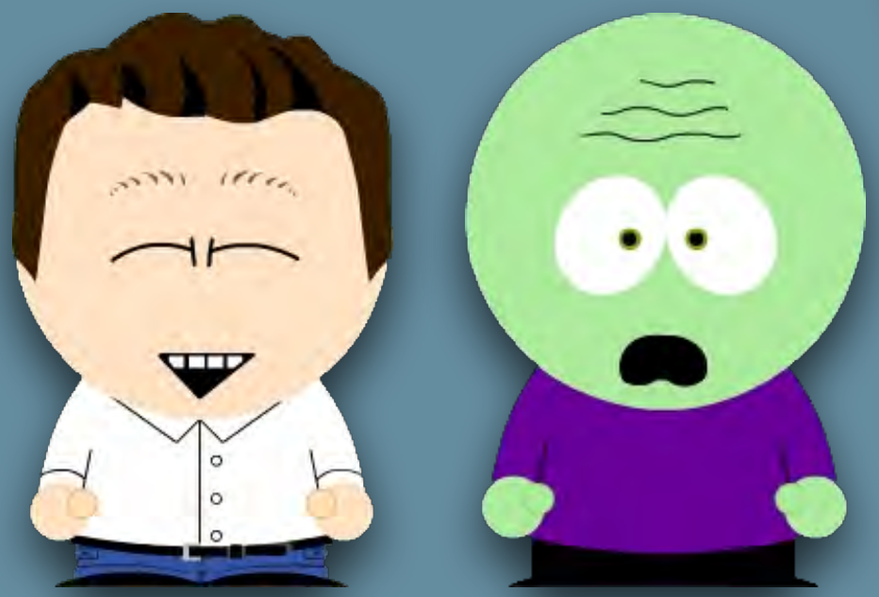




Hate Us on Facebook!

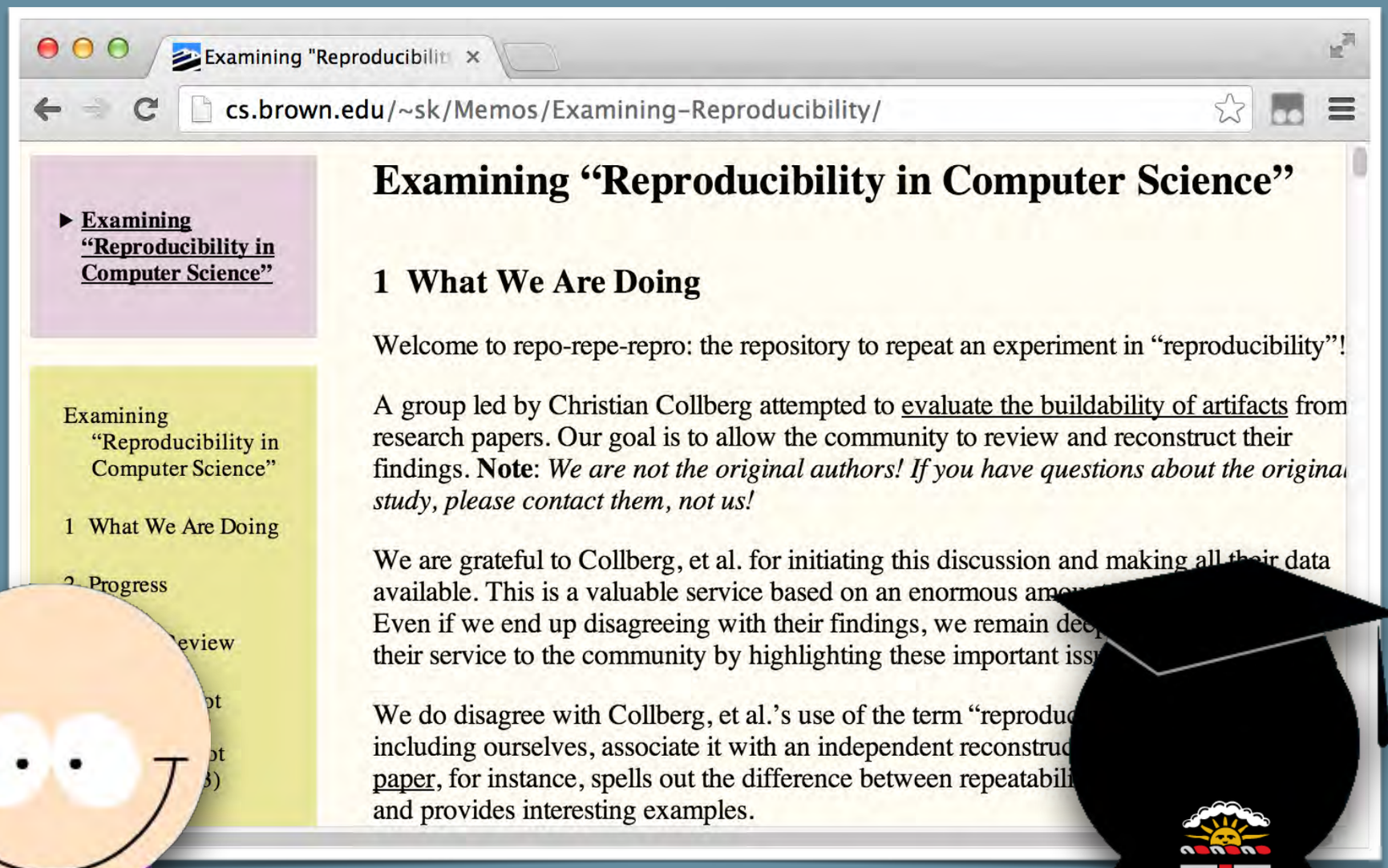
Hate Us on Facebook!





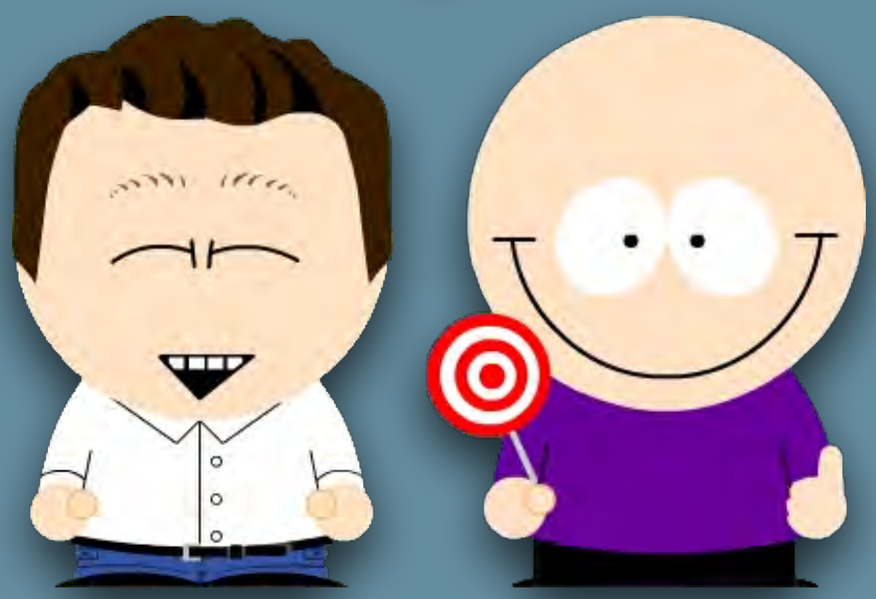
Hate Us on Facebook!

