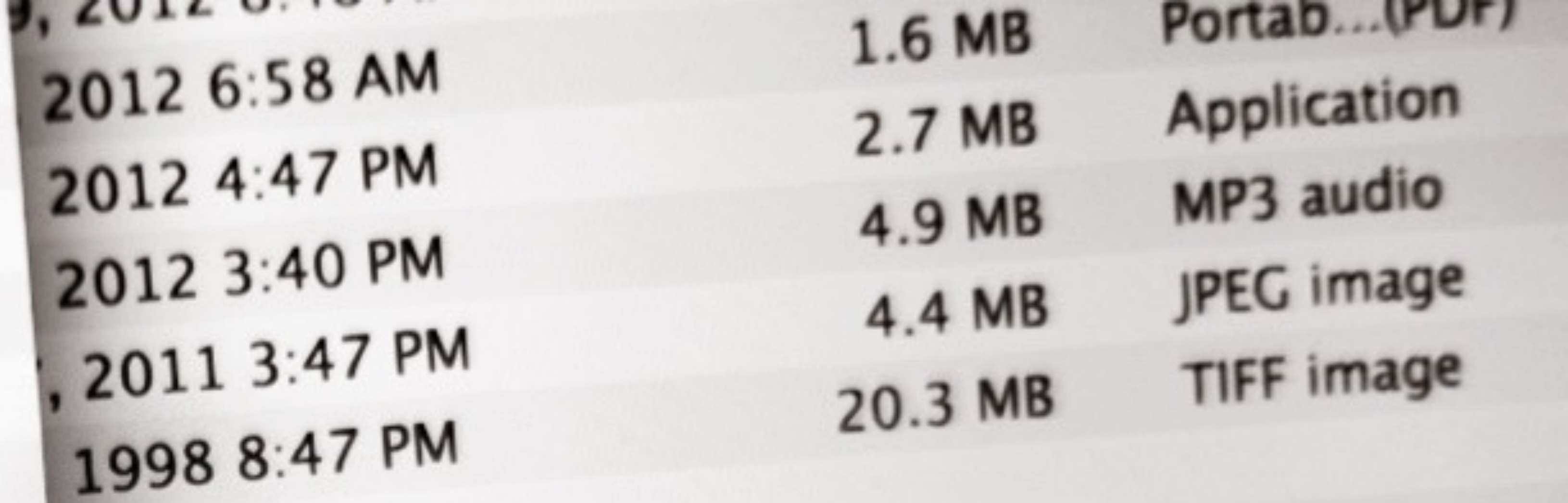


```
, 2012 6:58 AM       1.6 MB    Portab...(PDF)
2012 6:58 AM         2.7 MB    Application
2012 4:47 PM         4.9 MB    MP3 audio
2012 3:40 PM         4.4 MB    JPEG image
, 2011 3:47 PM      20.3 MB    TIFF image
1998 8:47 PM
```

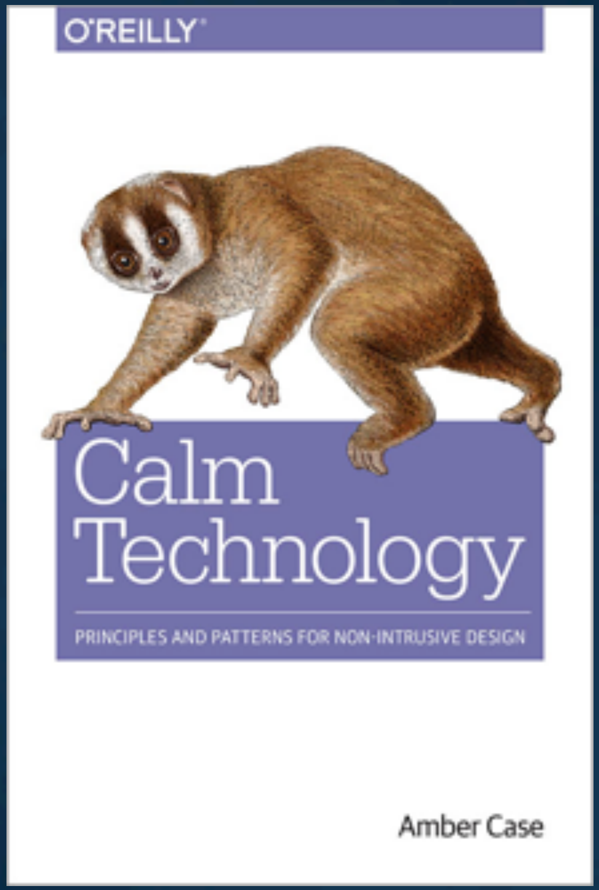
the limits of verification

Daniel Jackson (CSAIL, MIT)
Mandana Vaziri (IBM)

