# When testing just doesn't cut it

Lars Hupel
BOB Konferenz
2023-03-17

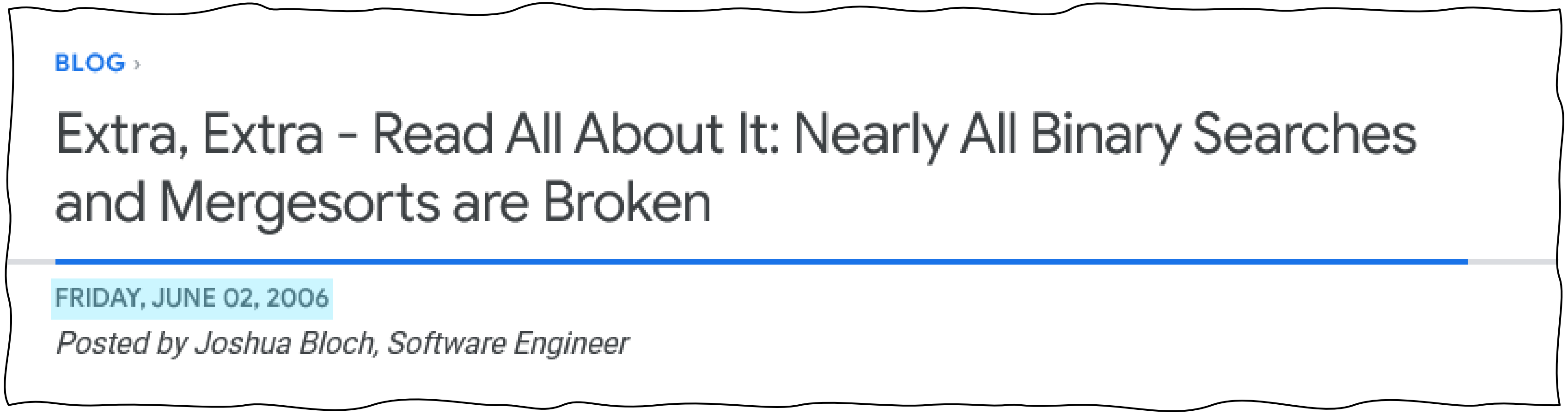
Giesecke+Devrient
Creating Confidence

# Where would this line be used?

```
int mid = (low + high) / 2
```

# … and what’s wrong with it?

```
int mid = (low + high) / 2
```



BLOG ›

# Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken

FRIDAY, JUNE 02, 2006

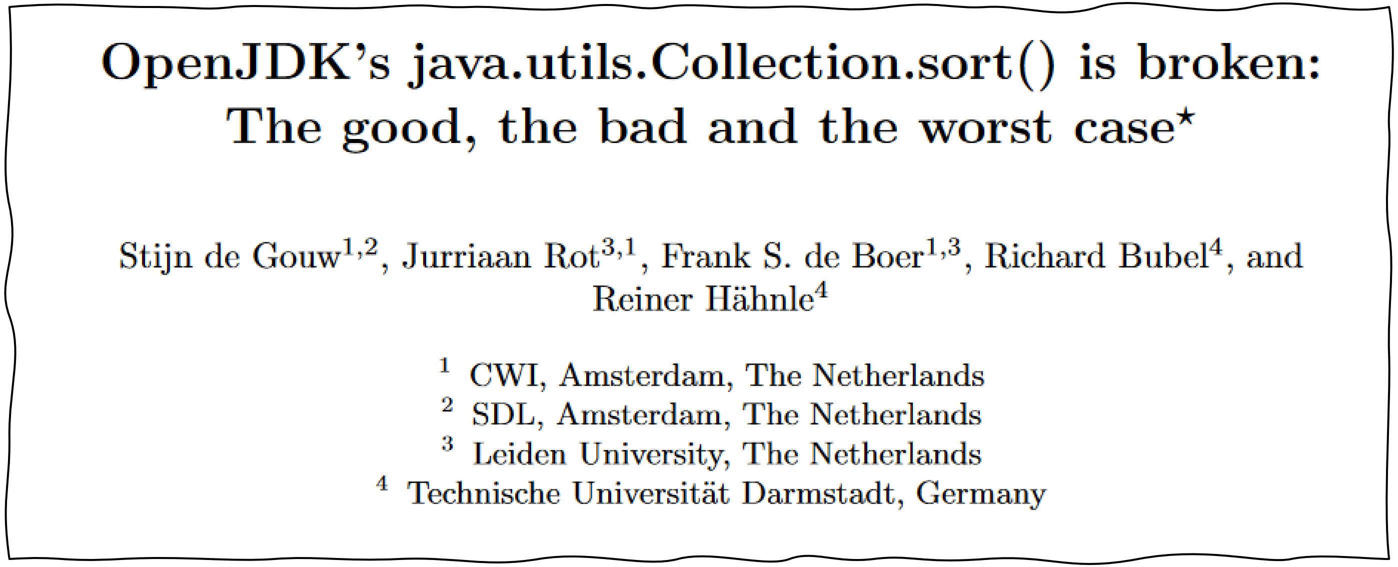
*Posted by Joshua Bloch, Software Engineer*