Turnabout is Fair Play!



http://cs.brown.edu/~sk/Memos/Examining-Reproducibility/

Please let us know if there's anything we can do in support of your efforts to examine our paper! We think your effort is terrific!



Reproducibility in Computer Sc ×

reproducibility.cs.arizona.edu/v1/index.html

Repeatability in Computer Science

Home::Table of Results — March 2014

Notice

Please disregard the results below — they are included for completeness but contain numerous errors. We have redone the study and report the new results.

We originally put up this website so that the reviewers of our submitted paper could have access to our raw data, code, and technical report, in case they wished to review it. It was never publicly announced. Nevertheless, the site became public knowledge, and over the last week we have received many emails pointing out many apparent errors in the data. Some of these errors are, no doubt, a consequence of the definition of reproducibility we used in the study:

> Can a CS student build the software within 30 minutes, including finding and installing any dependent software and libraries, and without bothering the authors?

As a result, we made another pass over the data, along with the people behind this site. We very much welcome these *reviews-of-the-reviews* - this is exactly the way science should work! Please don't hesitate to contact us should you have any further questions and comments.

This page is set to be unindexed by search engines.

Christian and Todd

f A group of independent researchers set out to verify our build results through a crowd-sourced effort; http://cs.brown.edu/~sk/Memos/Examining-Reproducibility

To encourage repeatable re
repeatability engineering a
commitments to sharing re

BY CHRISTIAN COLLBERG AND TODD

Repeatability in Computer Systems Research

backed up, we made a second ORA request, this time for the email messages among the authors, hoping to trace the whereabouts of the source code. The legal department first responded with: "... the records will not be produced pursuant to [ORA sub-clause]." When we pointed out reasons why this clause does not apply, the university relented but demanded $2,263.66 " ... to search for, retrieve, redact and produce such records." We declined the offer.

We instead made a Freedom of Information Act request to the National Science Foundation for the funded grant proposals that supported the research. In one, the principal investigator wrote, "We will also make our data and software available to the research community when appropriate." In the

IN 2012, WHEN reading a paper f
computer security conference, v
there is a clever way to defeat th
in the paper, and, in order to sh
the authors (faculty and graduat
ranked U.S. computer science d
for access to their prototype syst
no response. We thus decided t
algorithms in the paper but soo
obstacles, including a variable u
function defined but never used
formula that did not typecheck. We asked the authors for clarification and received a single response: "I unfortunately have few recollections of the work ... "

We next made a formal request to the university for the source code under the broad Open Records Act (ORA) of the authors' home state. The university's

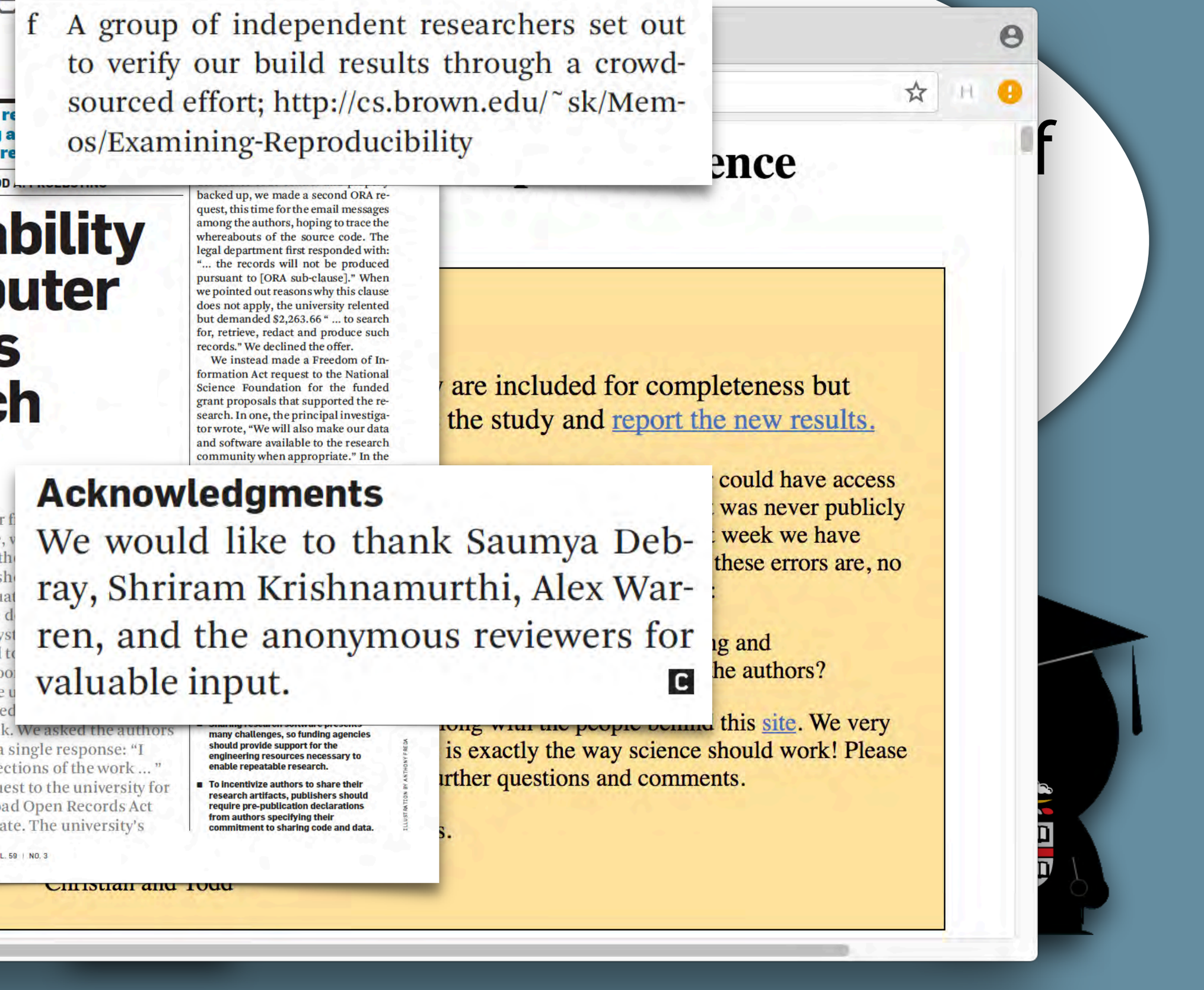
many challenges, so funding agencies should provide support for the engineering resources necessary to enable repeatable research.

- To incentivize authors to share their research artifacts, publishers should require pre-publication declarations from authors specifying their commitment to sharing code and data.

ILLUSTRATION BY ANTHONY FREDA

Acknowledgments

We would like to thank Saumya Debray, Shriram Krishnamurthi, Alex Warren, and the anonymous reviewers for valuable input.



f A group of independent researchers set out to verify our build results through a crowd-sourced effort; http://cs.brown.edu/~sk/Memos/Examining-Reproducibility

To encourage repeatable re
repeatability engineering a
commitments to sharing re

BY CHRISTIAN COLLBERG AND TODD

Repeatability in Computer Systems Re

backed up, we made a second ORA request, this time for the email messages among the authors, hoping to trace the whereabouts of the source code. The legal department first responded with: "... the records will not be produced pursuant to [ORA sub-clause]." When we pointed out reasons why this clause does not apply, the university relented but demanded $2,263.66 " ... to search for, retrieve, redact and produce such records." We declined the offer.