FSE · November 17, 2016

this file includes only Daniel's slides

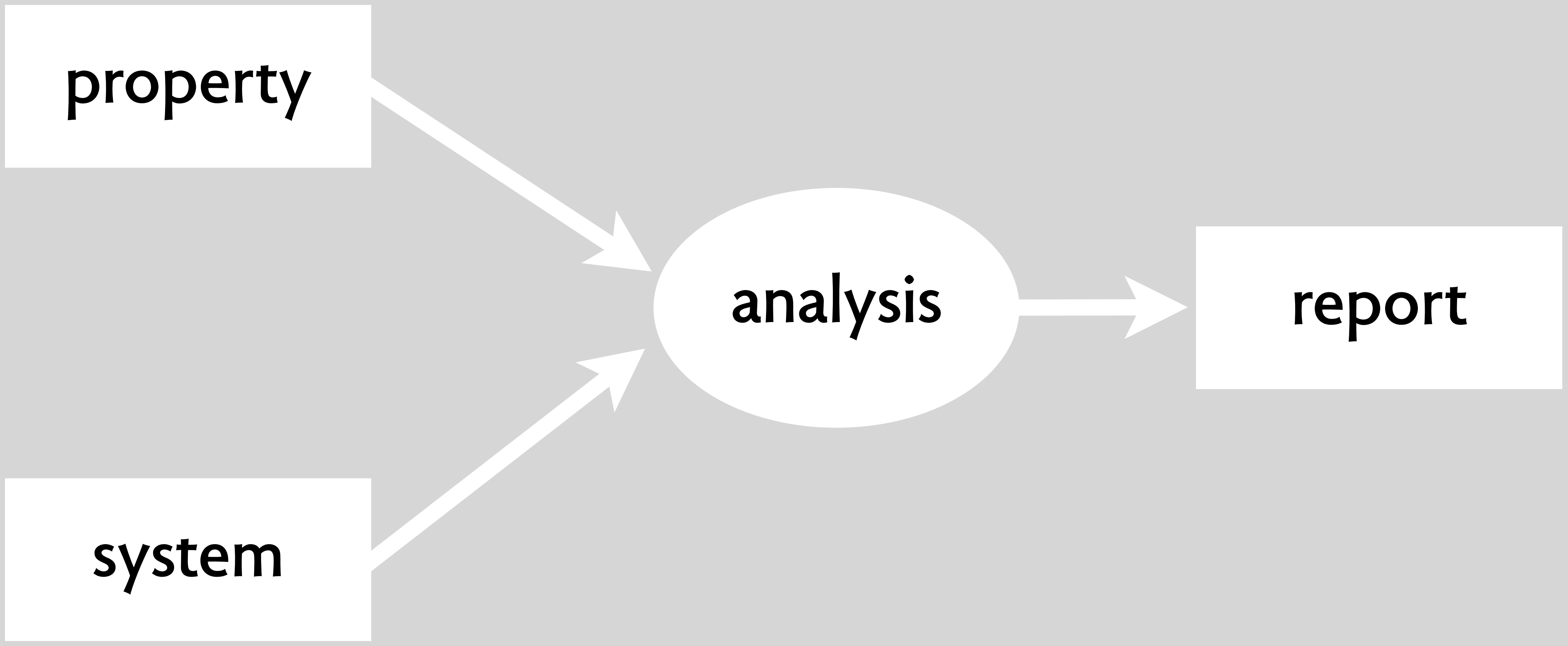
dependability isn't everything



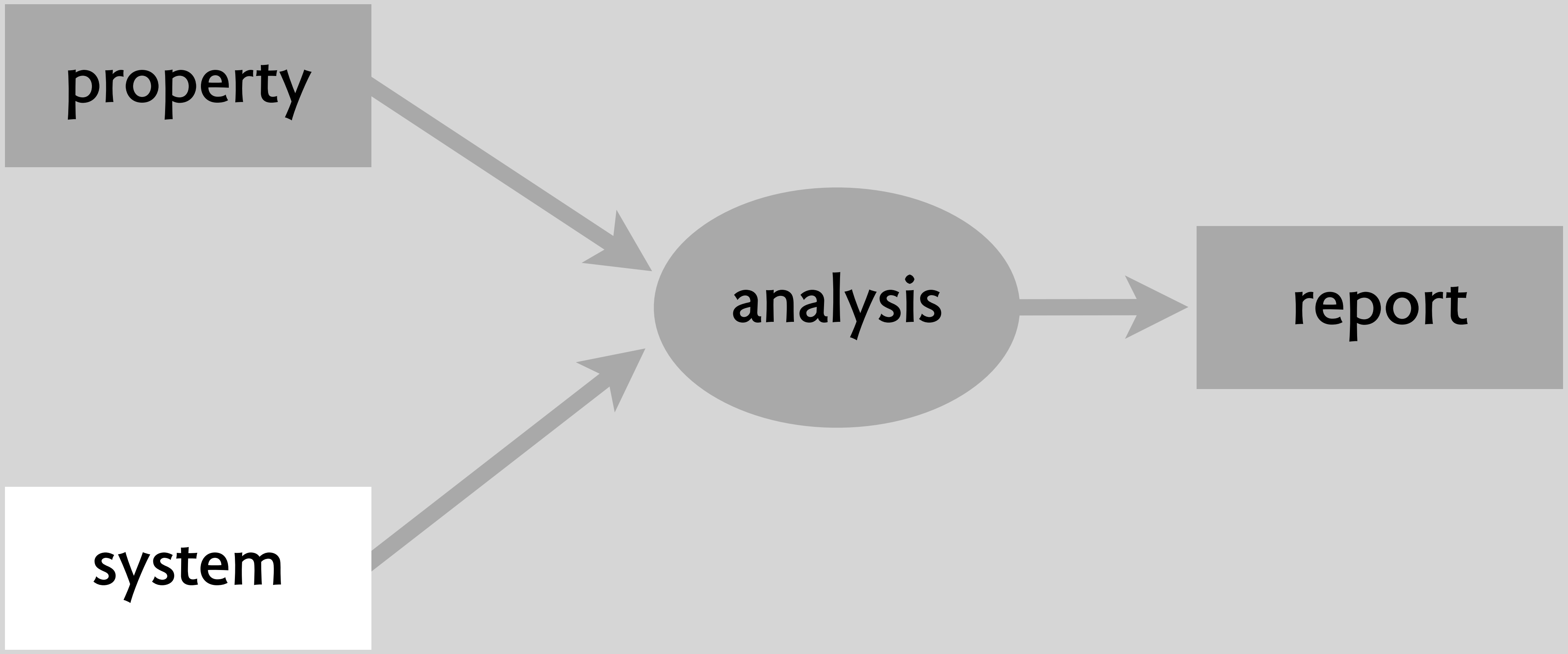
what is verification?

does this work in practice?

is the very idea flawed?



1: getting the system wrong



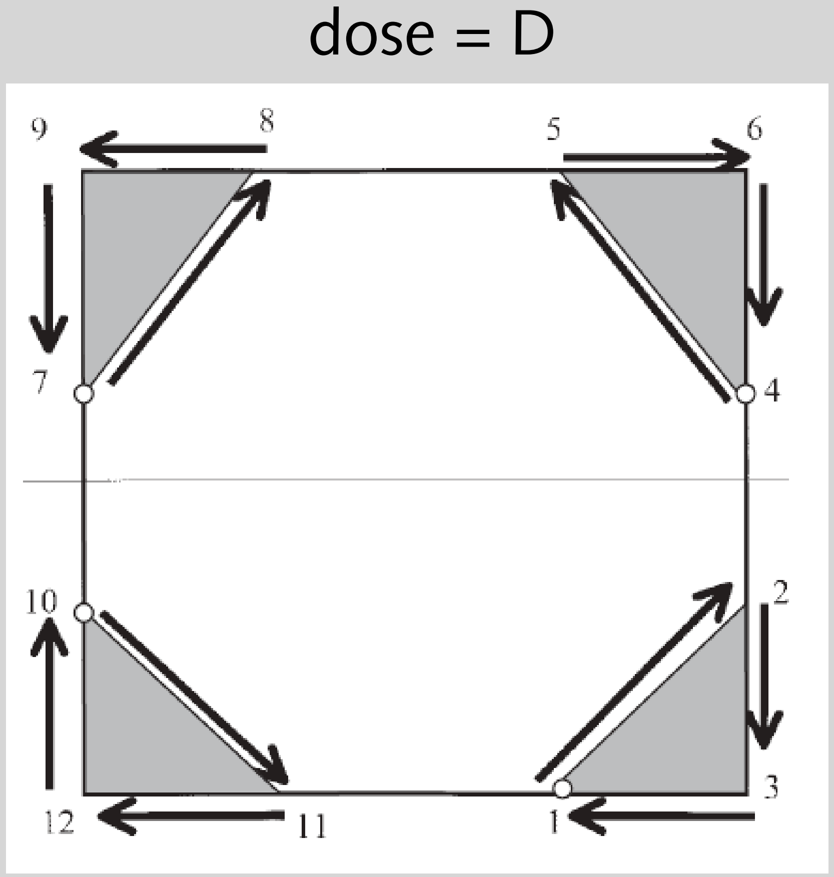
the system must include the user

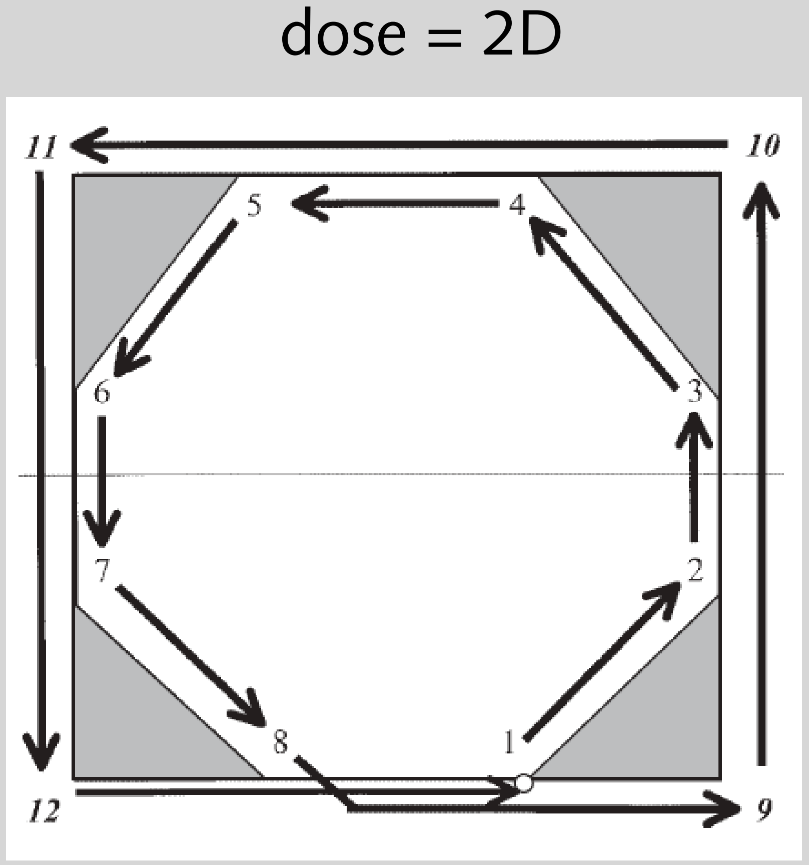
infusion pump ignores decimal point if number entered > 99
from study by Thimbleby et al: http://cs.swan.ac.uk/gcsharold/health/

“Infusion pumps, including the Baxter Colleague models, have been the source of persistent safety problems. In the past five years, the FDA has received more than 56,000 reports of adverse events associated with the use of infusion pumps. Those events have included serious injuries and more than 500 deaths.”

