Sharing Specifications

Christian Collberg Todd Proebsting

Department of Computer Science
University of Arizona

http://repeatability.cs.arizona.edu

Opening Gambit

Study

Proposal

Future Work

[redacted]

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.

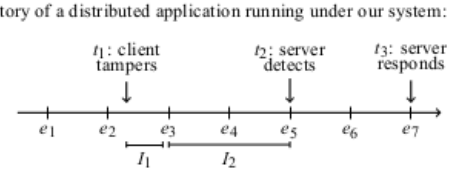
To illustrate the ubiquity of R-MATE vulnerabilities, consider the following four scenarios. First, in the *Advanced Metering Infrastructure* (AMI) for controlling the electrical power grid, networked devices ("*smart meters*") are installed at individual house-holds to allow two-way communication with 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

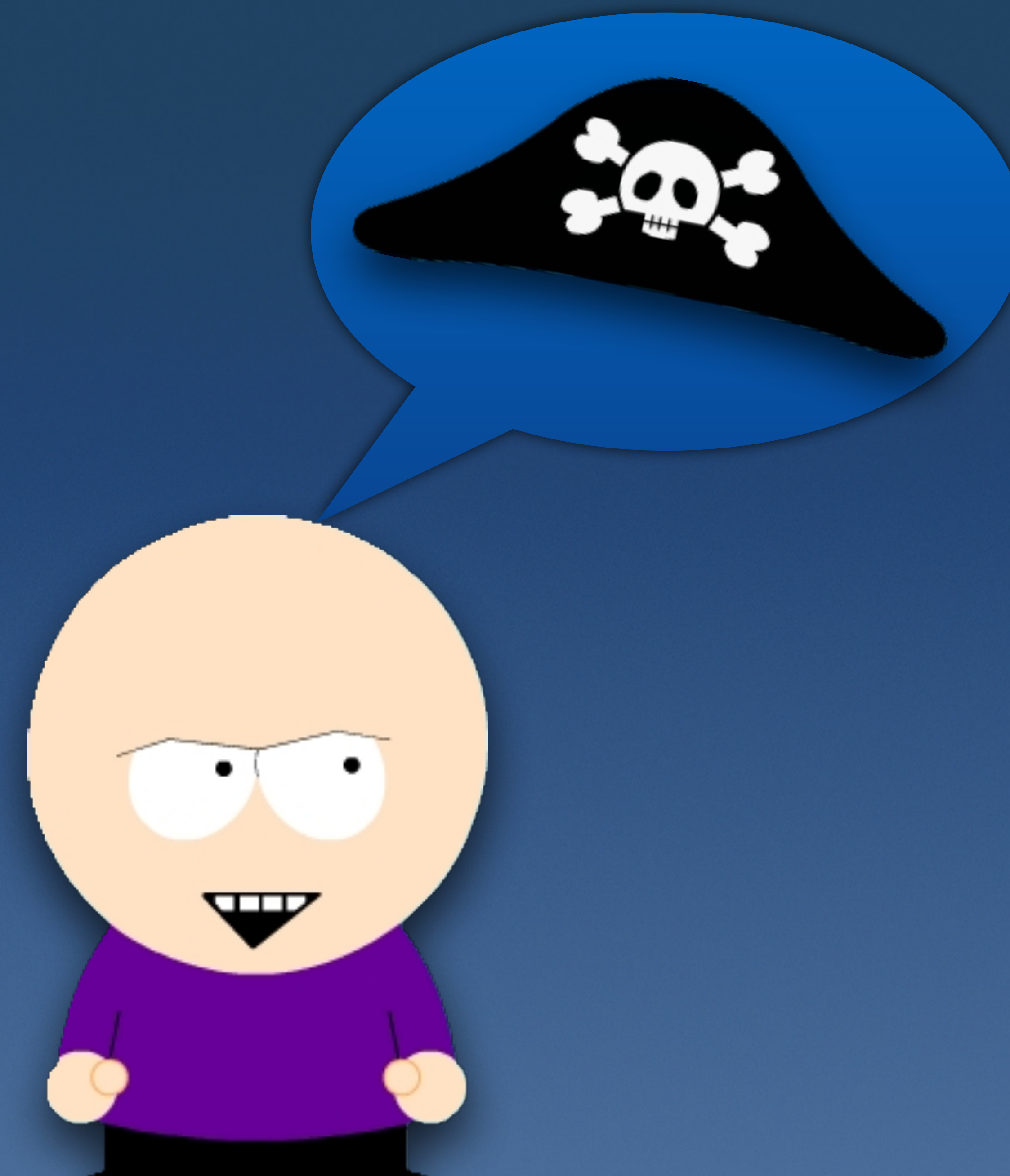
In each of the scenarios above the adversary's goal is to tamper with the client code and data under his control. The trusted server's goal is to *detect* any such integrity violations, after which countermeasures (such as severing connections, legal remedies, etc.) can be launched.

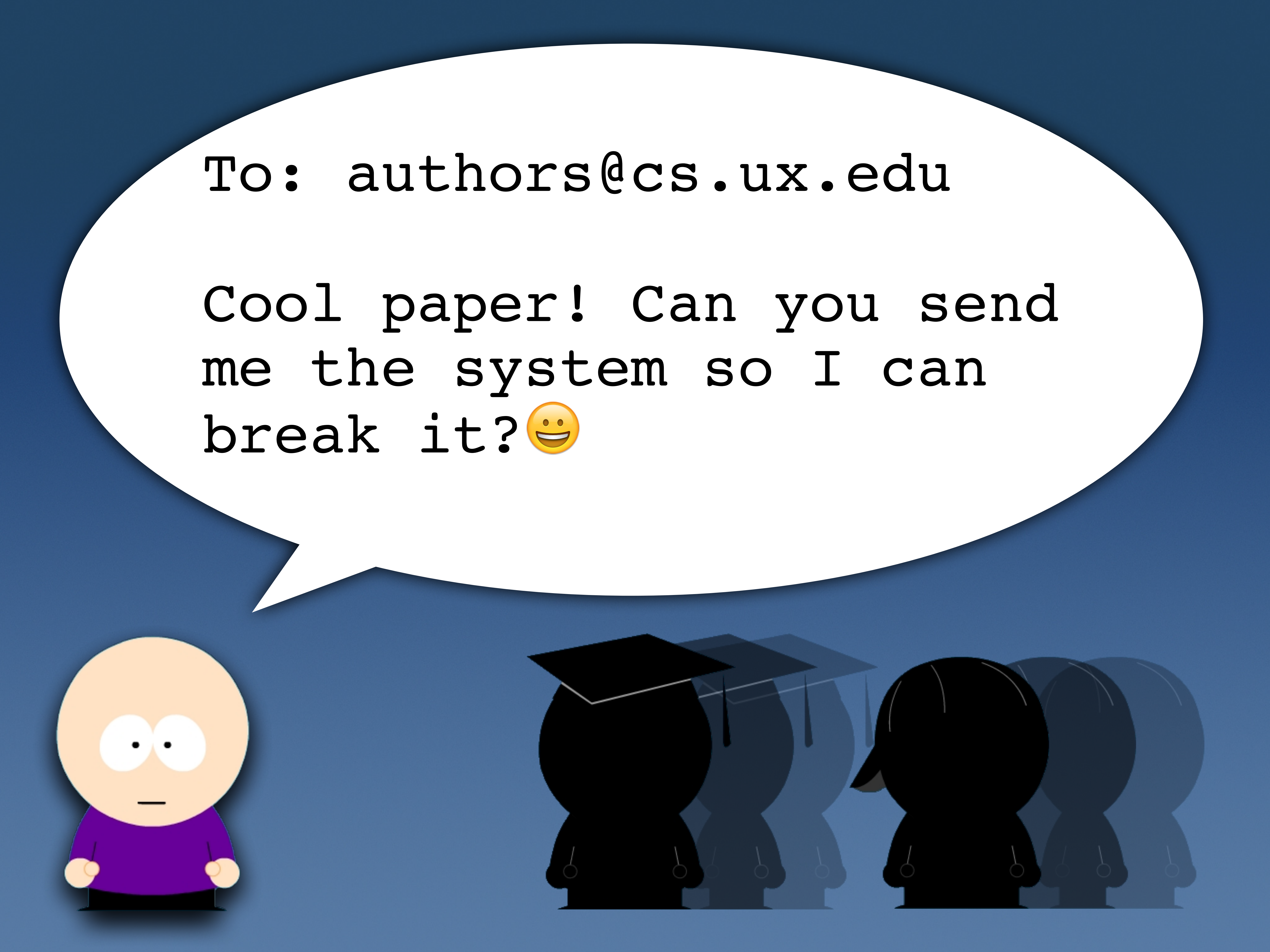
Security mechanisms. In this paper we present a system that achieves protection against R-MATE attacks through the extensive use of code diversity and continuous code replacement. In our system, the trusted server continuously and automatically generates diverse variants of client code, pushes these 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:

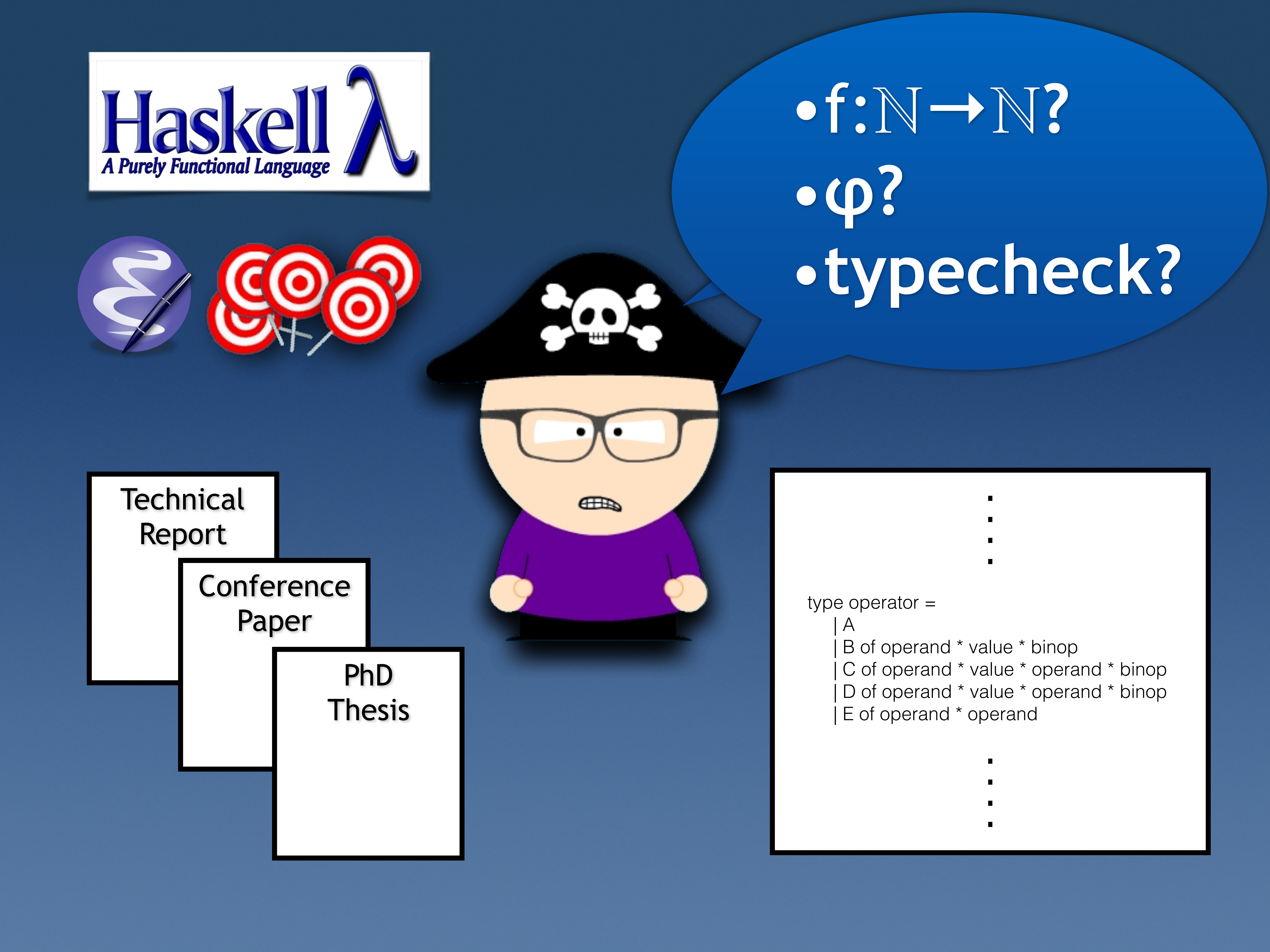


The e_i:s are *interaction events*, points in time when clients communicate with servers either to exchange application data or to perform code updates. At time t_1 the client tampers with the code under his control. Until the next interaction event, during interval I_1, the client runs autonomously, 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