IN 2012, W
computer
there is a
in the pap
the author
ranked U.
for access
no respon
algorithm
obstacles,
function
formula t
for clarific
unfortuna

We next
the source code under the broad Open Records Act (ORA) of the authors' home state. The university's

research artifacts, publishers should require pre-publication declarations from authors specifying their commitment to sharing code and data.

ILLUSTRATION

Repeatability and Benefaction in Computer Systems Research

A Study and a Modest Proposal

University of Arizona TR 14-04

Christian Collberg collberg@gmail.com
Todd Proebsting proebsting@cs.arizona.edu
Alex M Warren amwarren@email.arizona.edu

ence

s.

Christian and Todd

ShriramKrishnamurthi
@ShriramKMurthi

Follow

They did **crap** work, would not admit to when caught out and even pretended it hadn't happened.

https://twitter.com/ShriramKMurthi/status/863462366226370561

…these researchers have done a disservice to science by publishing a study they knew to be **horse manure**, and then piling more **bull crap** on it when caught … they are simply trying to build a reputation off a problem they don't really care to solve …

https://www.facebook.com/jvitekjr/posts/10155809013435351

To the University of Arizona Institutional Review Board:

Revoke their IRB permission!

1. Their deception study was bad
   — I don't trust them!

FindResearch.org

1. Their deception study was bad
   — I don't trust them!
2. They're violating my privacy!

FindResearch.org

The authors

- *have*
- *have not*

verified

1. Their deception study was bad
   — I don't trust them!
2. They're violating my privacy!
3. They're spying on my computer!

FindResearch.org

The authors

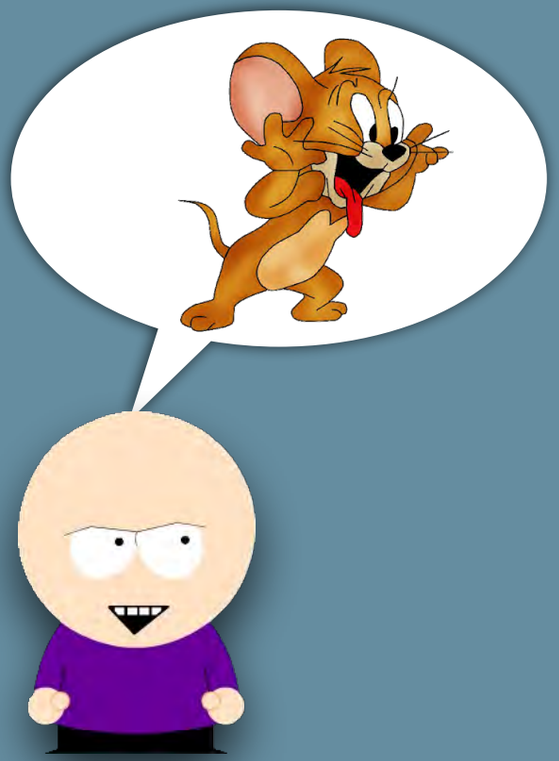
- ***have***
- ***have not***

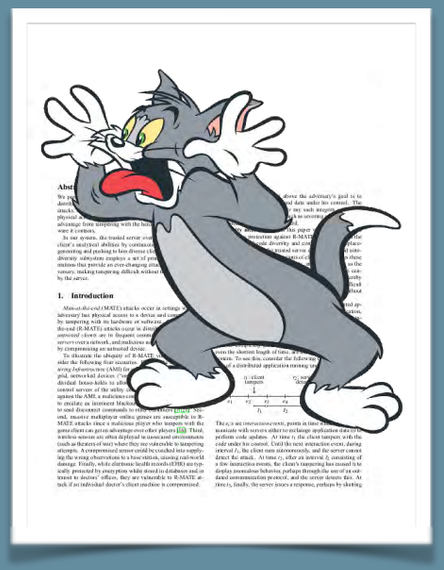
verified

3rd Law of Artifact Sharing

Without a culture of respectful academic interchange, where failure is seen as an accepted part of the progression of science, sharing will not become default behavior.