FDA Recall notice (2010)
http://www.fda.gov/NewsEvents/Newsroom/PressAnnouncements/ucm210664.htmd

more UIs that killed people





Panama City Hospital, 2001
Multidata therapy planning system
kills 18 patients

PLUGR, Afghanistan 2001

the system must include the plant

Airbus A320
reverse thrust protection
disable when aircraft is airborne

Warsaw 1993
strong cross winds, water on runway
aircraft aquaplaned & brakes failed
reverse thrust disabled

more disasters from ignoring plant

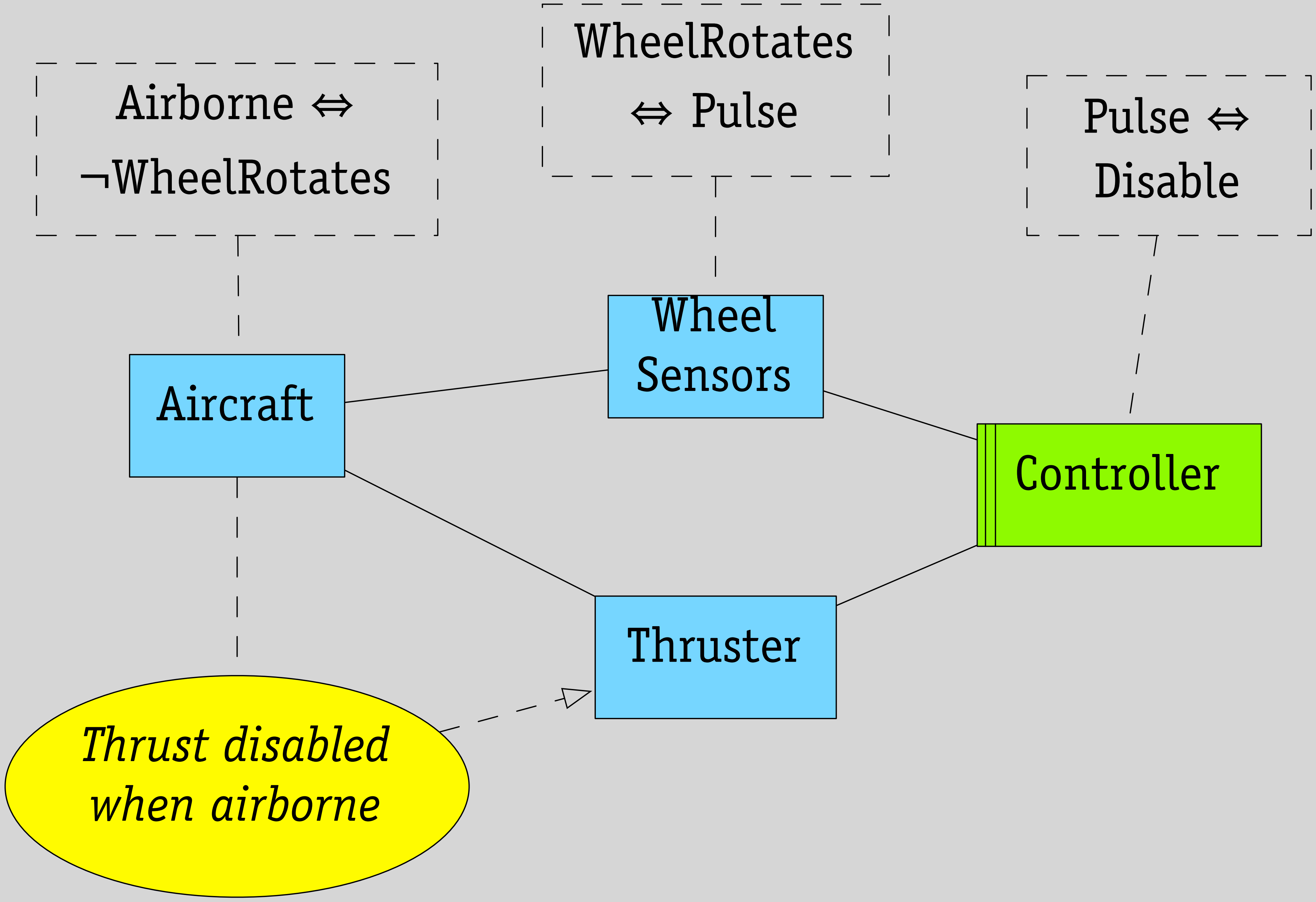
Ariane 5 (1996)

didn't account for change in lateral acceleration

Mars Polar Lander (1999)

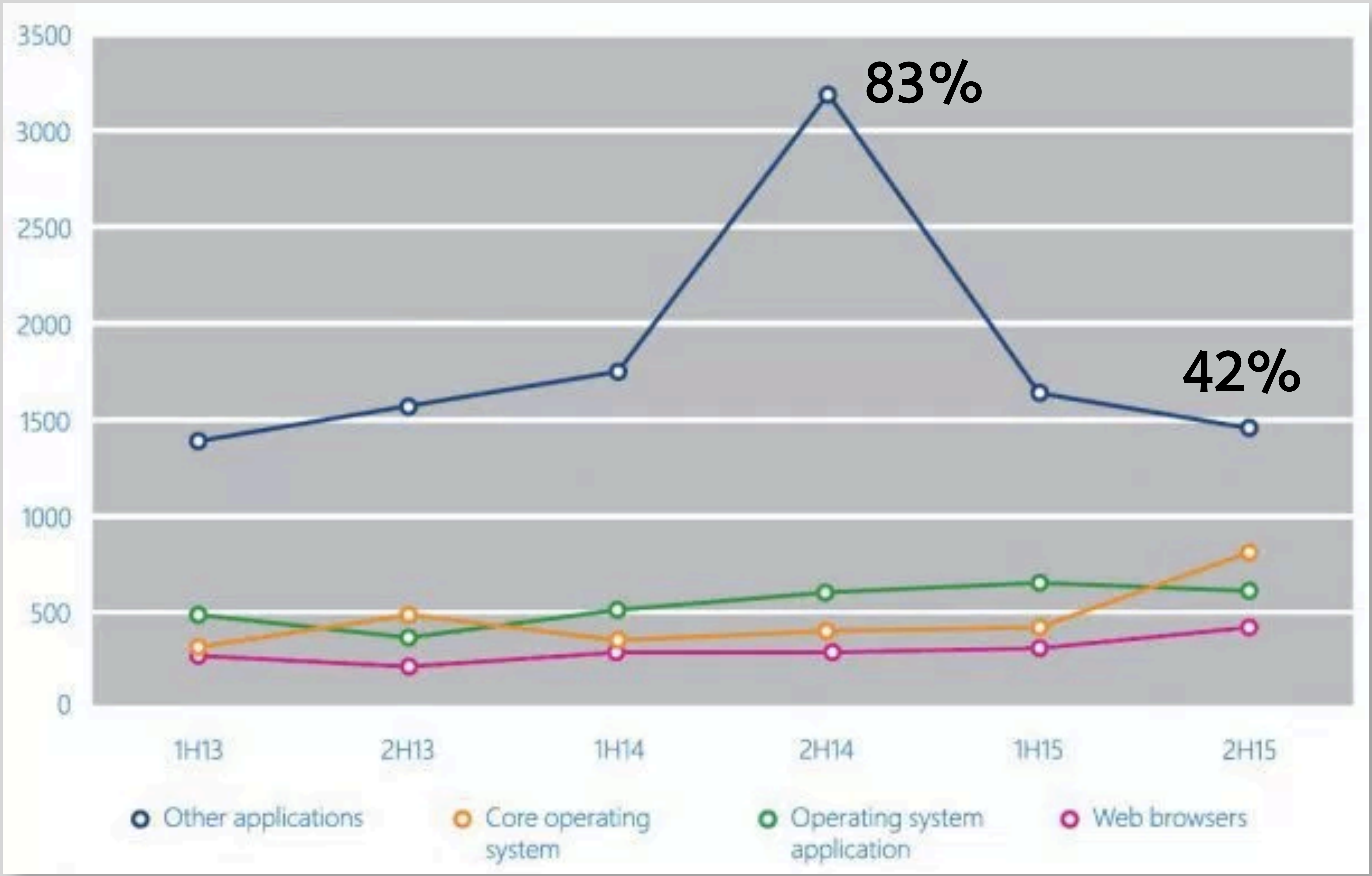
didn't account for leg compressions prior to landing

lesson: the software is not the system



see:
Gunter et al, A Reference Model for Requirements and Specifications
Michael Jackson, *Problem Frames*, Addison Wesley, 2001

infrastructure or application?