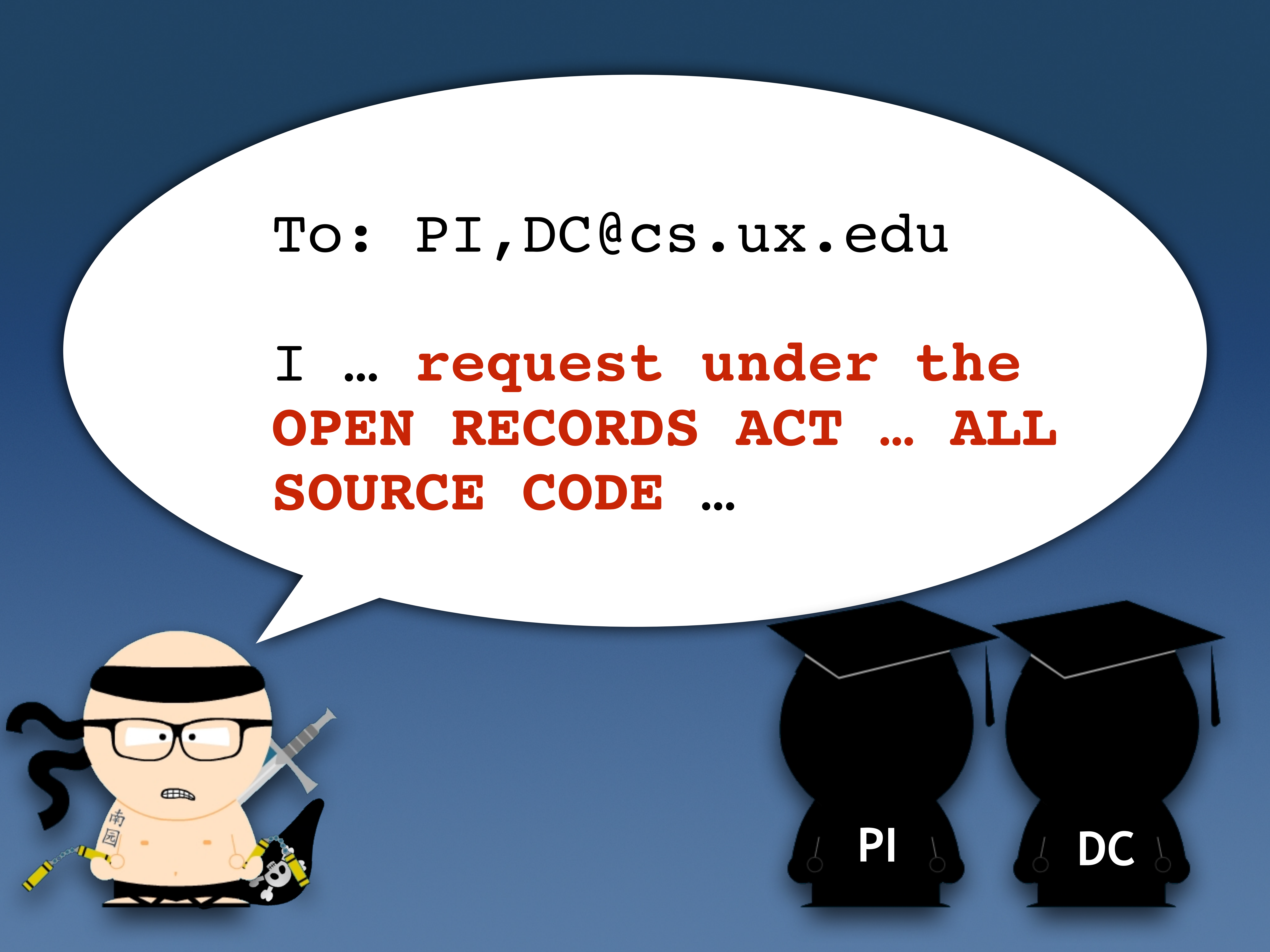




To: authors@cs.ux.edu
Cool paper! Can you send me the system so I can break it?😀

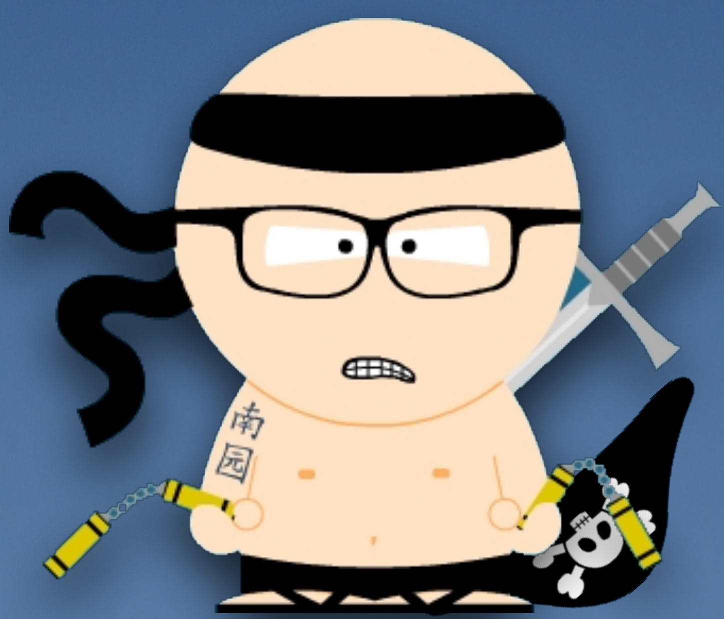


Haskell
A Purely Functional Language
•f:N→N?
•φ?
•typecheck?
Technical Report
Conference Paper
PhD Thesis
type operator =
| A
| B of operand * value * binop
| C of operand * value * operand * binop
| D of operand * value * operand * binop
| E of operand * operand


To: PI,DC@cs.ux.edu
I … request under the OPEN RECORDS ACT … ALL SOURCE CODE …
PI
DC

From: legal@cs.ux.edu

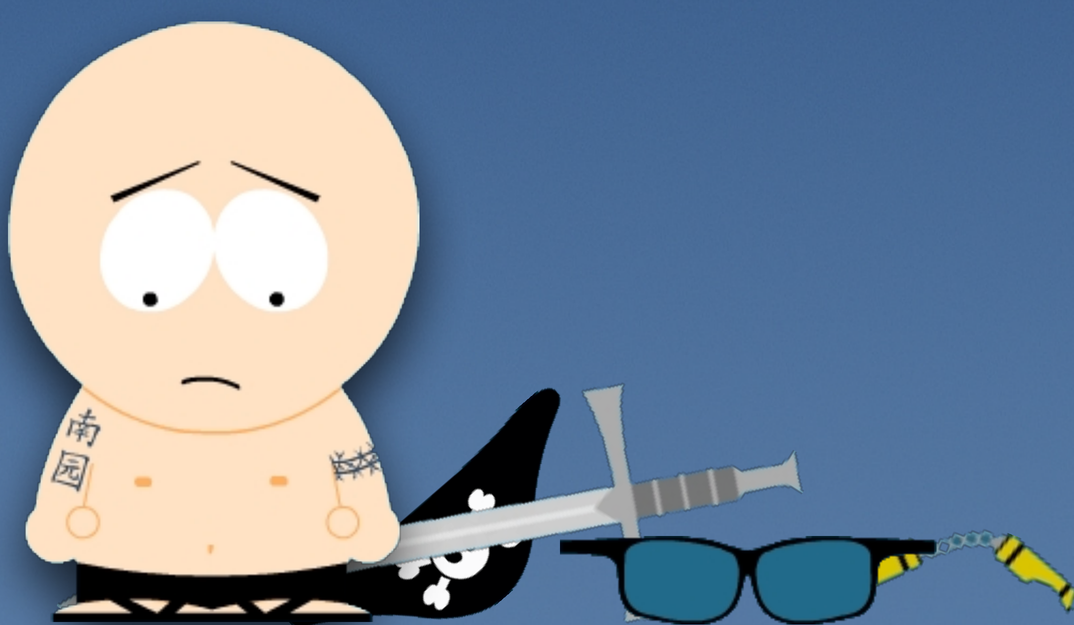
… to the extent such records may exist, **they will not be produced pursuant to ORA**.




… we estimate 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.