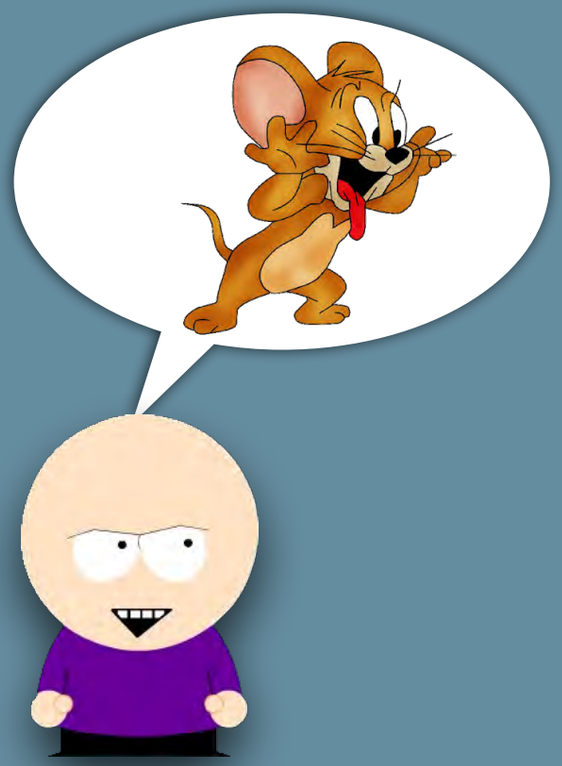
Risks vs. Rewards

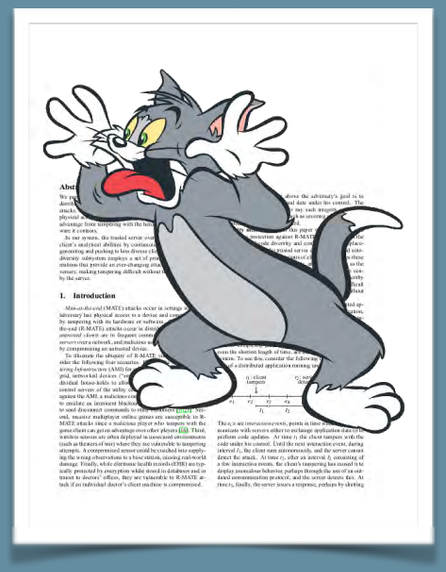


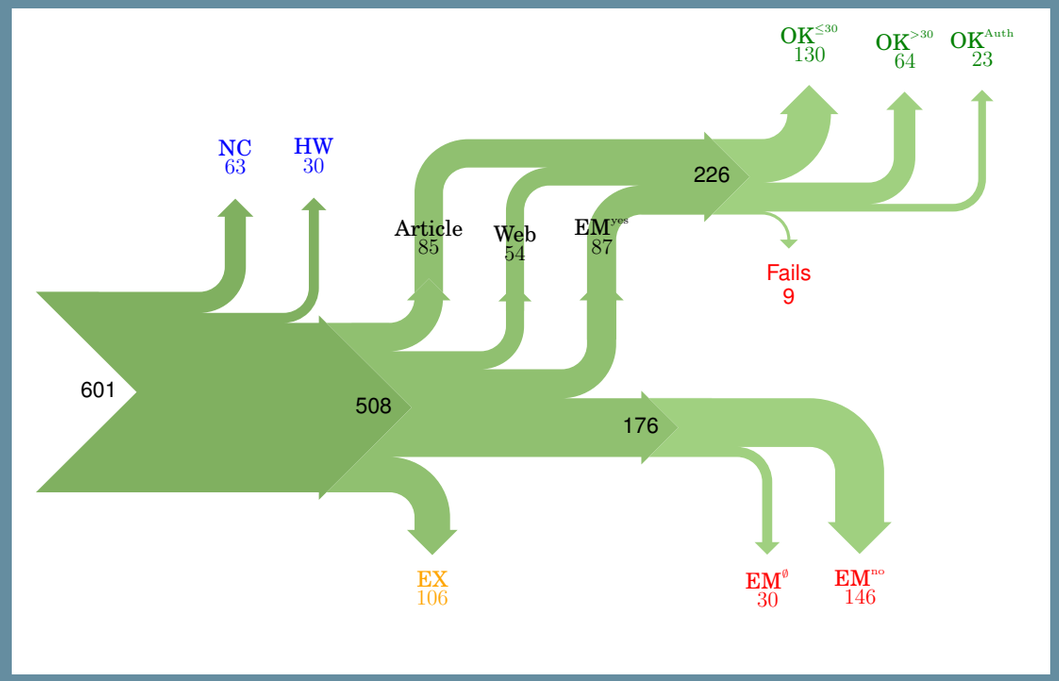


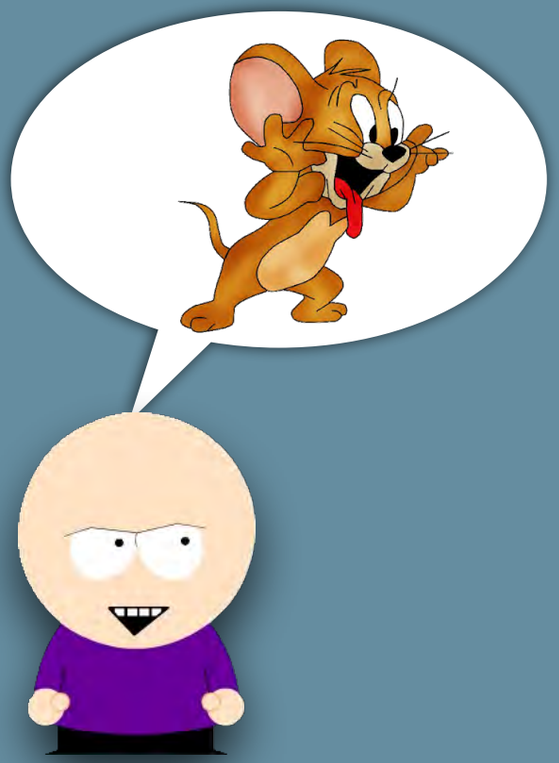
Abst

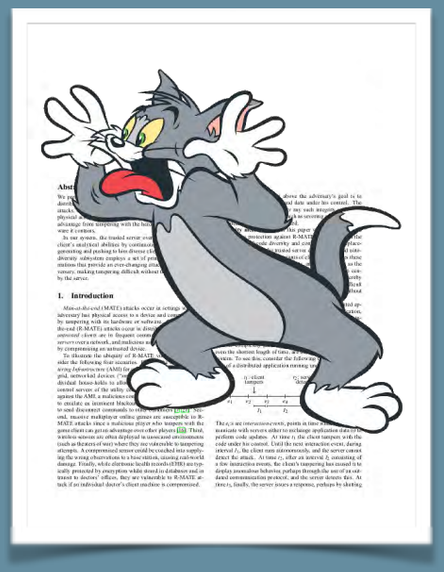
1. **Introduction**