not just infrastructure: more warnings

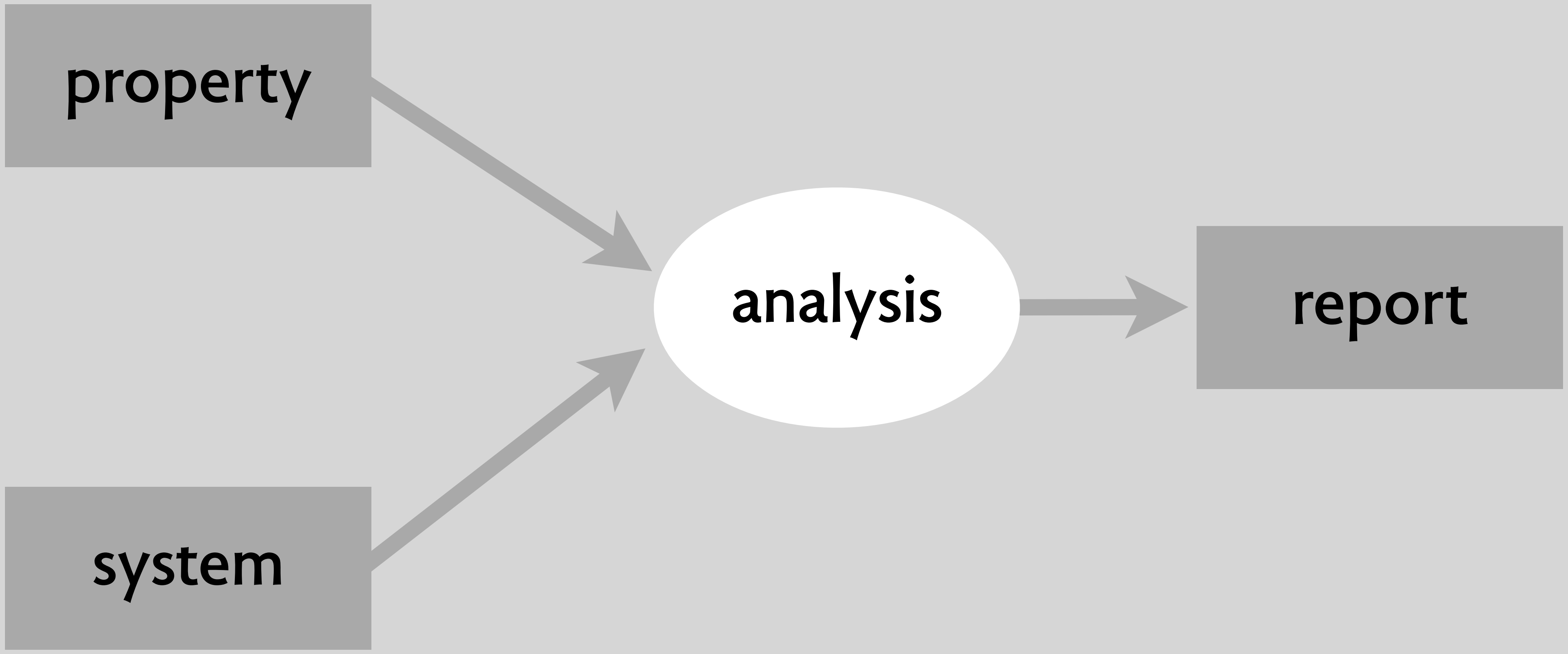
cryptographic software failures
83% of crypto vulnerabilities from how primitives used
only 17% from the crypto libraries themselves

Why does cryptographic software fail?
(Lazar, Chen Wang & Zeldovich, 2014)

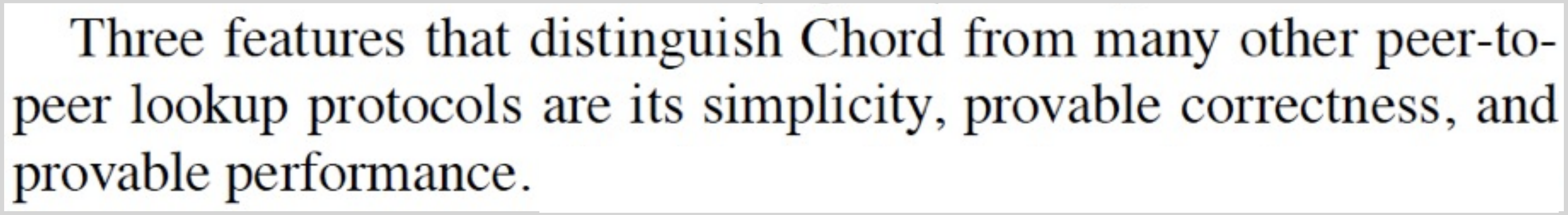
web application vulnerabilities
96% of apps contain security bugs
nearly half are application-specific

Cenzic Vulnerability Trends Report (2013)