HULK SMASH THE STATE
HULK SMASH THE STATE
HULK SMASH

# Sorting in Java

```
list so
```

```
java.lang.ArrayIndexOutOfBoundsException: 19
        at java.util.ComparableTimSort.pushRun(ComparableTimSort.java:352)
        at java.util.ComparableTimSort.sort(ComparableTimSort.java:181)
        at java.util.ComparableTimSort.sort(ComparableTimSort.java:146)
        at java.util.Arrays.sort(Arrays.java:472)
        at BreakTimSort.run(BreakTimSort.java:68)
        at BreakTimSort.main(BreakTimSort.java:72)
```

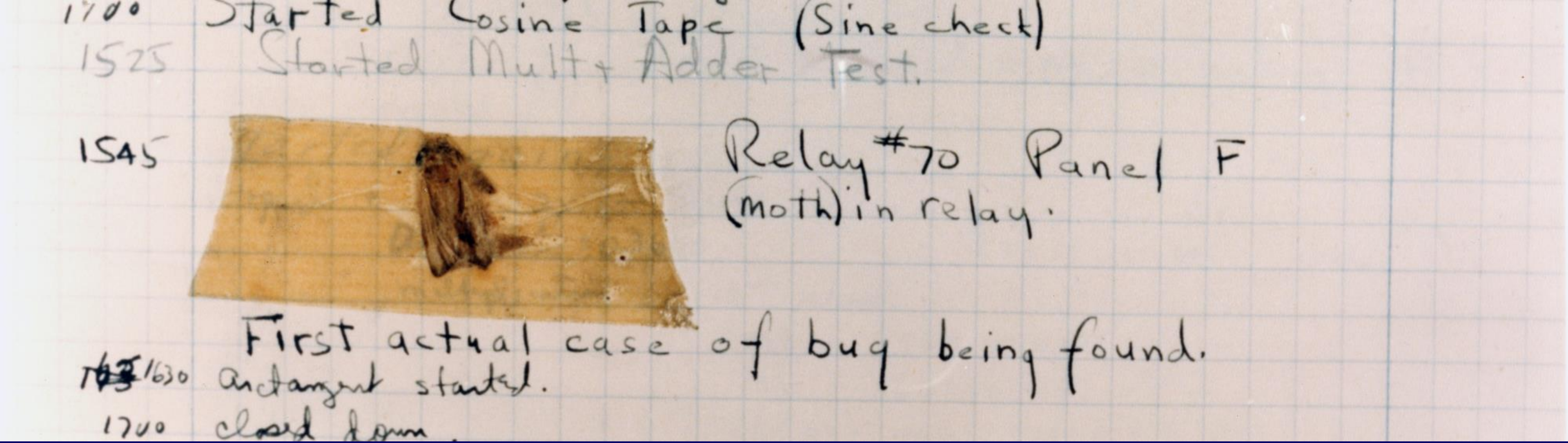


# OpenJDK's java.utils.Collection.sort() is broken: The good, the bad and the worst case*

Stijn de Gouw[1,2], Jurriaan Rot[3,1], Frank S. de Boer[1,3], Richard Bubel[4], and Reiner Hähnle[4]

[1] CWI, Amsterdam, The Netherlands
[2] SDL, Amsterdam, The Netherlands
[3] Leiden University, The Netherlands
[4] Technische Universität Darmstadt, Germany

CAV 2015



1700 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545 Relay #70 Panel F
(moth) in relay.