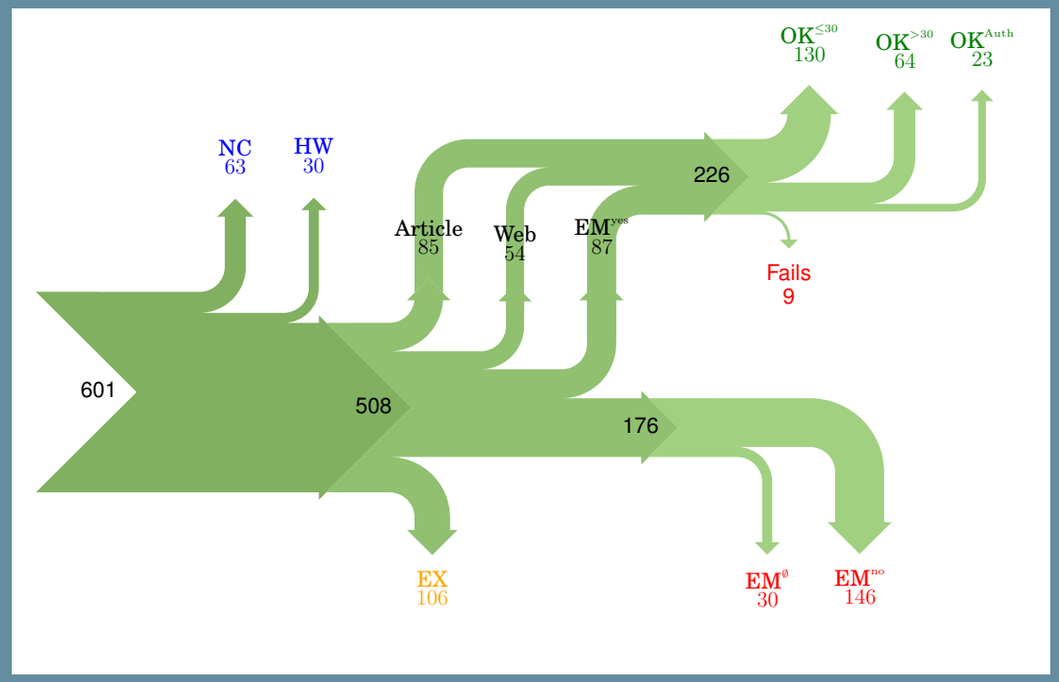


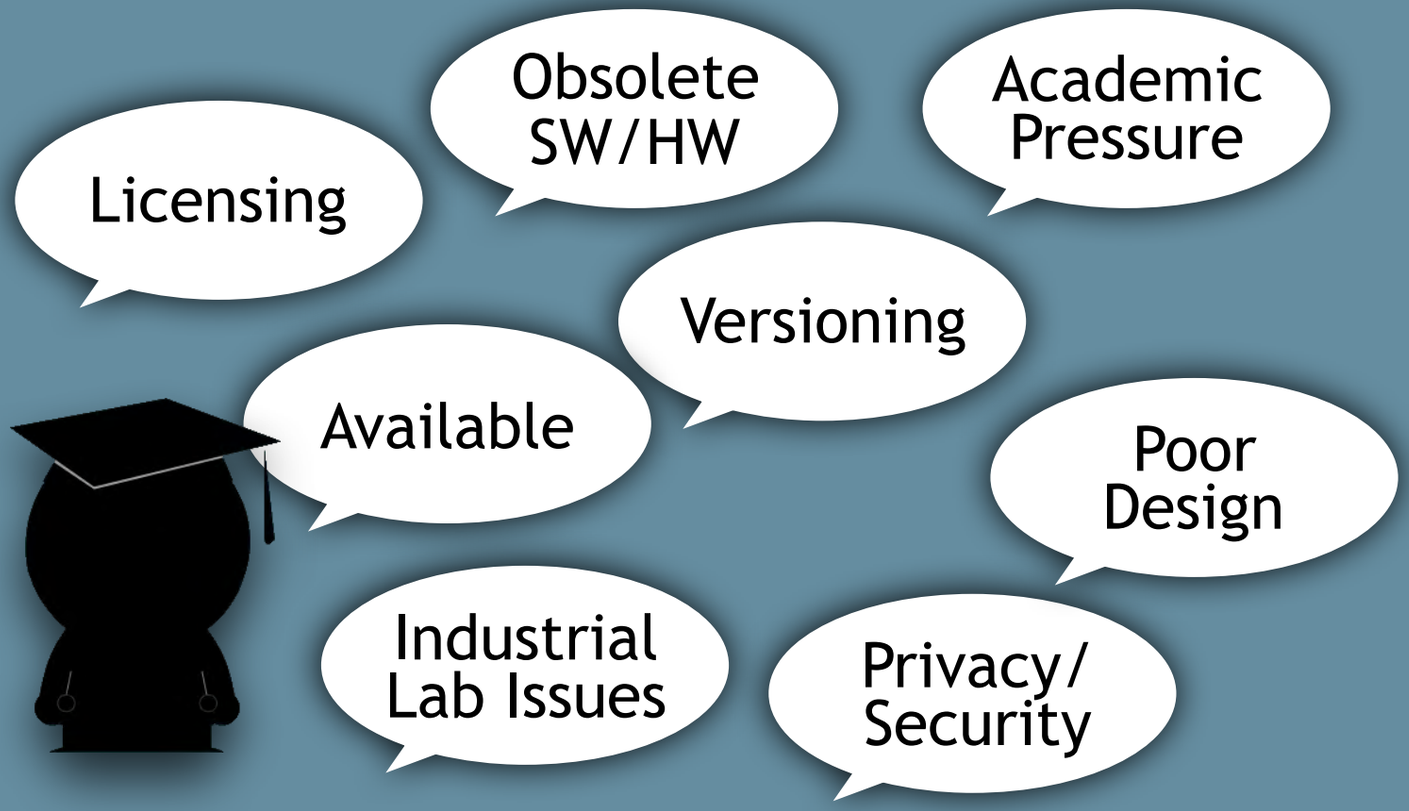


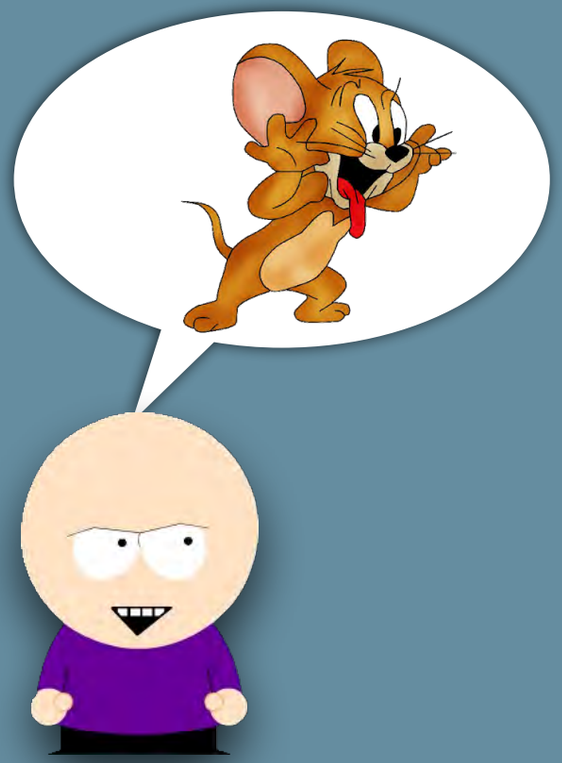

OK≤30
130
OK>30
64
OKAuth
23
NC
63
HW
30
226
Article
85
Web
54
EMyes
87
Fails
9
601
508
176
EX
106
EMϕ
30
EMno
146