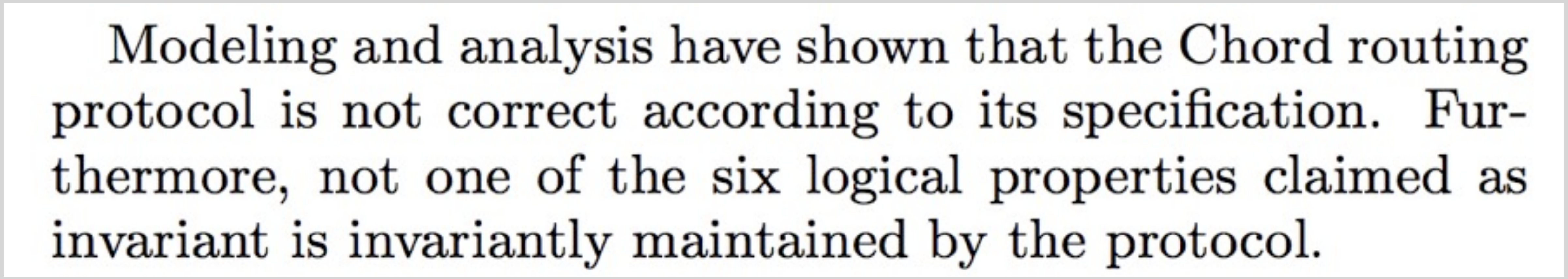
2: getting the analysis wrong



risks of informal reasoning

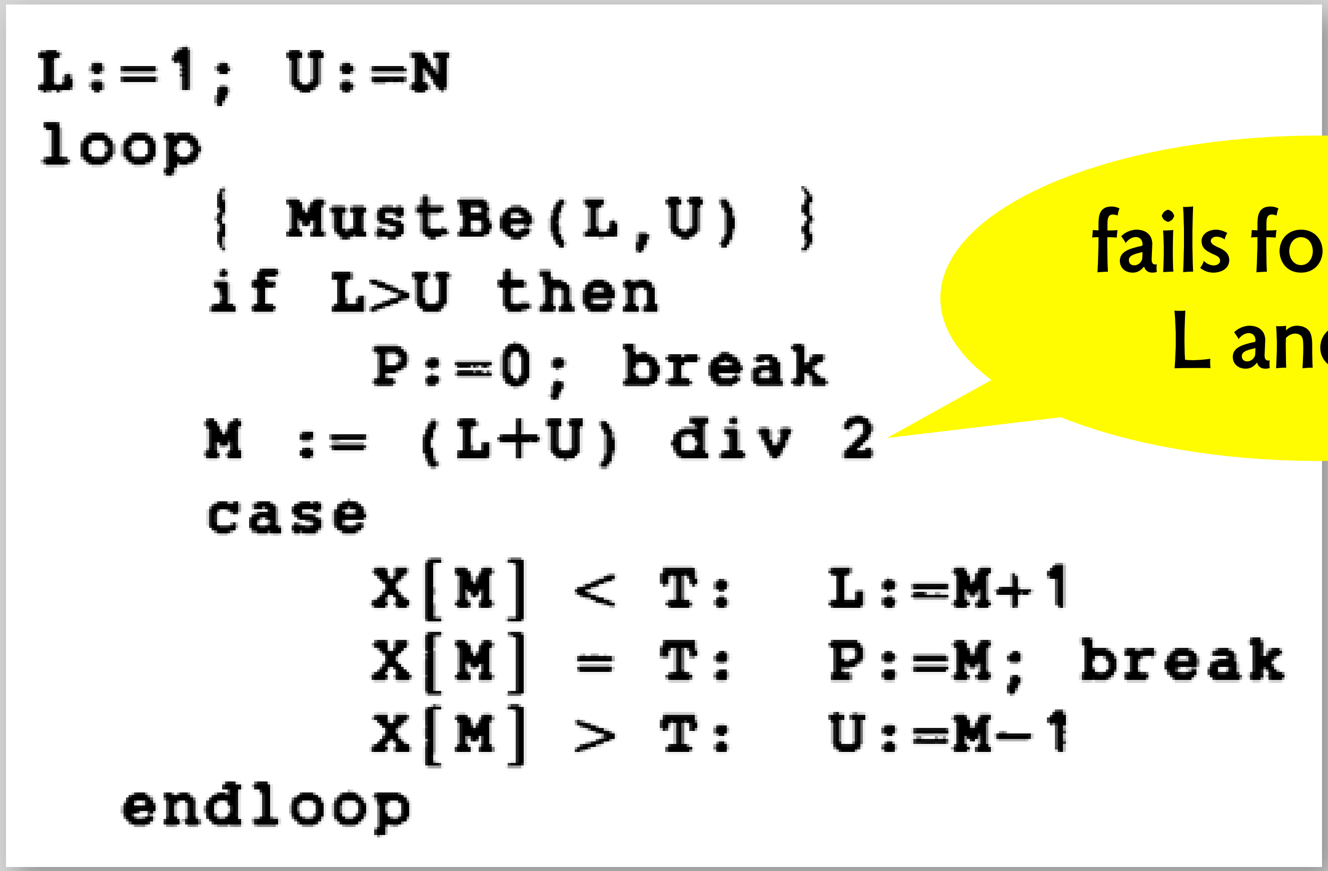

Three features that distinguish Chord from many other peer-to-peer lookup protocols are its simplicity, provable correctness, and provable performance.

Ion Stoica et al. Chord: A Scalable Peer to Peer Lookup Service for Internet Applications, SIGCOMM 2001 (also TON, 2003)


Modeling and analysis have shown that the Chord routing protocol is not correct according to its specification. Furthermore, not one of the six logical properties claimed as invariant is invariantly maintained by the protocol.

Pamela Zave. Invariant-Based Verification of Routing Protocols: The Case of Chord, 2009

risks of axiomatization



```
L:=1; U:=N
loop
    { MustBe(L,U) }
    if L>U then
        P:=0; break
    M := (L+U) div 2
    case
        X[M] < T:  L:=M+1
        X[M] = T:  P:=M; break
        X[M] > T:  U:=M-1
  endloop
```

fails for large L and U

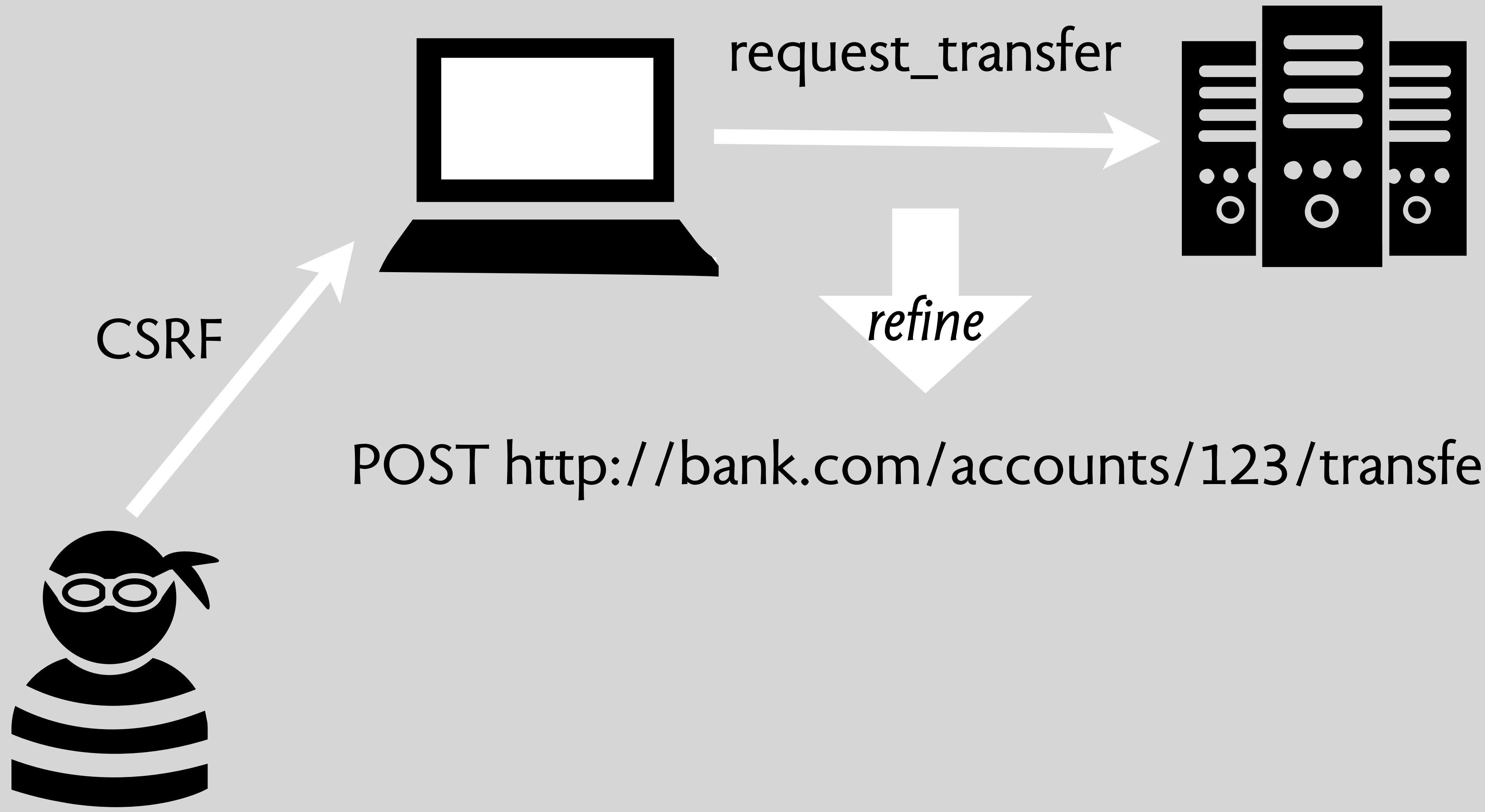
from Jon Bentley, *Programming Pearls* (1983)

"Nearly all Binary Searches and Mergesorts are Broken"
Josh Bloch (2006)

https://research.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html

risks of abstraction

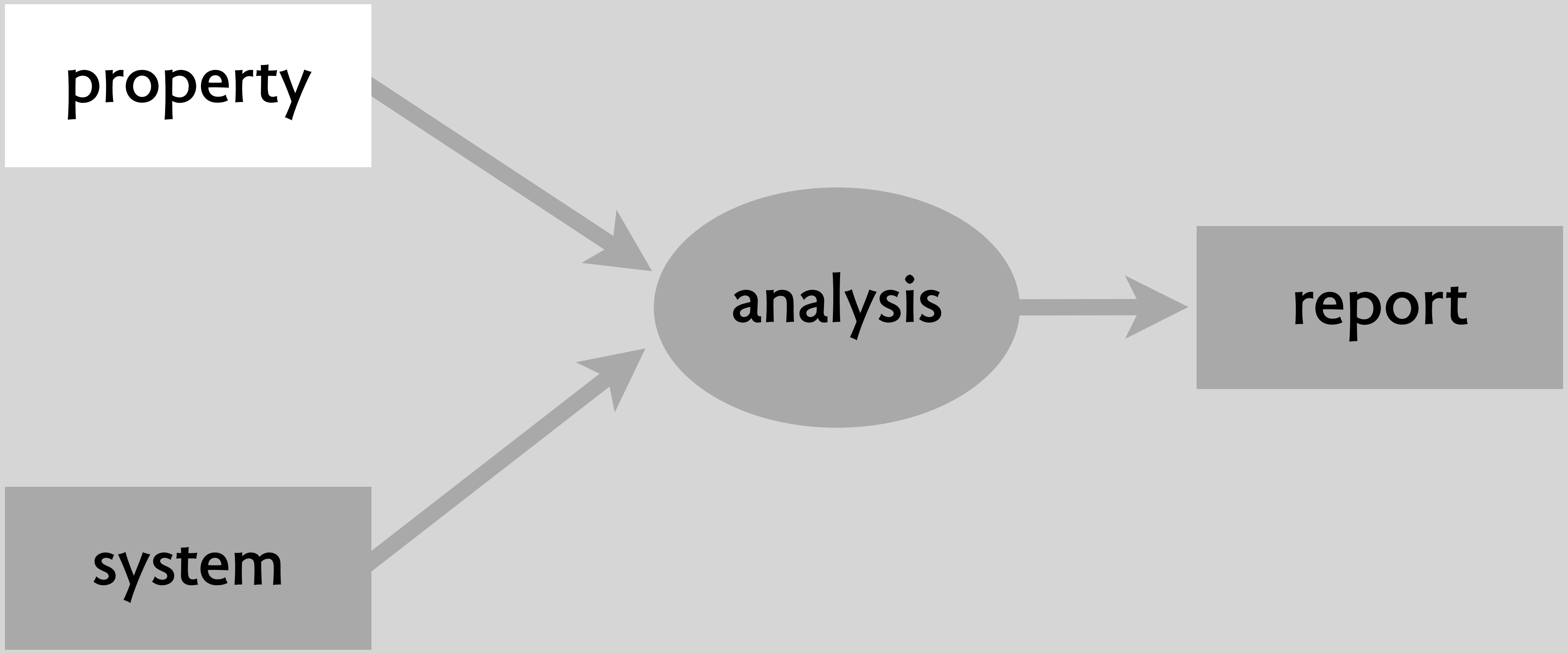
refinement isn't sound if interference is possible