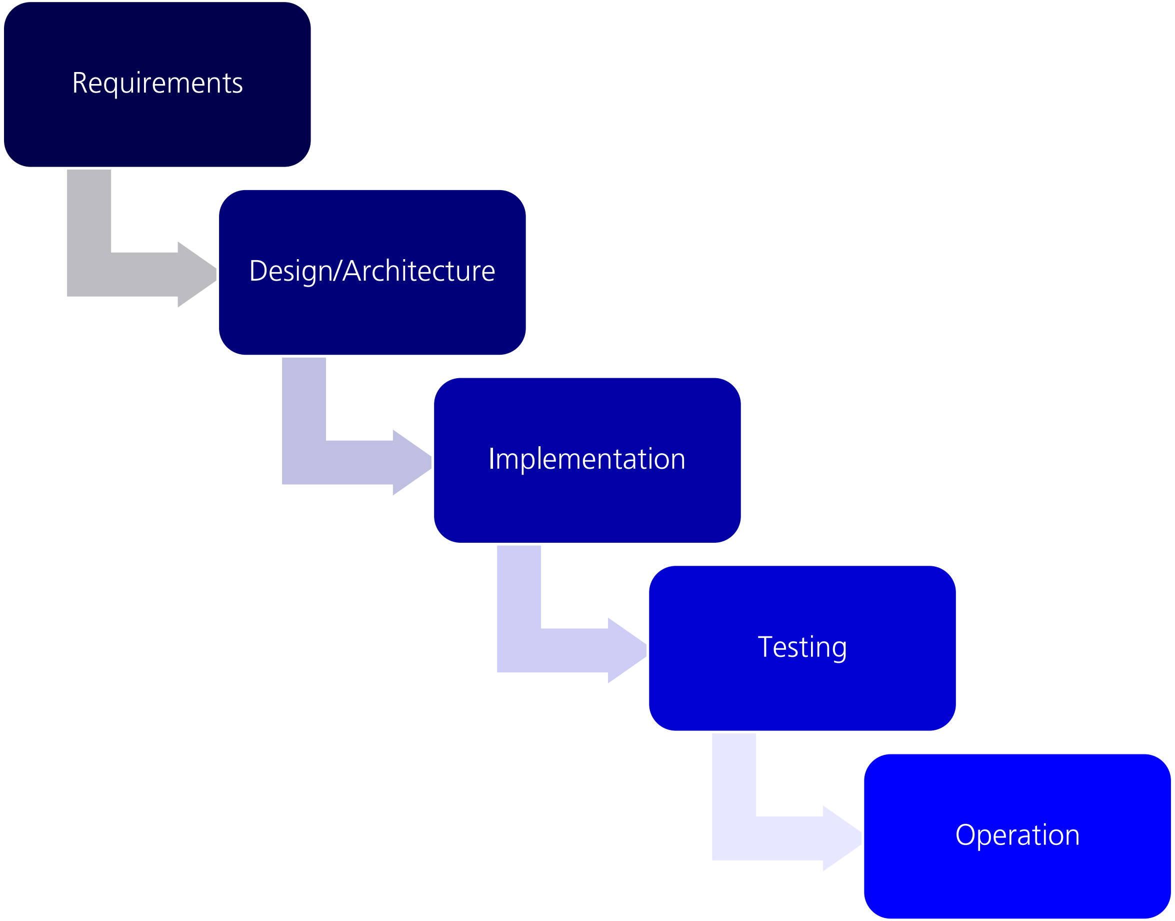
First actual case of bug being found.

1630 Antangent started.

1700 closed down.