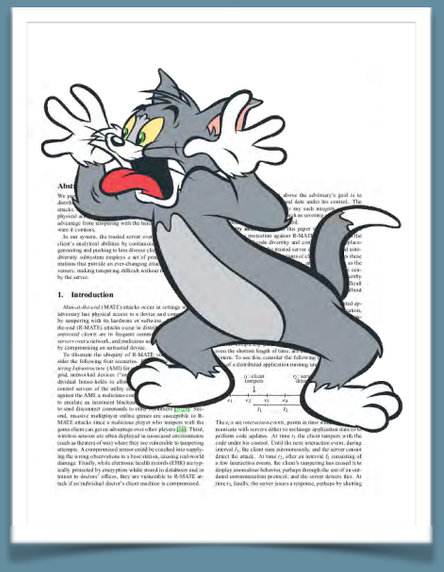


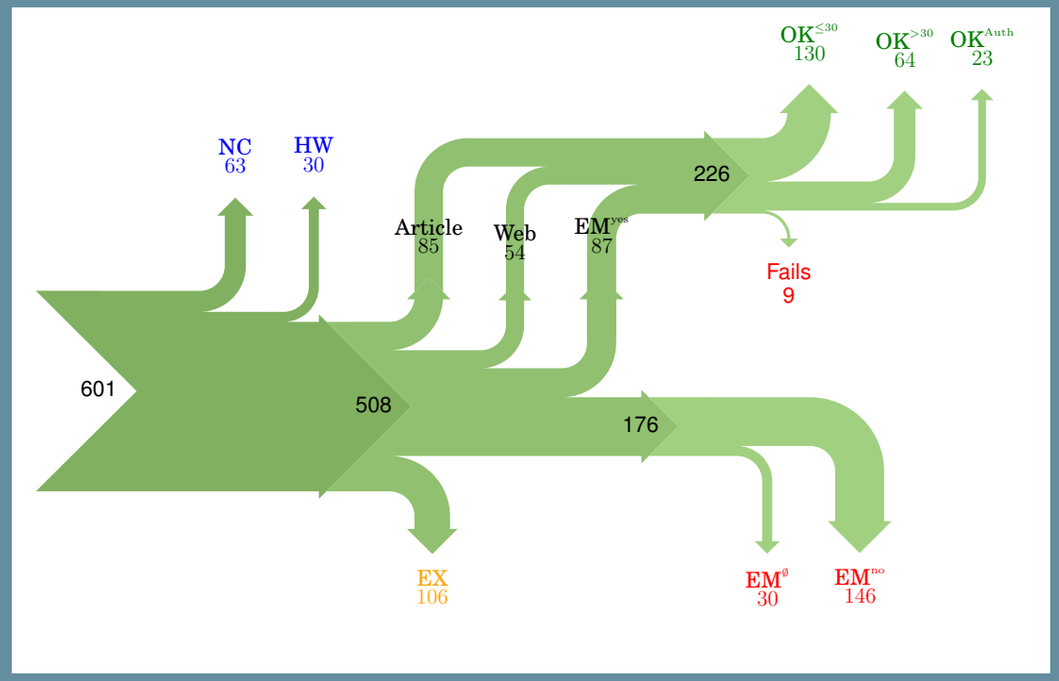


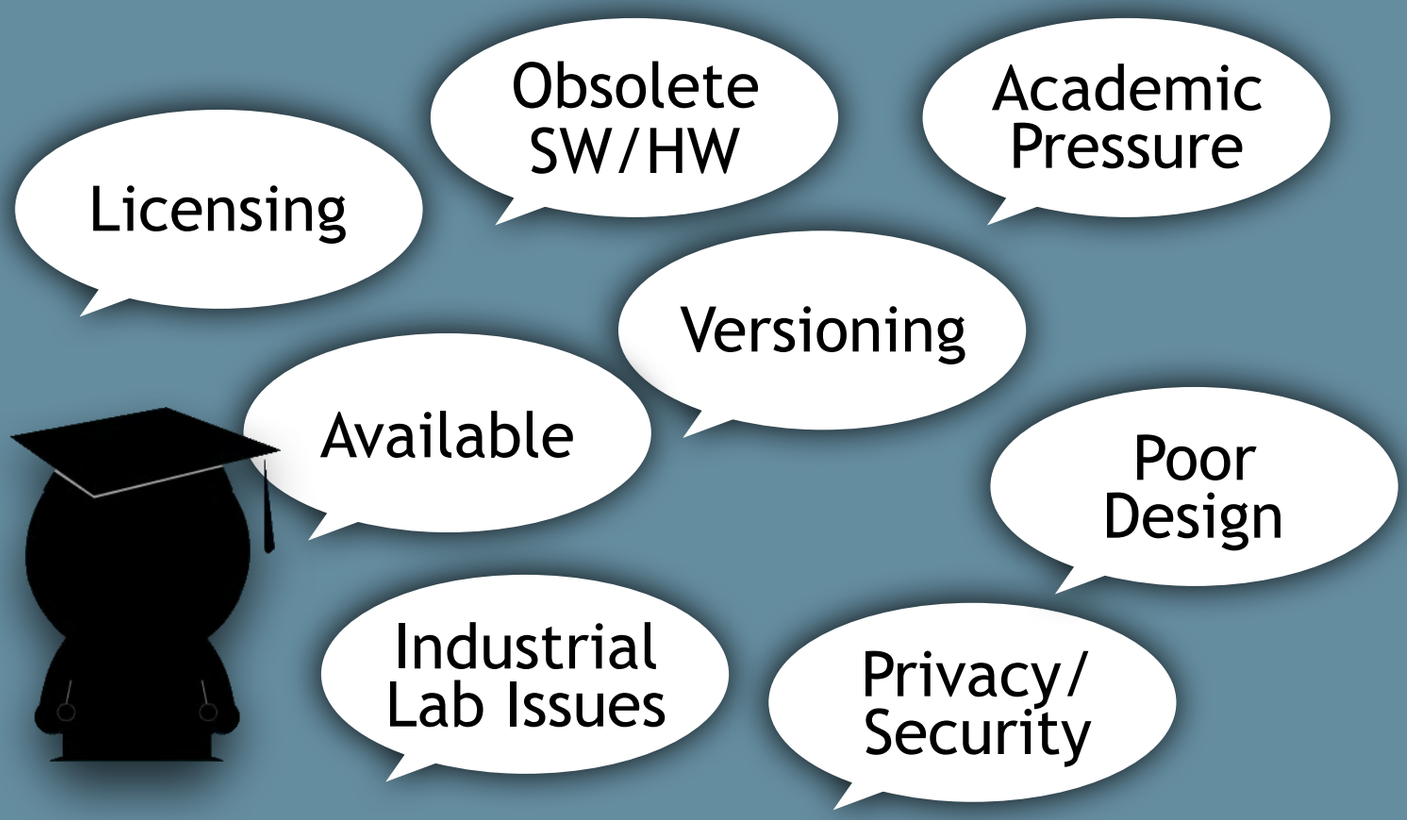

OK≤30
130
OK>30
64
OKAuth
23
NC
63
HW
30
226
Article
85
Web
54
EMyes
87
Fails
9
601
508
176
EX
106
EMϕ
30
EMno
146

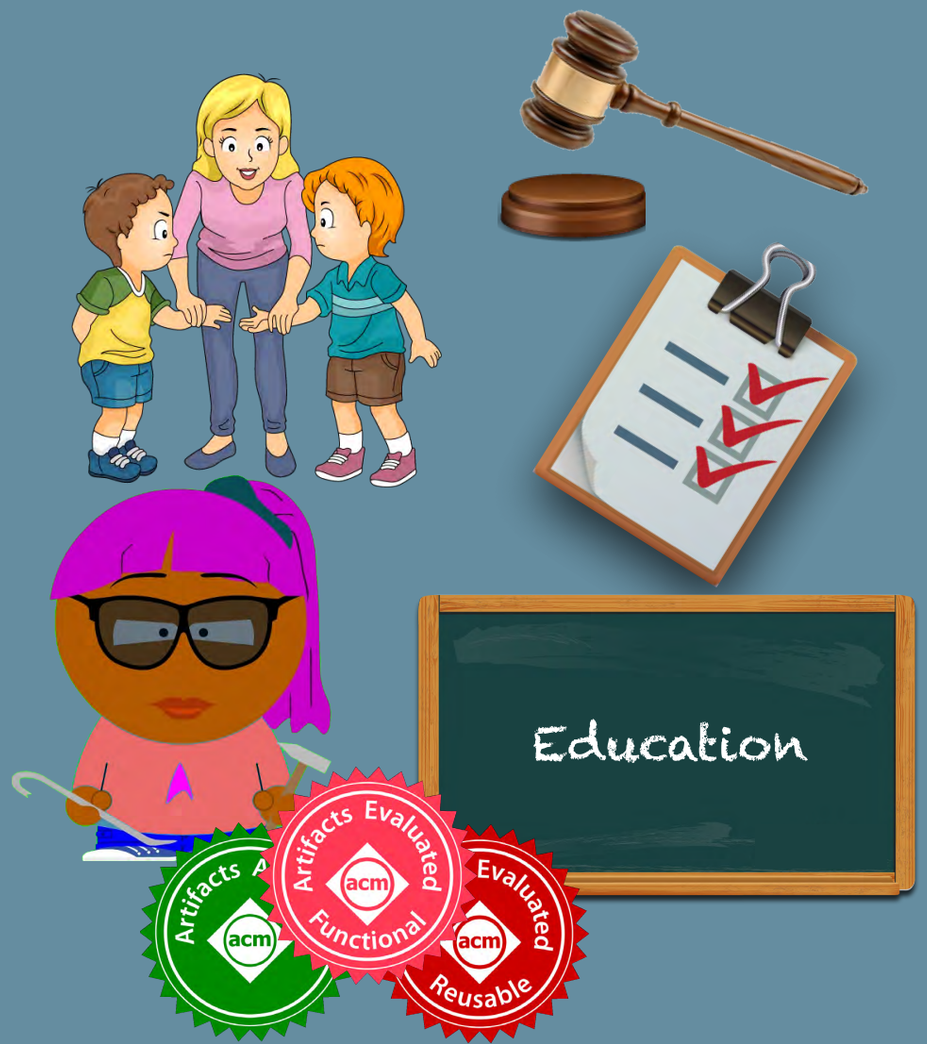

Licensing
Obsolete SW/HW
Academic Pressure
Versioning
Available
Poor Design
Industrial Lab Issues
Privacy/ Security



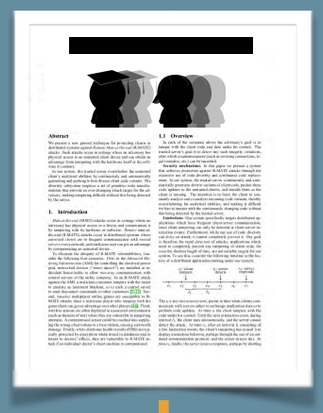



OK≤30
130
OK>30
64
OKAuth
23
NC
63
HW
30
226
Article
85
Web
54
EMyes
87
Fails
9
601
508
176
EX
106
EMϕ
30
EMno
146


Licensing
Obsolete SW/HW
Academic Pressure
Versioning
Available
Poor Design
Industrial Lab Issues
Privacy/ Security


Education
Artifacts Evaluated
acm
Functional
Artifacts
Evaluated
Reusable

FindResearch.org



Find artifacts...