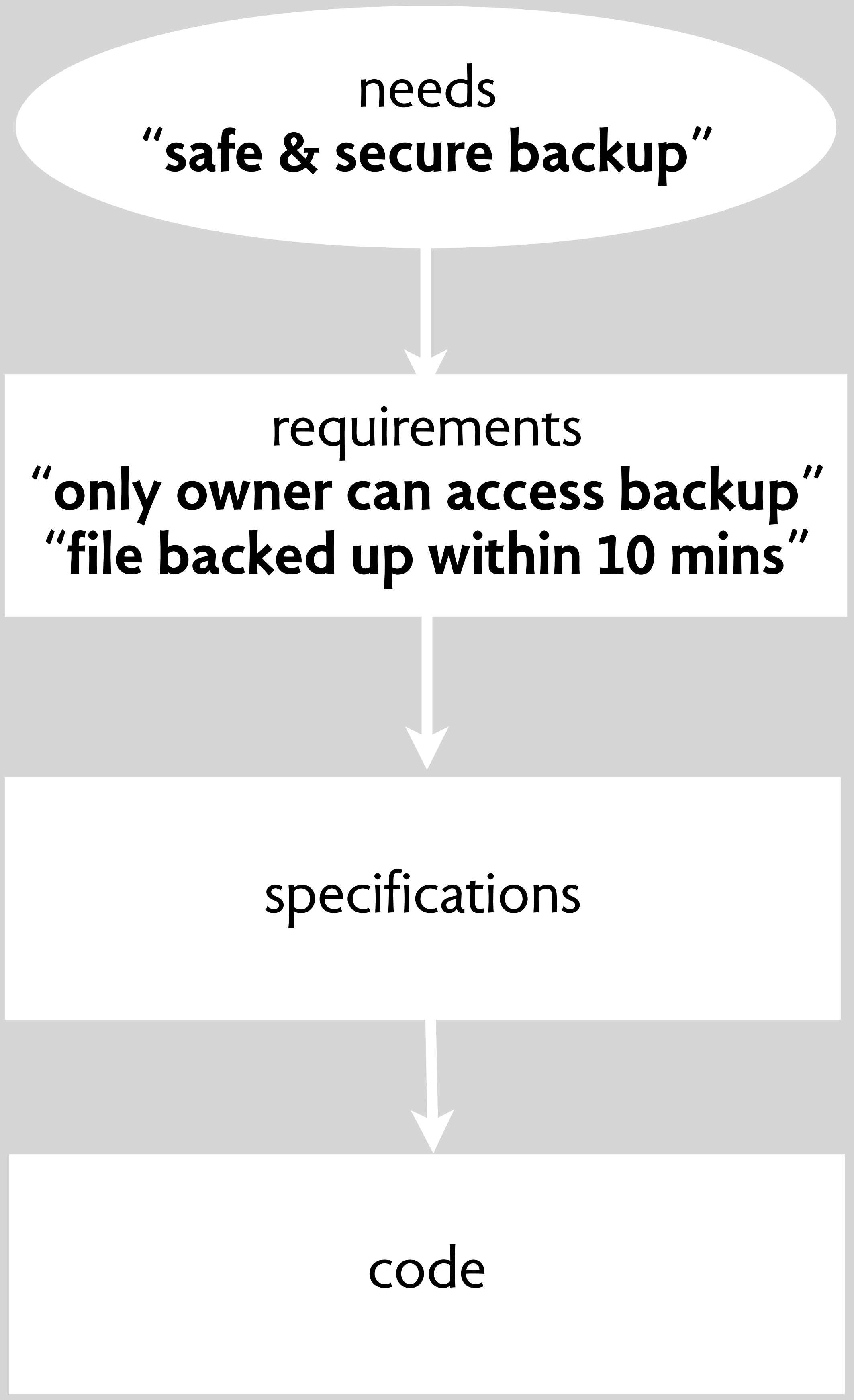
POST http://bank.com/accounts/123/transfers

Eunsuk Kang, Aleksandar Milicevic, Daniel Jackson
Multi-Representational Security Analysis, FSE 2016

3: getting the property wrong



when requirements are designs


needs
“**safe & secure backup**”
requirements
“**only owner can access backup**”
“**file backed up within 10 mins**”
specifications
code

root of the problem: requirements are **design** properties

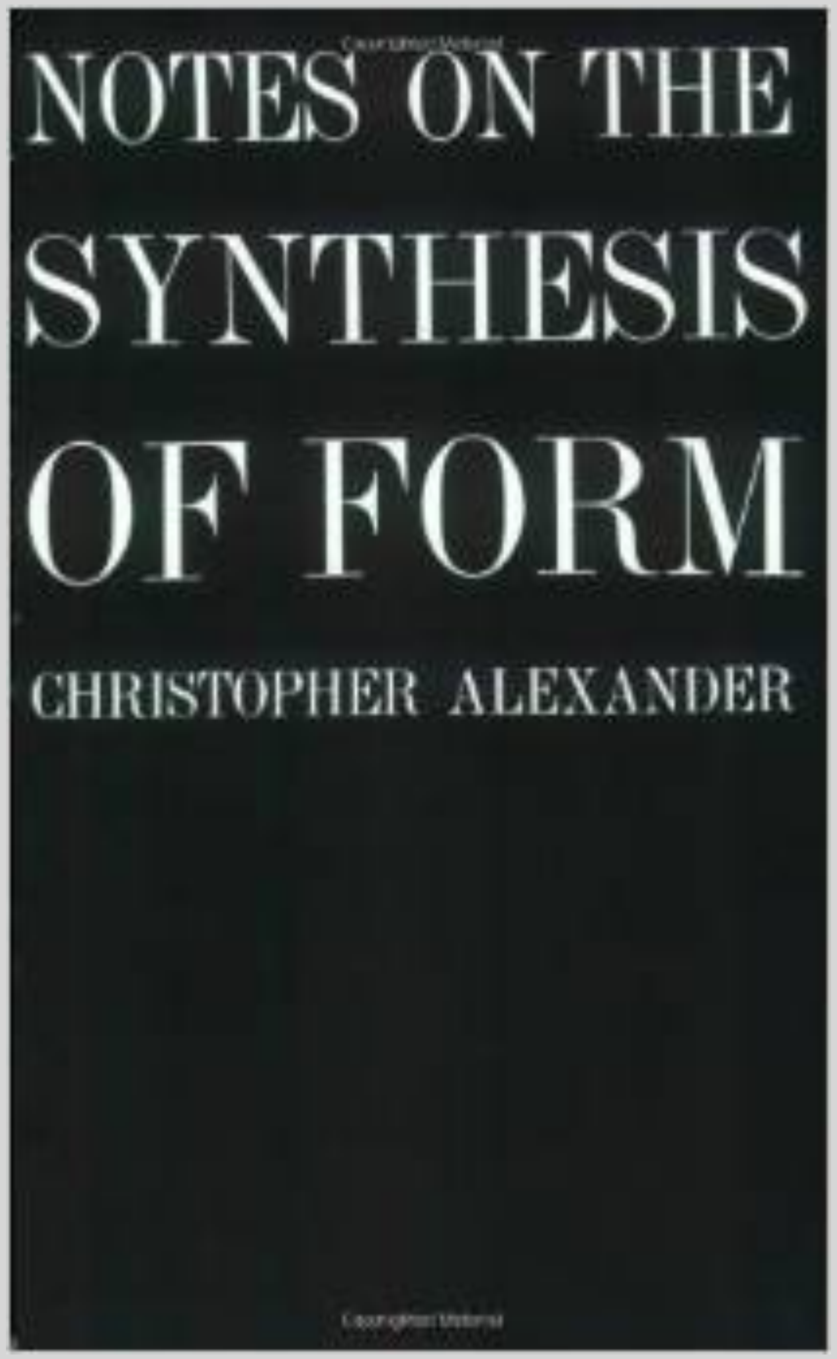
not the wrong property: the wrong **kind** of property