# Programming & Bugs

Giesecke+Devrient
Creating Confidence



Requirements

Design/Architecture

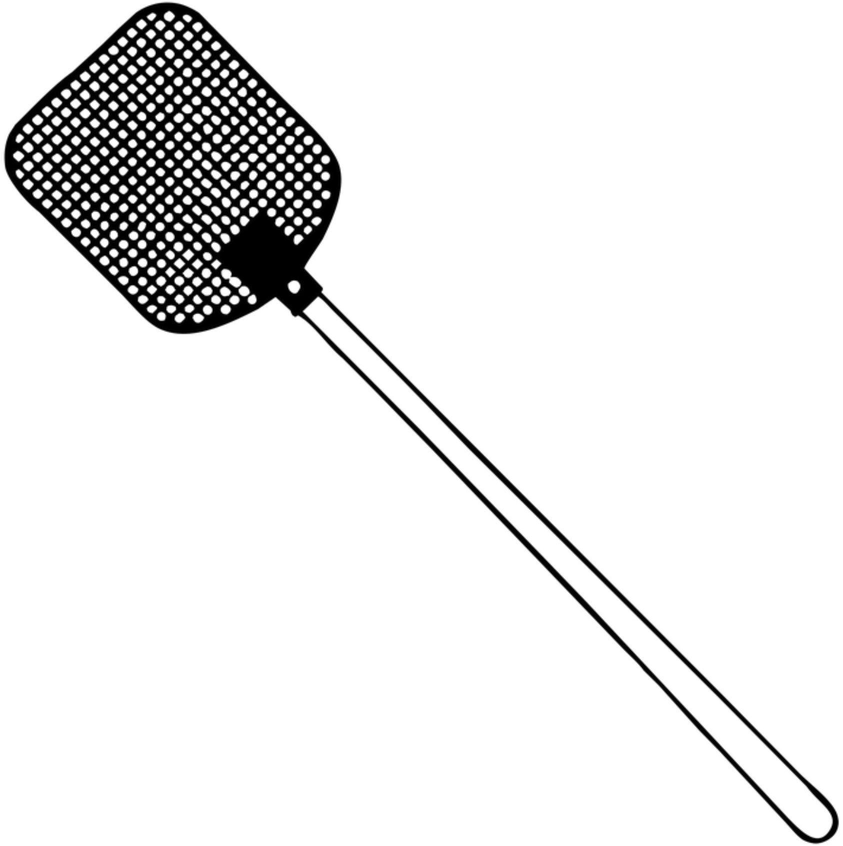
Implementation

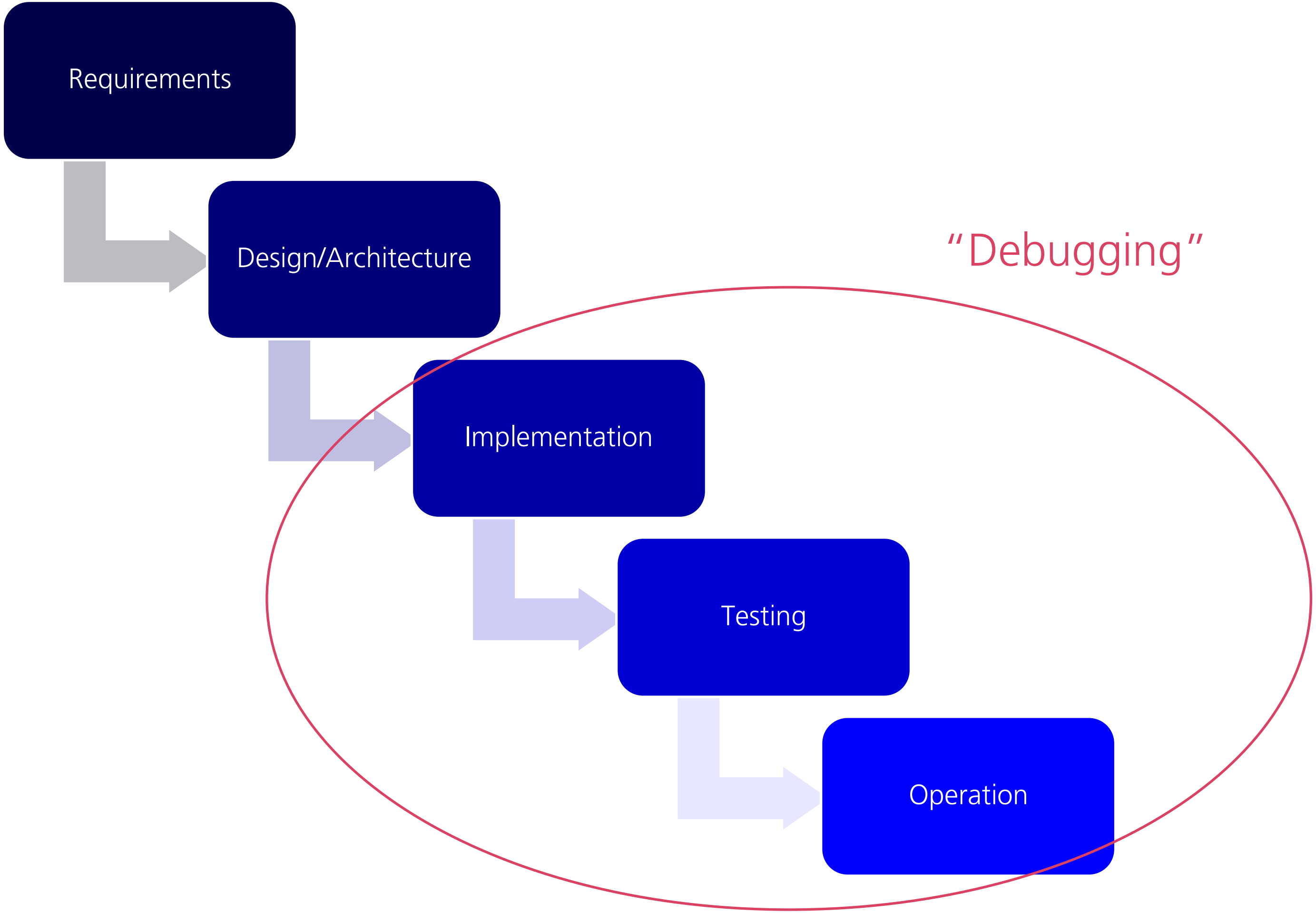
Testing

Operation

# Bugs: We don’t like them

Yet, they keep cropping up …




Requirements

Design/Architecture

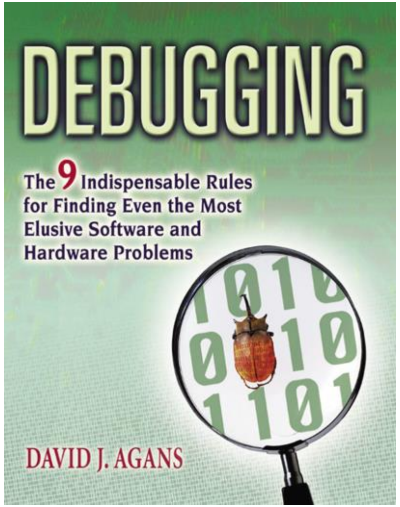