Study

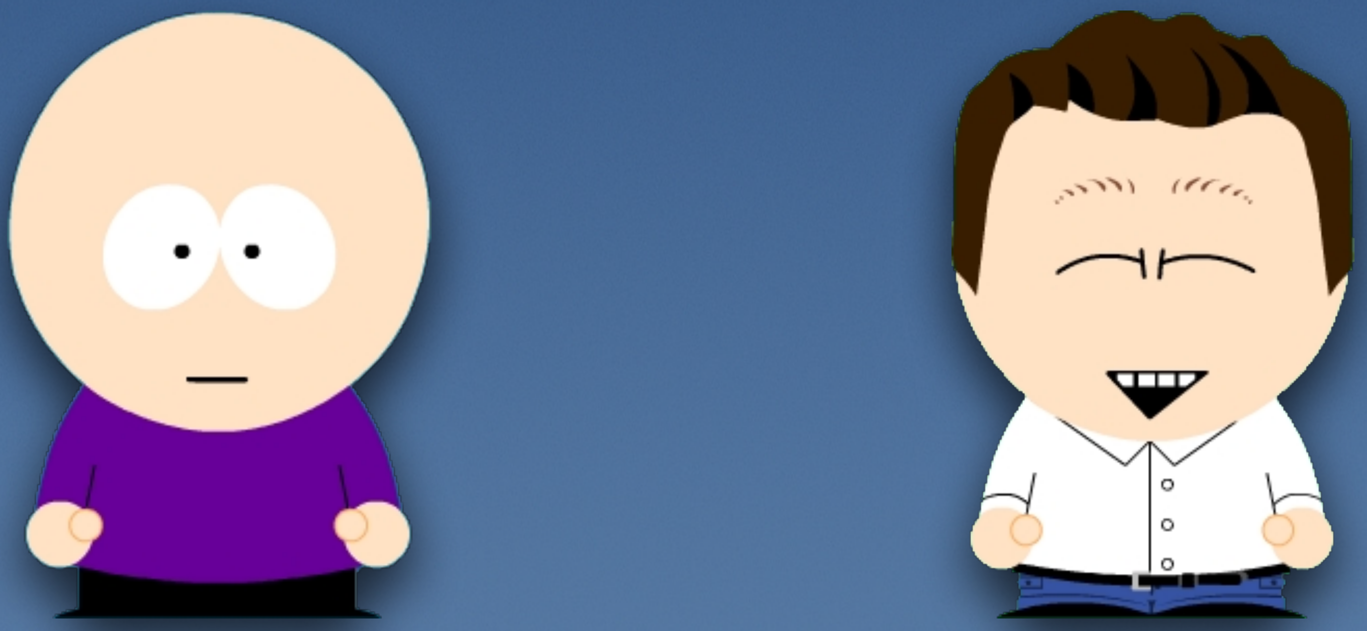
Repeatability

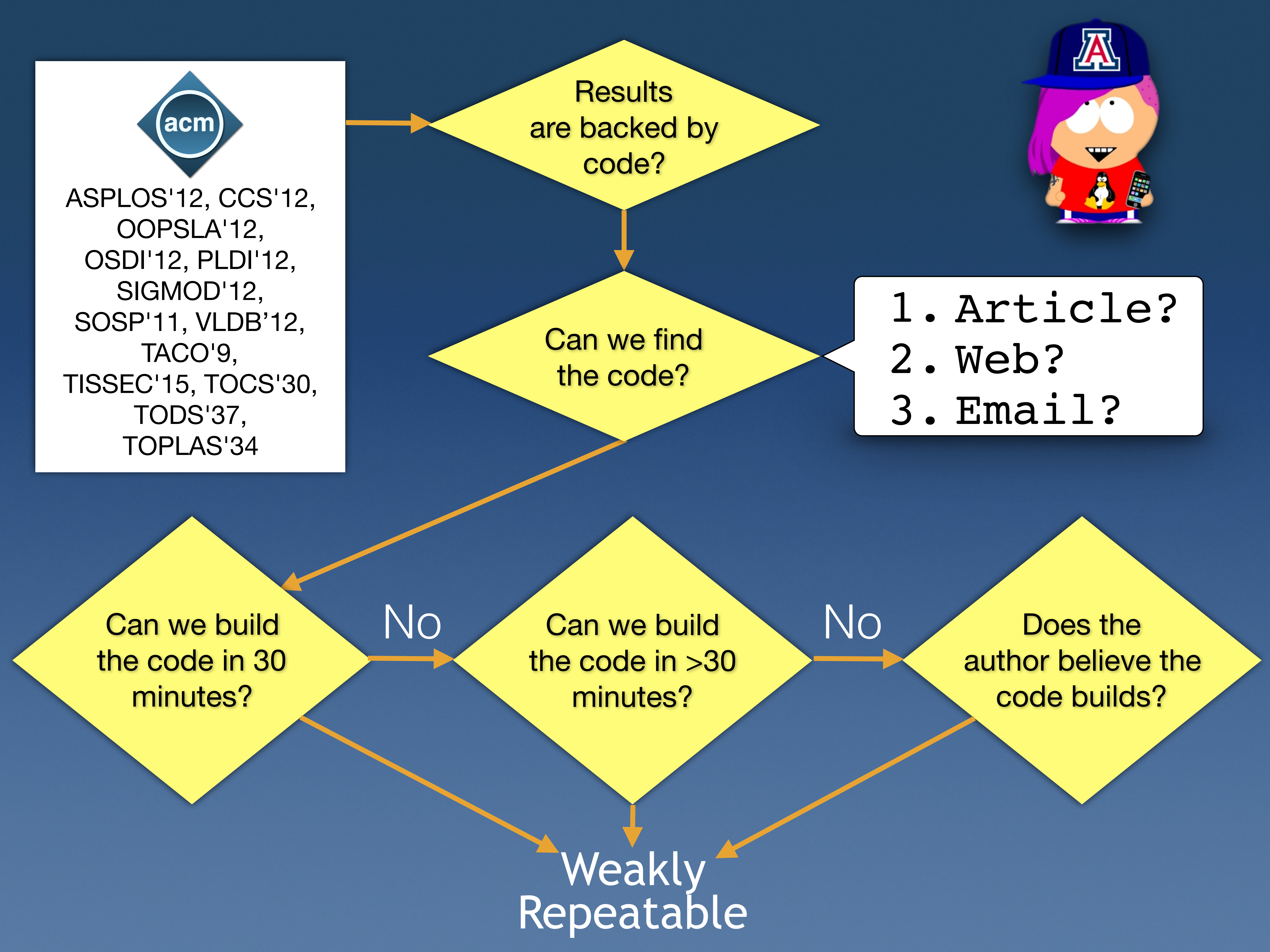
[T]he ability to re-run the exact same experiment with the same method on the same or similar system and obtain the same or very similar result.

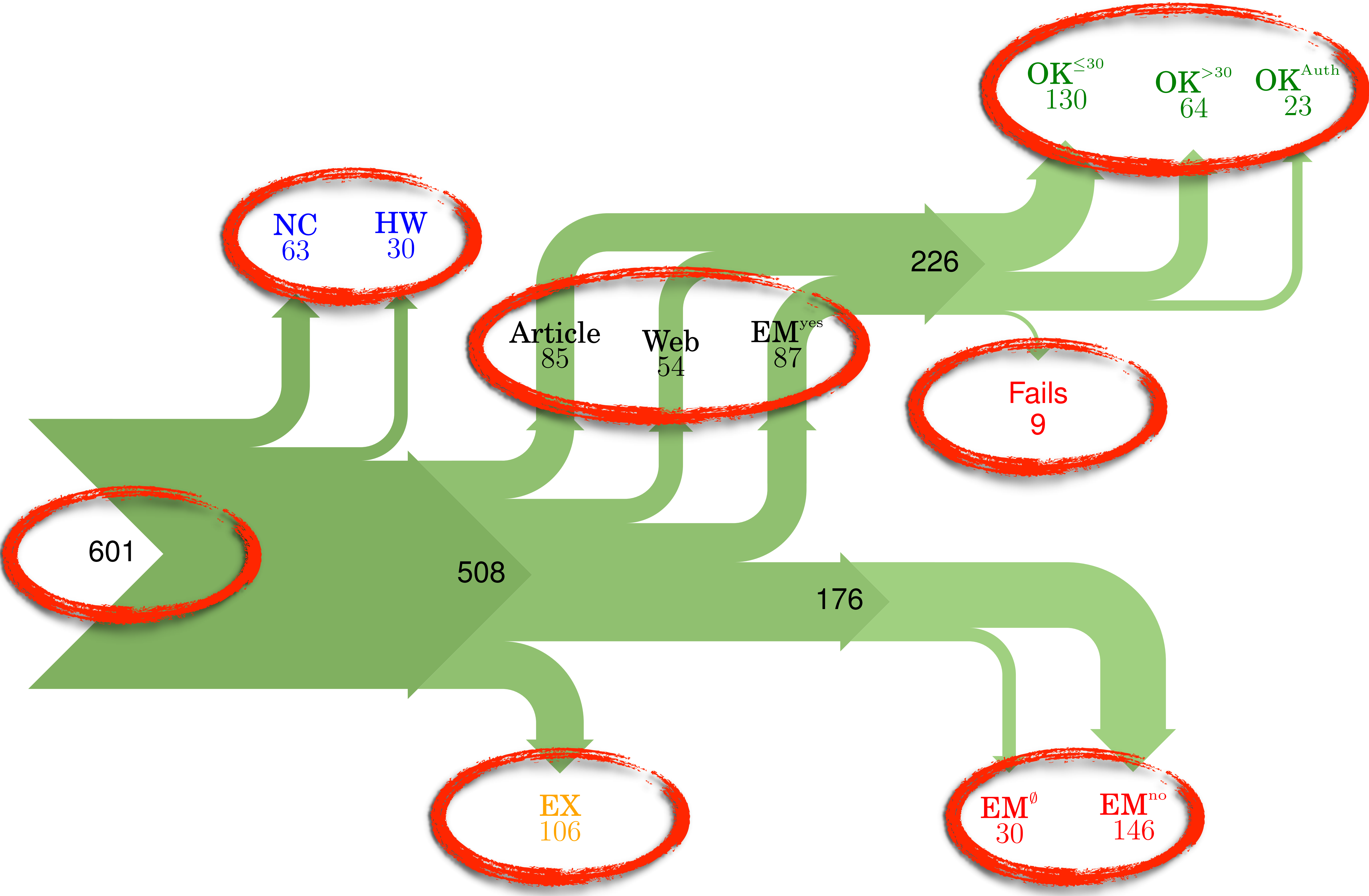
Vitek, Kalibera: R3 – Repeatability, Reproducibility and Rigor