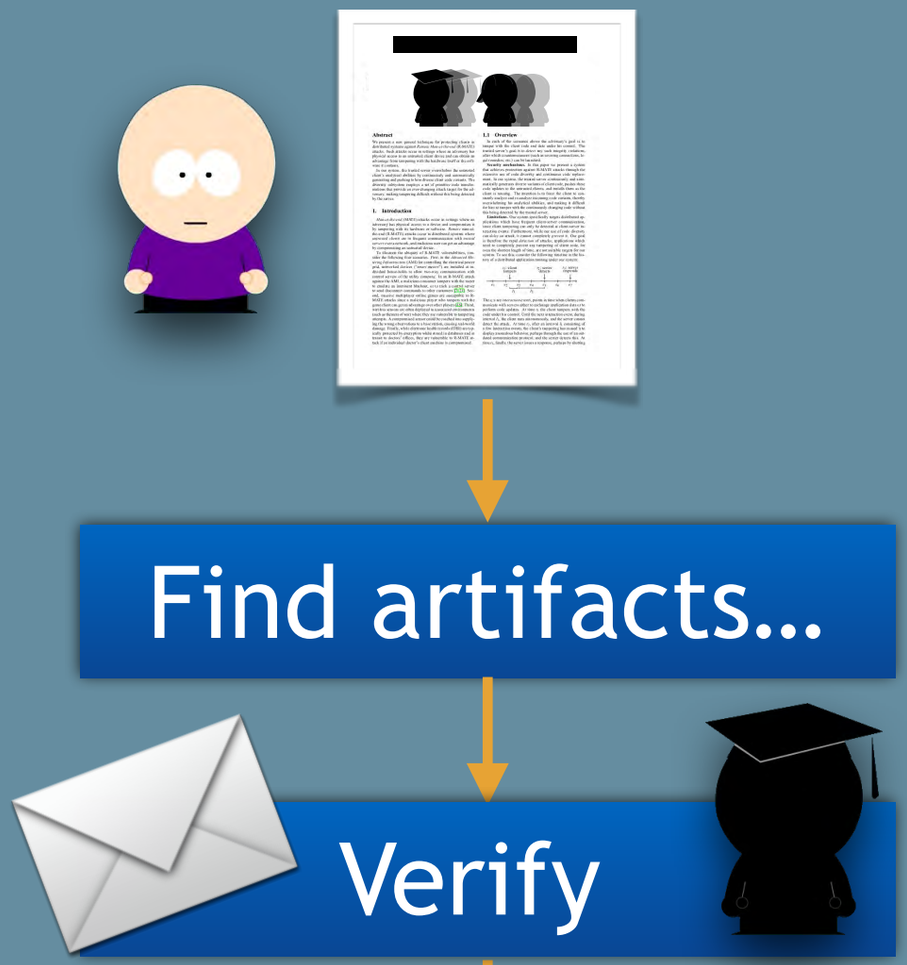
Verify

Publish

search.org FAQ Privacy policy Contact

ACM Programming Language Design and Implementation, PLDI 2014

Title/Authors	Research Artifacts [?]	Details
Optimal inference of fields in row-polymorphic records Axel Simon		Discussion Comments: 0 Verification: Author has not verified information More...
VeriCon: towards verifying controller programs in software-defined networks Thomas Ball, Nikolaj Bjørner, Aaron Gember, Shachar Itzhaky, Aleksandr Karbyshev, Mooly Sagiv, Michael Schapira, Asaf Valadarsky	• http://www.cs.tau.ac.il/~shachar	Discussion Comments: 0 Verification: Authors have not verified informat... More...
Tracelet-based code search in executables Yaniv David, Eran Yahav	• https://github.com/Yanivmd/TRACY	Discussion Comments: 0 Verification: Authors have not verified informat... More...
Modular control-flow integrity Ben Niu, Gang Tan		Discussion Comments: 0 Verification: Authors have not verified informat... More...
Doppio: breaking the browser language barrier John Vilk, Emery D. Berger	• http://www.doppiojvm.org/ Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat... More...
Laws of concurrent programming Tony Hoare		Discussion Comments: 0 Verification: Author has not verified information More...
Test-driven repair of data races in structured parallel programs Rishi Surendran, Raghavan Raman, Swarat Chaudhuri, John M. Mellor-Crummey, Vivek Sarkar	• http://dl.acm.org/ft_gateway.cfm?id=25943... Artifact evaluation badge awarded	Discussion Comments: 0 Verification: Authors have not verified informat... More...



Find artifacts...

Verify

Publish

FindResearch.org

FindResearch.org | Browse | Statistics | Rewards | FAQ | Privacy policy | Contact | Blog

Statistics

Counts ○ Percents

General

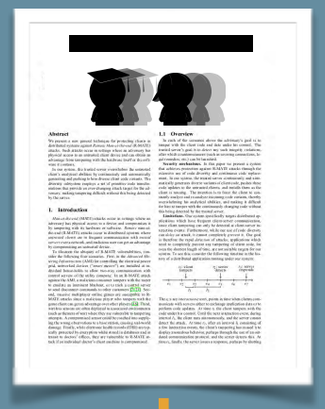
	2017	2016	2015	2014	2013	Total
Conferences	25	52	27	16	1	121
Articles	1,950	5,704	3,131	1,501	30	12,316
Article authors [?]	7,698	21,899	11,811	5,558	144	47,110
Survey emails sent [?]	6,885	20,307	10,860	5,018	117	43,187

All articles

	2017	2016	2015	2014	2013	Total
Article authors with an email address [?]	7,125	20,722	11,208	5,347	122	44,524
Article authors without an email address	573	1,177	603	211	22	2,586
Total article authors	7,698	21,899	11,811	5,558	144	47,110
Articles with at least one author email address	1,948	5,699	3,129	1,500	30	12,306

9.4% of articles are verified

FindResearch.org



Find artifacts...