temporary file with secret data gets exposed

ransomware encrypts entire backup

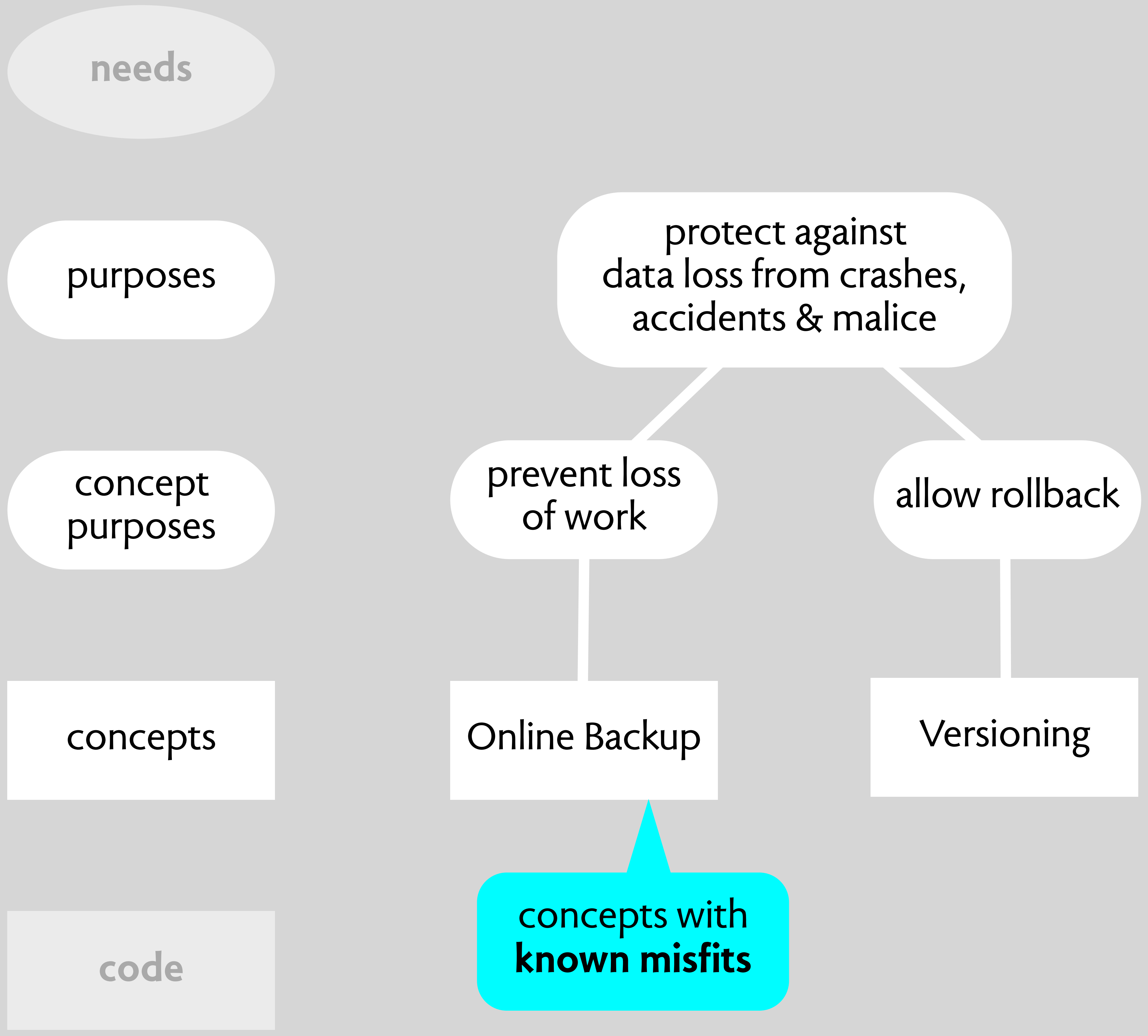
churn on large video causes loss of old versions

christopher alexander knew this

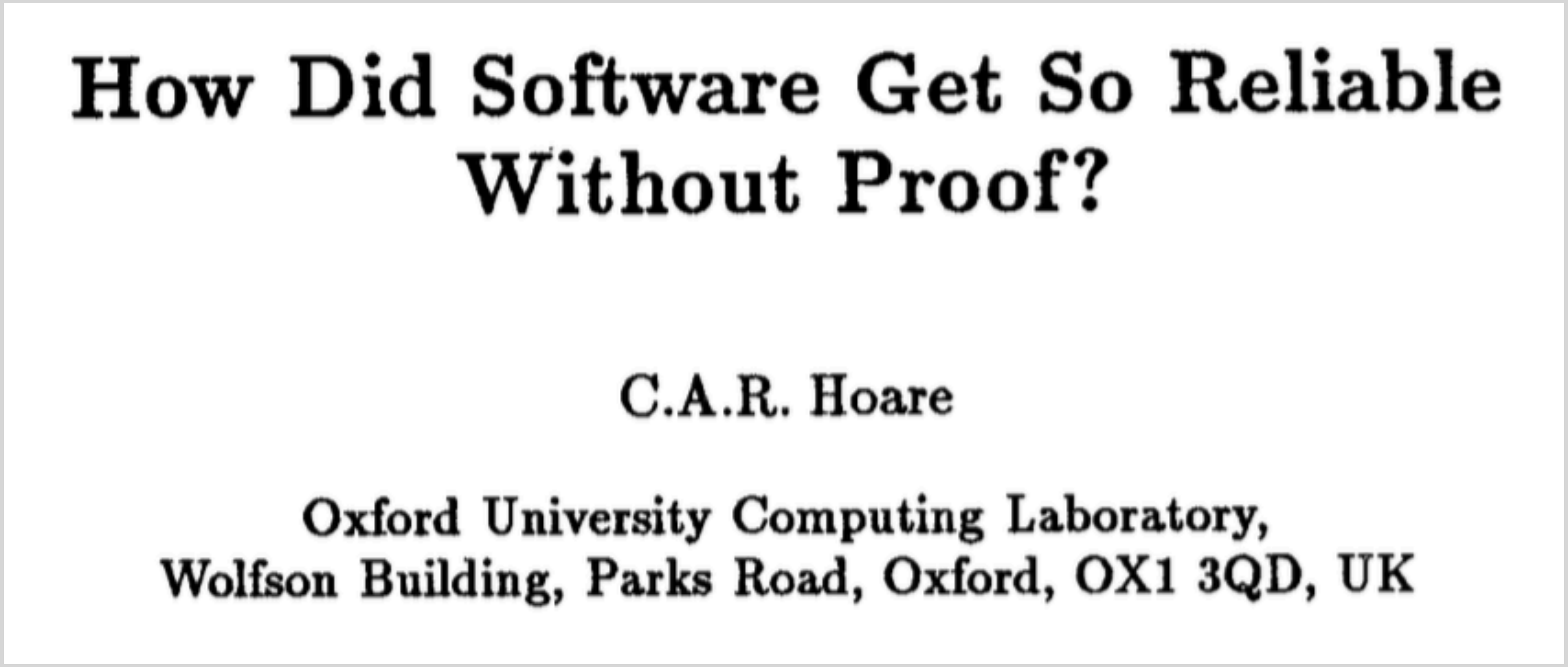

NOTES ON THE SYNTHESIS OF FORM

CHRISTOPHER ALEXANDER

Such a list of requirements is potentially endless... But if we think of the requirements from a negative point of view, as potential misfits, there is a simple way of picking a finite set. This is because it is through misfit that the problem originally brings itself to our attention. We take just those relations between form and context which obtrude most strongly, which demand attention most clearly, which seem most likely to go wrong. We cannot do better than this.


needs
purposes
concept purposes
concepts
code
protect against data loss from crashes, accidents & malice
prevent loss of work
allow rollback
Online Backup
Versioning
concepts with **known misfits**

is verification even necessary?


How Did Software Get So Reliable Without Proof?