Weak Repeatability

Do authors make the source code used to create the results in their article available, and will it build?



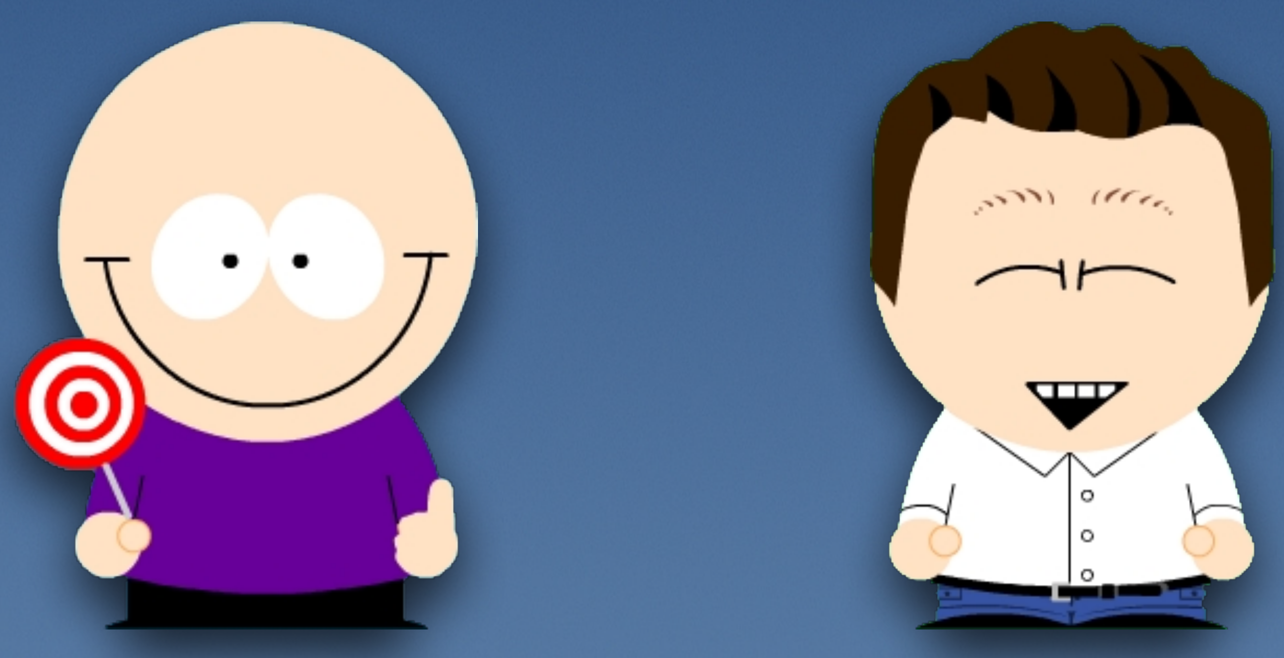

acm
ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12, TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34
Results are backed by code?
Can we find the code?
1. Article?
2. Web?
3. Email?
Can we build the code in 30 minutes?
No
Can we build the code in >30 minutes?
No
Does the author believe the code builds?
Weakly Repeatable



601
508
NC 63
HW 30
Article 85
Web 54
EM^{yes} 87
226
$OK^{\leq 30}$ 130
$OK^{>30}$ 64
OK^{Auth} 23
Fails 9
176
EX 106
$EM^{\emptyset}$ 30
EM^{no} 146

Reasons for not Sharing?

The email responses we received were pleasant, accommodating, and apologetic if code could not be provided.



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 …

Available Soon

The code was **never intended to be released** so is not in any shape for general use.