Verify

Publish

FindResearch.org Browse Statistics Rewards FAQ Privacy policy Contact Blog

Statistics

Counts Percents

General

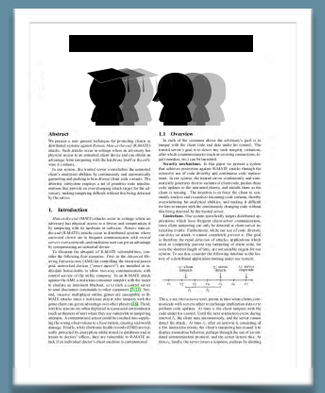
	2017	2016	2015	2014	2013	Total
Conferences	25	52	27	16	1	121
Articles	1,950	5,704	3,131	1,501	30	12,316
Article authors [?]	7,698	21,899	11,811	5,558	144	47,110
Survey emails sent [?]	6,885	20,307	10,860	5,018	117	43,187

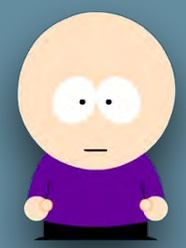
All articles

	2017	2016	2015	2014	2013	Total
Article authors with an email address [?]	7,125	20,722	11,208	5,347	122	44,524
Article authors without an email address	573	1,177	603	211	22	2,586
Total article authors	7,698	21,899	11,811	5,558	144	47,110
Articles with at least one author email address	1,948	5,699	3,129	1,500	30	12,306

9.4% of articles are verified

FindResearch.org





Find artifacts...

Verify

Publish

FindResearch.org Browse Statistics Rewards FAQ Privacy policy Contact Blog

Statistics

Counts Percents

General

	2017	2016	2015	2014	2013	Total
Conferences	25	52	27	16	1	121
Articles	1,950	5,704	3,131	1,501	30	12,316
Article authors [?]	7,698	21,899	11,811	5,558	144	47,110
Survey emails sent [?]	6,885	20,307	10,860	5,018	117	43,187

All articles

	2017	2016	2015	2014	2013	Total
Article authors with an email address [?]	7,125	20,722	11,208	5,347	122	44,524
Article authors without an email address	573	1,177	603	211	22	2,586
Total article authors	7,698	21,899	11,811	5,558	144	47,110
Articles with at least one author email address	1,948	5,699	3,129	1,500	30	12,306

10.2% of articles are verified with

Rewards	Risks

Rewards	Risks
Credibility: Colleagues may trust your work more when they can try it.	

Rewards	Risks
Credibility: Colleagues may trust your work more when they can try it. **Visibility**: Colleagues may notice your work when they can build on the code.	

Rewards	Risks
Credibility: Colleagues may trust your work more when they can try it. **Visibility**: Colleagues may notice your work when they can build on the code. **Correctness**: Colleagues may help you find and correct problems.	

Rewards	Risks
Credibility: Colleagues may trust your work more when they can try it. **Visibility**: Colleagues may notice your work when they can build on the code. **Correctness**: Colleagues may help you find and correct problems. **Collaboration**: Colleagues may want to help you extend your system.	

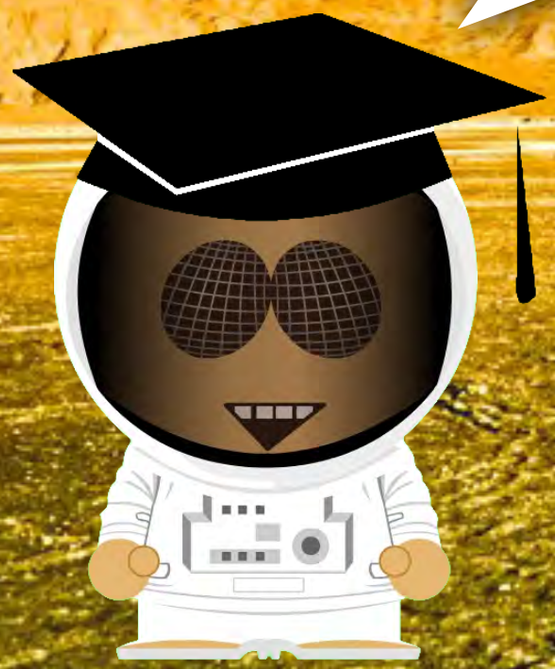
Rewards	Risks
Credibility: Colleagues may trust your work more when they can try it. **Visibility**: Colleagues may notice your work when they can build on the code. **Correctness**: Colleagues may help you find and correct problems. **Collaboration**: Colleagues may want to help you extend your system.	**Credibility**: Colleagues may find bugs and think your results are wrong.

Rewards	Risks
Credibility: Colleagues may trust your work more when they can try it.	**Credibility**: Colleagues may find bugs and think your results are wrong.
Visibility: Colleagues may notice your work when they can build on the code.	**Return-on-investment**: Colleagues may ignore your code in spite of your efforts to share it.
Correctness: Colleagues may help you find and correct problems.	
Collaboration: Colleagues may want to help you extend your system.	

2nd Law of Artifact Sharing

The root of the scientific transparency problem is sociological, not technological: we do not share solid artifacts because there is little professional glory to be gained from doing so.

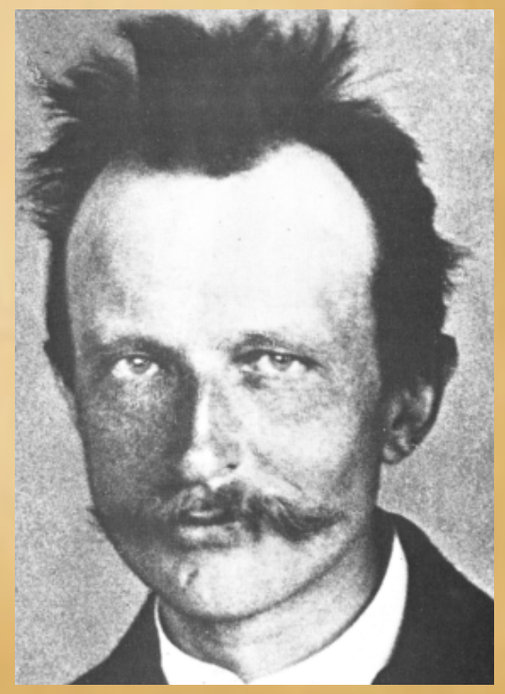
Can you believe, back in the 21st century, scientists would make up excuses why they shouldn’t have to share their research artifacts!



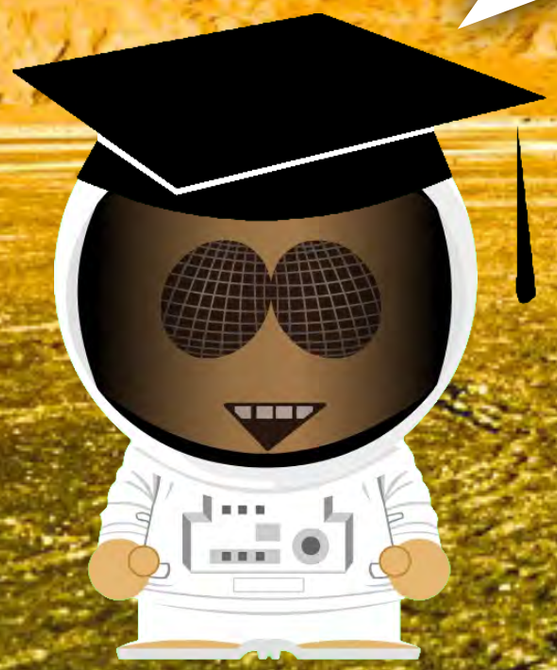
1st Law of Artifact Sharing

(Corollary to Max Planck's Quip)

Scientific transparency advances one funeral at a time.



1. Propose a **Research Methods** course!
2. Help us populate FindResearch.org!
3. Prepare to share, use **checklists**!
4. Insist on **Sharing Contracts**!



Thank you!