C.A.R. Hoare

Oxford University Computing Laboratory,
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

my hypothesis: clean **concepts** + unit testing + natural selection

conclusion #1

stop looking under the lamppost!

comfortable research
formal & empirical
produces algorithms & tools
focused on programmers
and the code they write

industry prefers this too

uncomfortable research
informal & philosophical
produces design theory & method
focused on stakeholders
and the whole system

Who could fault an approach that offers greater credibility at reduced cost?

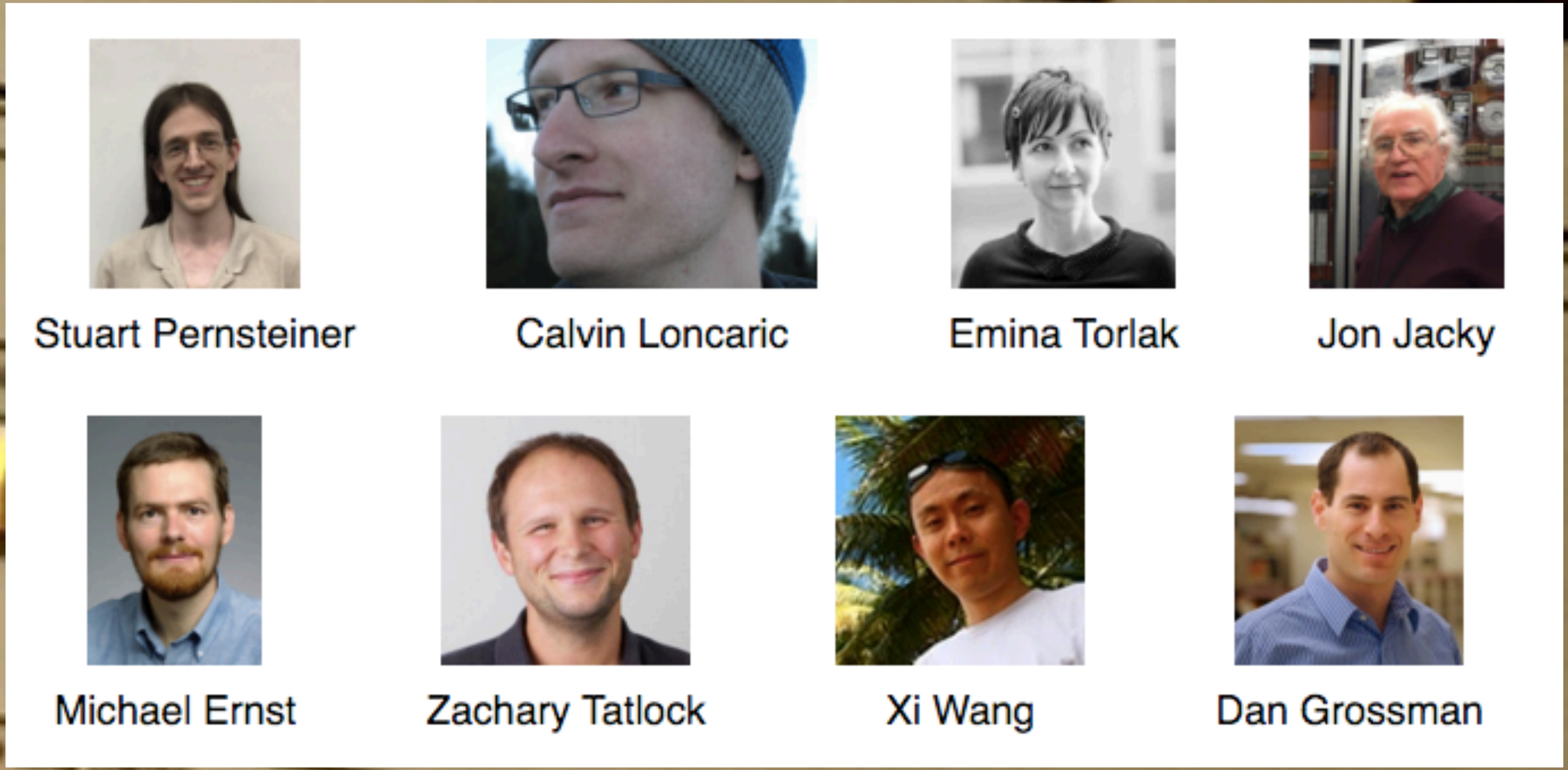
CACM
April 2009

BY DANIEL JACKSON

A Direct Path to Dependable Software

SOFTWARE PLAYS A fundamental role in our society, bringing enormous benefits to all fields. But because many of our current systems are highly centralized and tightly coupled,[33] we are also susceptible to massive and coordinated failure.

UW radiotherapy project



Stuart Pernsteiner | Calvin Loncaric | Emina Torlak | Jon Jacky

Michael Ernst | Zachary Tatlock | Xi Wang | Dan Grossman

conclusion #2

loosen up, don't be dogmatic

a (resurgent?) narrow view
soundness > completeness
false positives don't matter
proof: you have no bugs!

a more open view
soundness of counterexamples too
confidence is not binary
proof: sorry, I can't find more bugs!

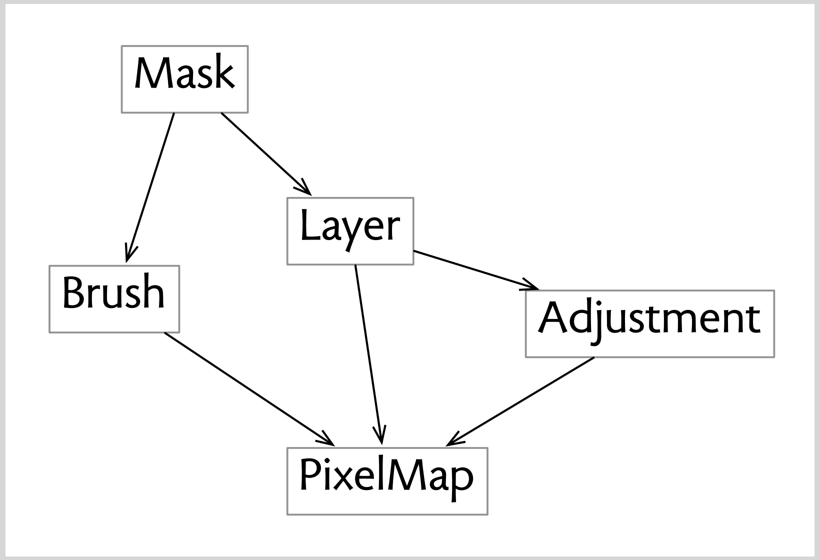
conclusion #3

rethink software design

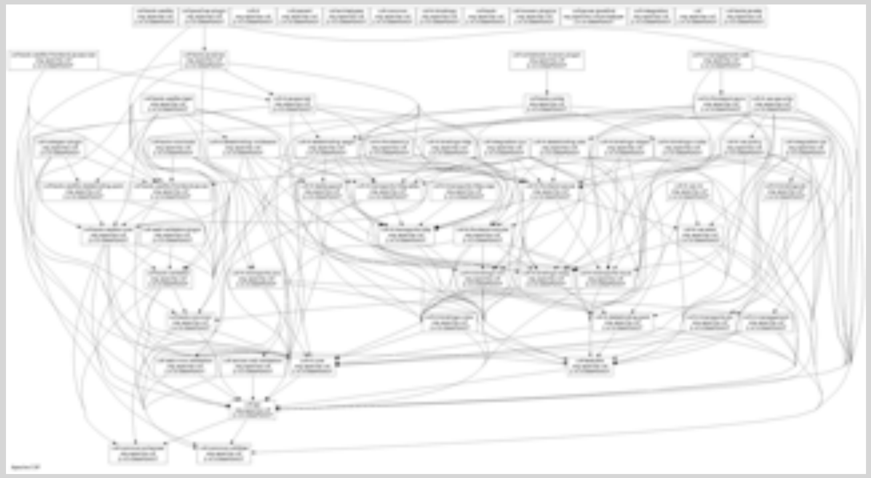
UI design
soft & human
about presentation

programming
hard & technical
about content

a better view of software design



conceptual design:
essential concepts
& behavior



representation design:
organization & performance

some research avenues

lightweight verification of code
trading confidence for automation

new programming paradigms
correctness by construction

robust system-level analysis
beyond hazard analysis, FMEA, etc

design thinking for software
going beyond process & sensibility

architecture for dependability
shrinking the trusted base

inferring confidence from tests
based on the software alone