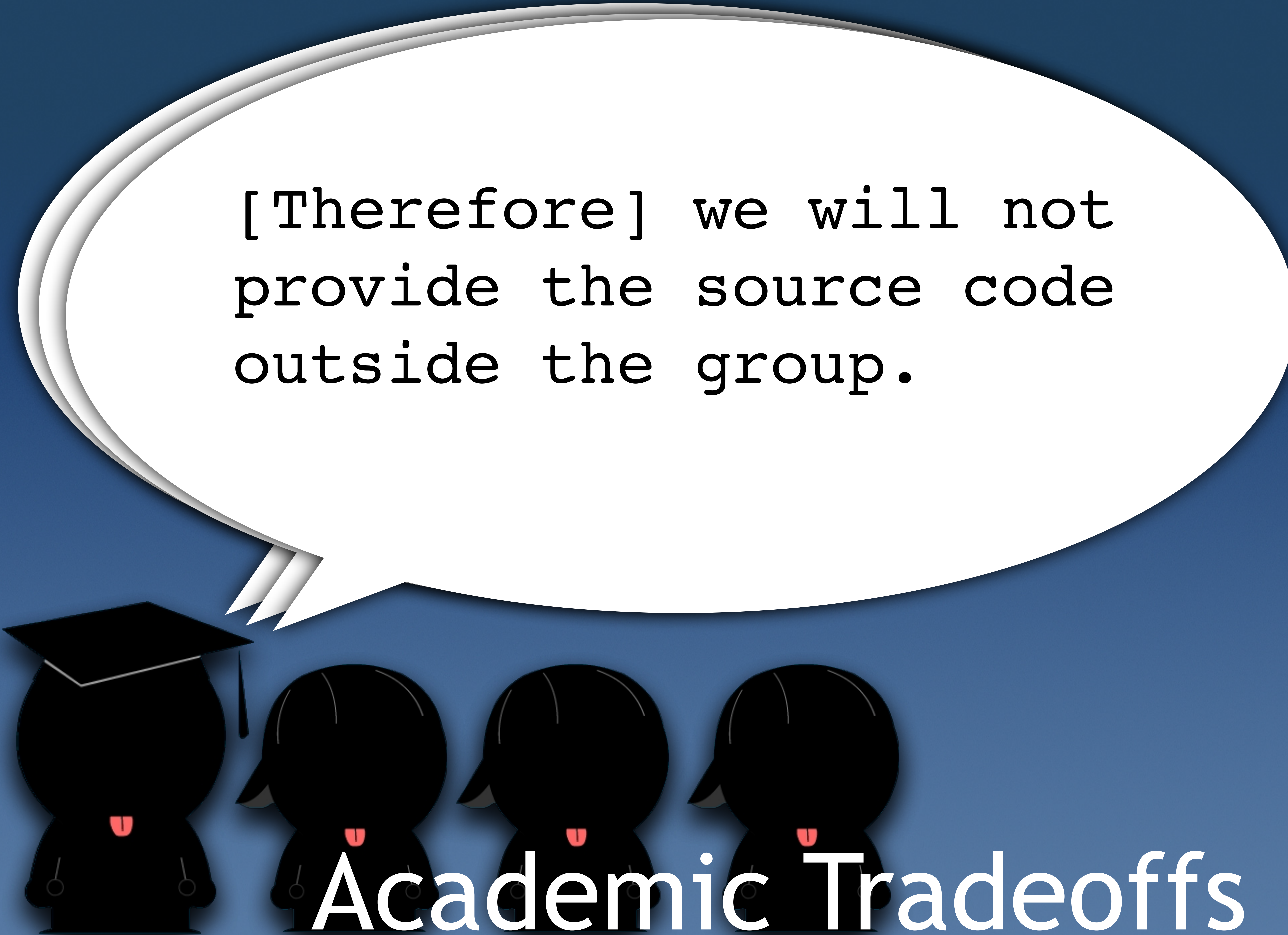
No Intention to Share

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

Personnel Issues

Lost Code


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



Academic Tradeoffs

Industrial Lab 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.

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

Privacy/Security

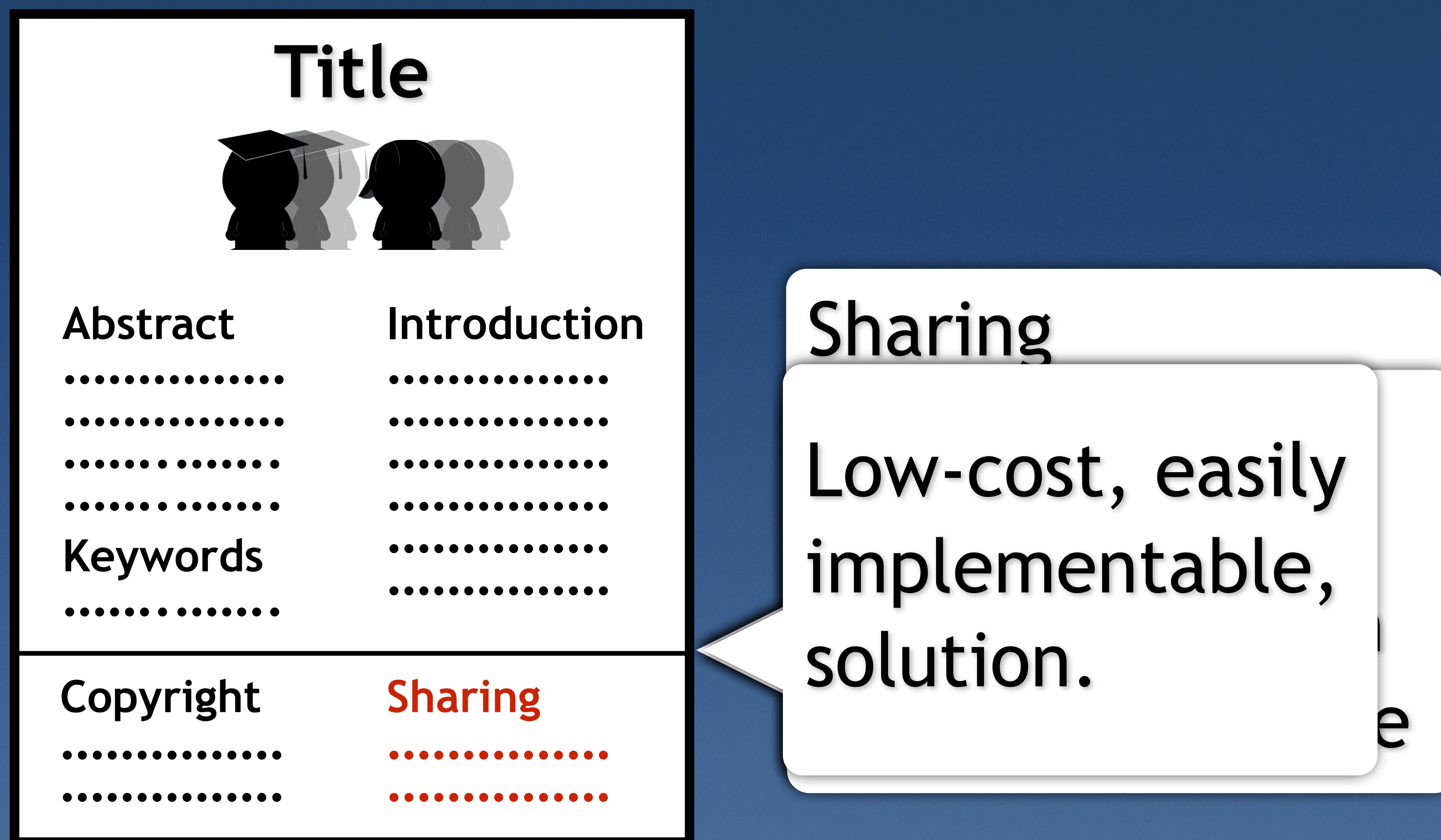
Proposal

Three Modest Proposals

1. Funding agencies should encourage researchers to request additional funds for repeatability engineering

2. Agencies should conduct random audits to ensure that research artifacts are shared in accordance with what was promised in the grant application

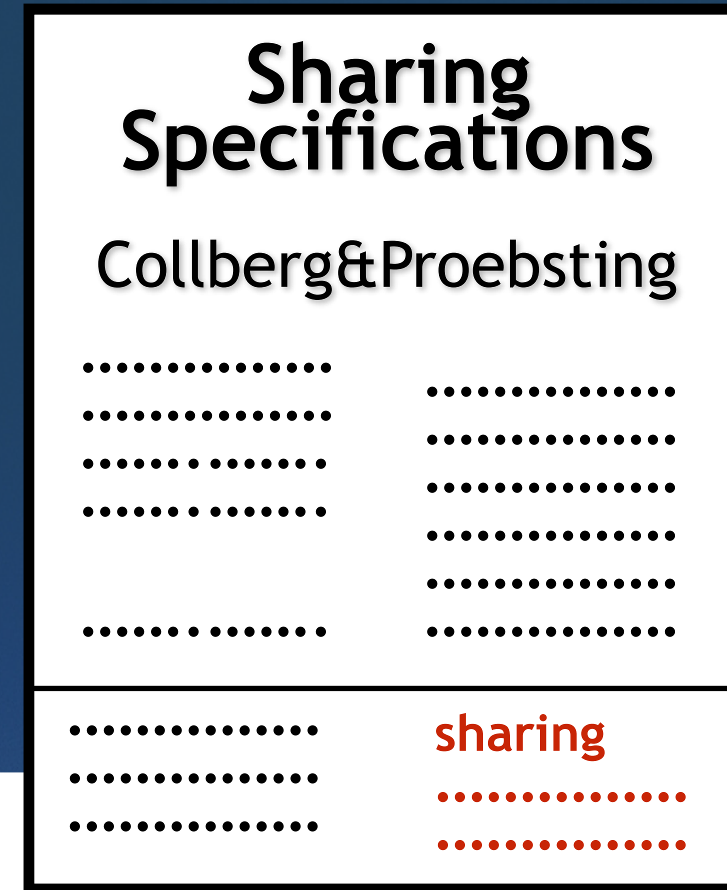
Three Modest Proposals



3. Publishers should require articles to contain a sharing contract specifying the level of repeatability to which its authors will commit