Implementation

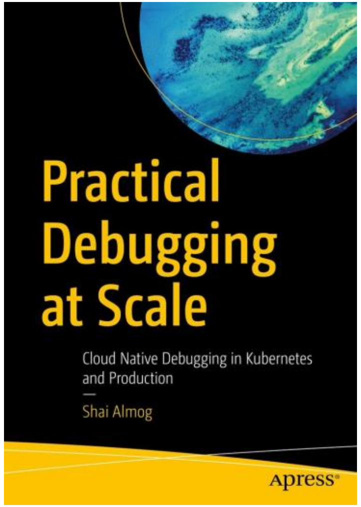
Testing

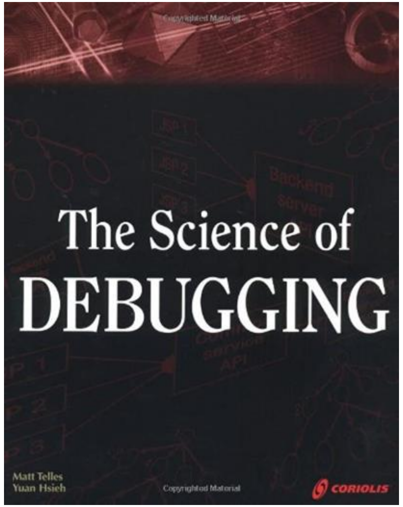
Operation

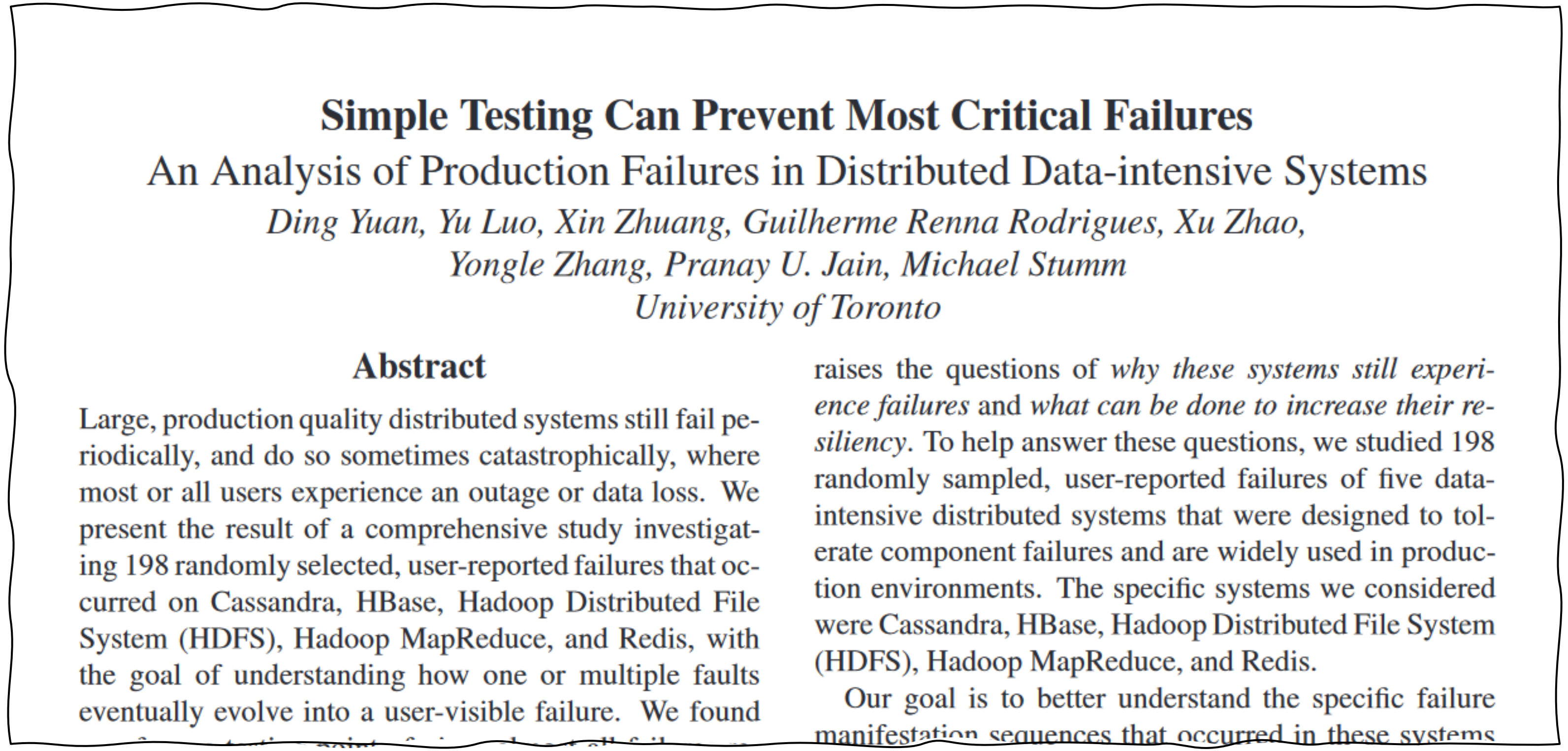
“Debugging”

# Debugging is a core skill


DEBUGGING
The 9 Indispensable Rules for Finding Even the Most Elusive Software and Hardware Problems
DAVID J. AGANS


Practical Debugging at Scale
Cloud Native Debugging in Kubernetes and Production
Shai Almog
Apress





# Simple Testing Can Prevent Most Critical Failures

## An Analysis of Production Failures in Distributed Data-intensive Systems

*Ding Yuan, Yu Luo, Xin Zhuang, Guilherme Renna Rodrigues, Xu Zhao,*
*Yongle Zhang, Pranay U. Jain, Michael Stumm*
*University of Toronto*