Location	• email address and/or web site
Resource	• **types**: code, data, media, documentation • **availability**: no access, access, NDA access • **expense**: free, non-free, free for academics • **distribution form**: source, binary, service • **expiration date** • **license** • **comment**
Support	• **kinds**: resolve installation issues, fix bugs, upgrade to new language and operating system versions, port to new environments, improve performance, add features • **expense**: free, non-free, free for academics • **expiration date**

Sharing Contract

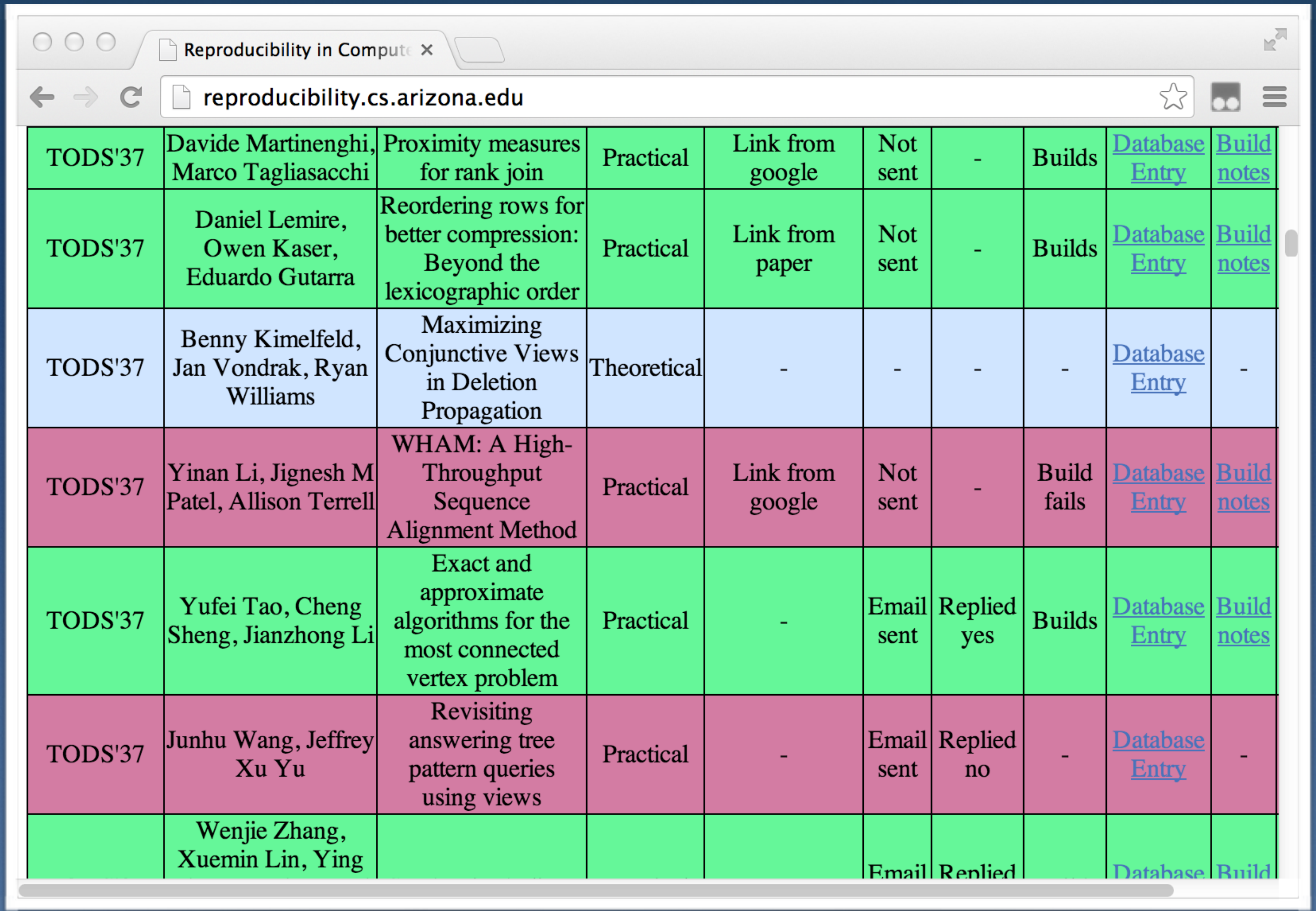



```
sharing
  repeatability.cs.arizona.edu;
  collberg@gmail.com;
  code: access,free,source;
  data: access,free,source,"sanitized";
  support: installation,bug fixes,free,
  2015-12-31;
```

Discussion and Future Work

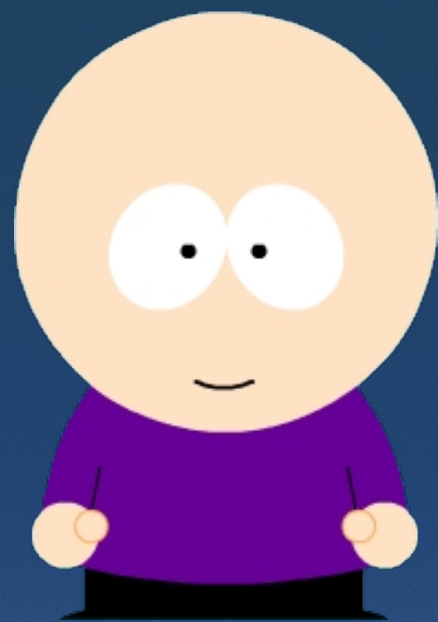
repeatability.cs.arizona.edu

Technical Report


Reproducibility in Comput
reproducibility.cs.arizona.edu

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

To appear in The Communication of the ACM



1. Demanding everyone to share code always is unrealistic.

2. Sharing specifications are a low-cost alternative that can be implemented now.

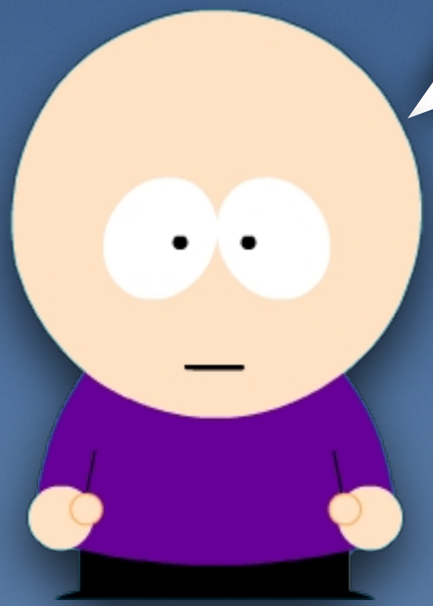
3. We believe sharing specifications will be an incentive to authors to produce solid computational artifacts.

Longitudinal Study

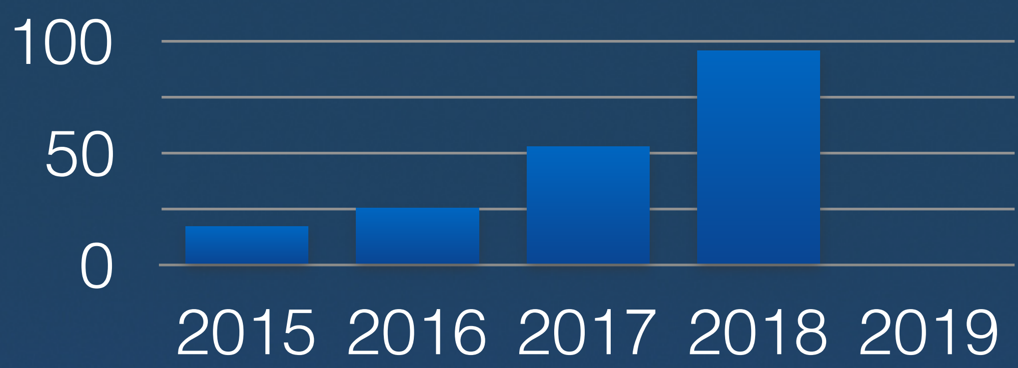
To: author@cs.ux.edu

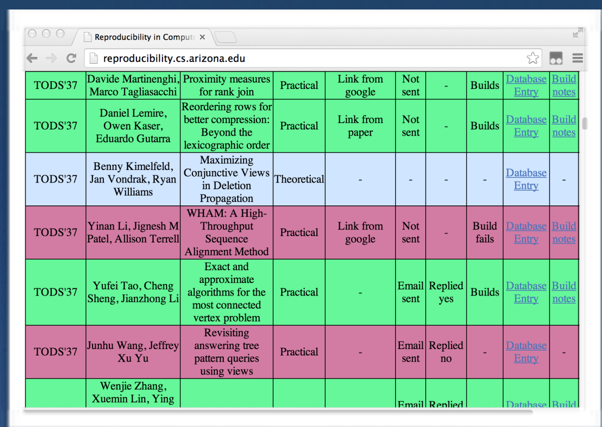
Congrats on your new paper!

- **Will you share?**
- **Under what license?**
- **URL to code/data?**



LARGE NUMBER OF CONFERENCES OVER 5 YEARS



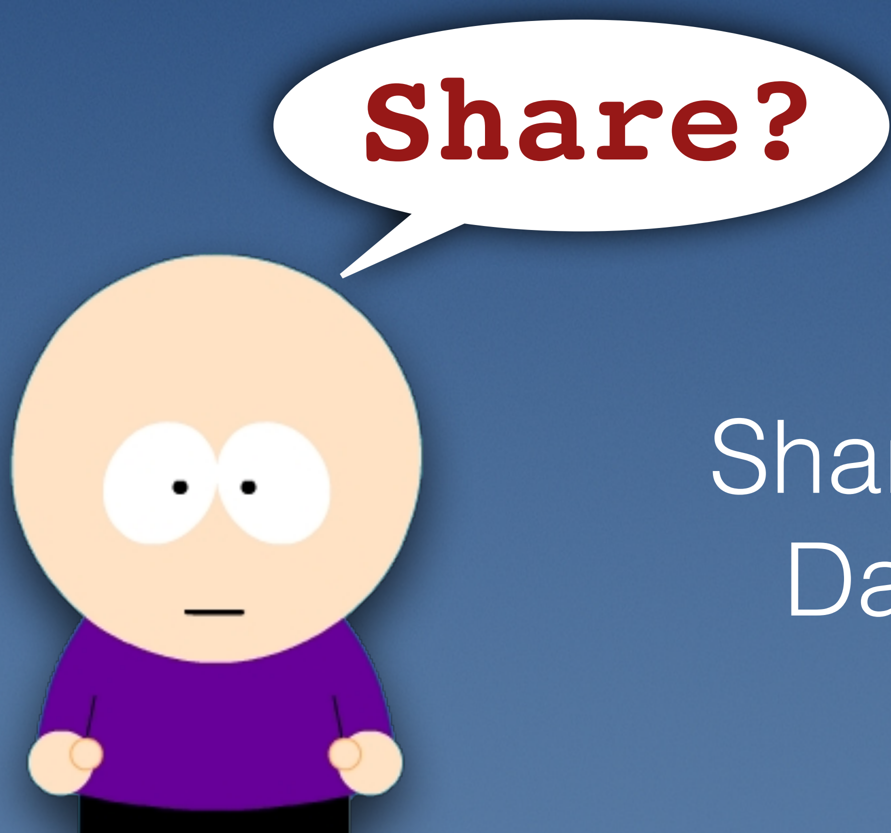


Reproducibility in Comput

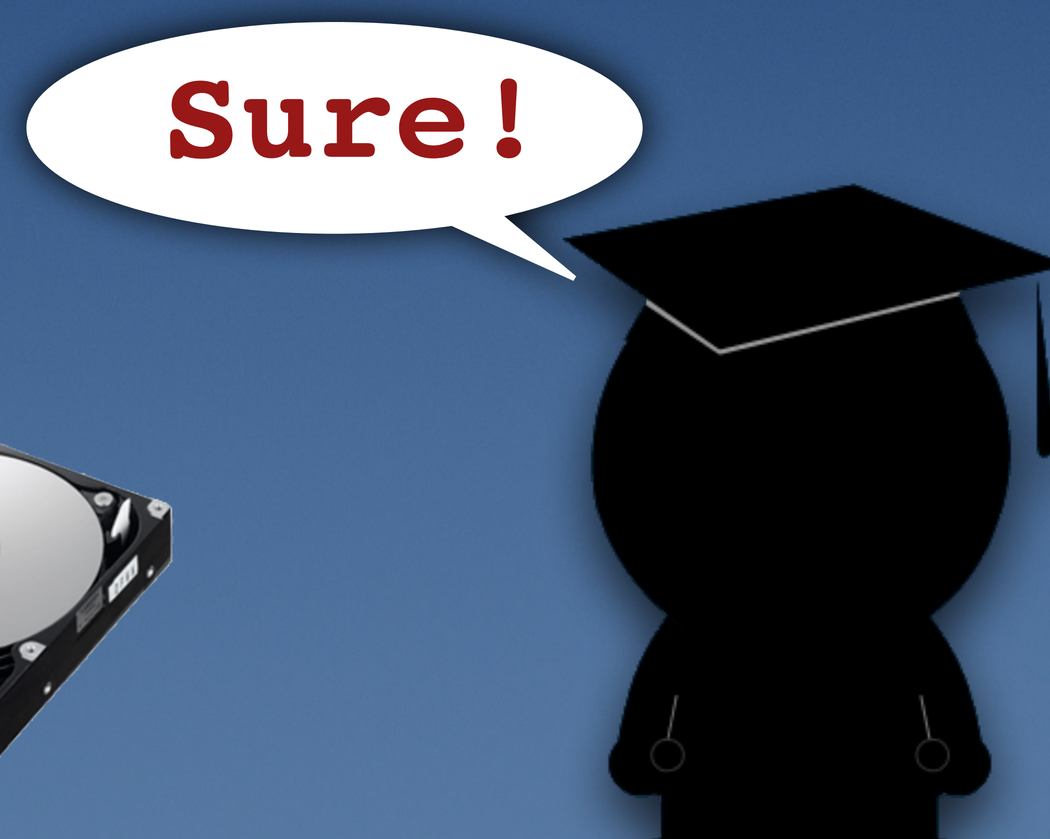
reproducibility.cs.arizona.edu

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

1. Data for reproducibility research
2. Trending data for funding agencies
3. Directory of research artifacts
4. Motivating researchers to share



Sharing Data Database



Questions?