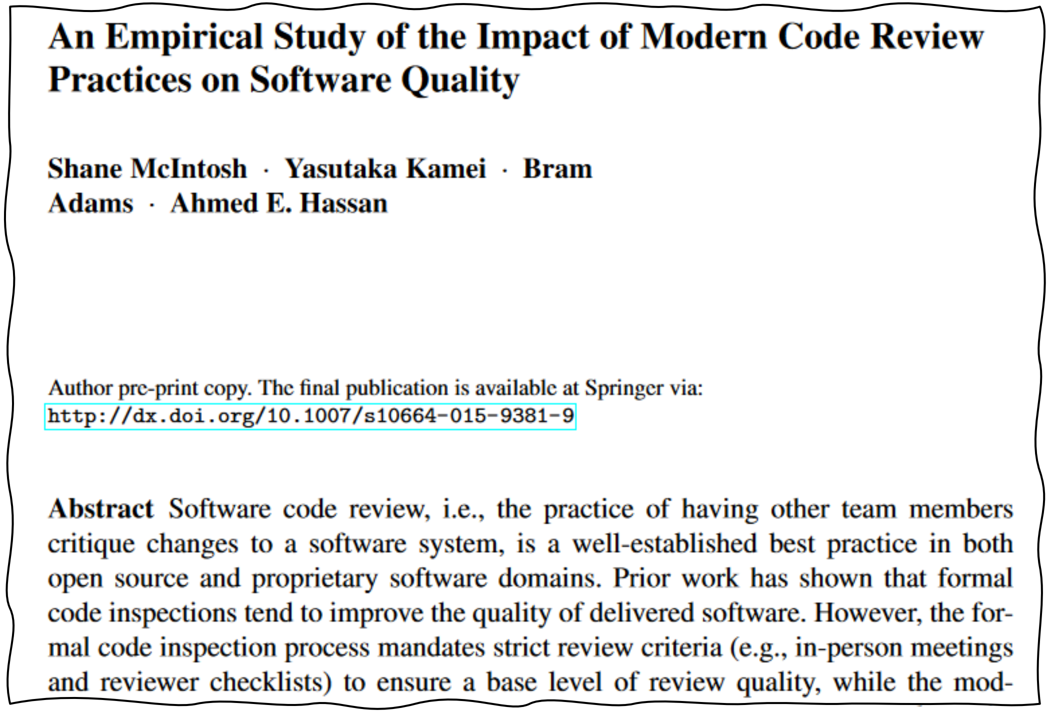
## Abstract

Large, production quality distributed systems still fail periodically, and do so sometimes catastrophically, where most or all users experience an outage or data loss. We present the result of a comprehensive study investigating 198 randomly selected, user-reported failures that occurred on Cassandra, HBase, Hadoop Distributed File System (HDFS), Hadoop MapReduce, and Redis, with the goal of understanding how one or multiple faults eventually evolve into a user-visible failure. We found

raises the questions of *why these systems still experience failures* and *what can be done to increase their resiliency*. To help answer these questions, we studied 198 randomly sampled, user-reported failures of five data-intensive distributed systems that were designed to tolerate component failures and are widely used in production environments. The specific systems we considered were Cassandra, HBase, Hadoop Distributed File System (HDFS), Hadoop MapReduce, and Redis.

Our goal is to better understand the specific failure manifestation sequences that occurred in these systems

OSDI 2014



# An Empirical Study of the Impact of Modern Code Review Practices on Software Quality

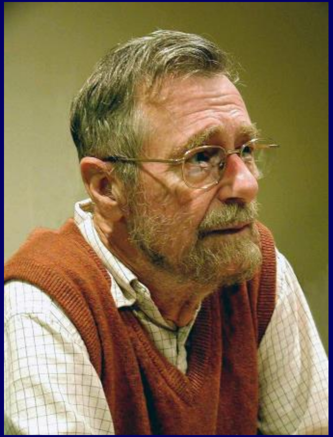
Shane McIntosh · Yasutaka Kamei · Bram Adams · Ahmed E. Hassan

Author prc-print copy. Thc final publication is availablc at Springer via:
http://dx.doi.org/10.1007/s10664-015-9381-9

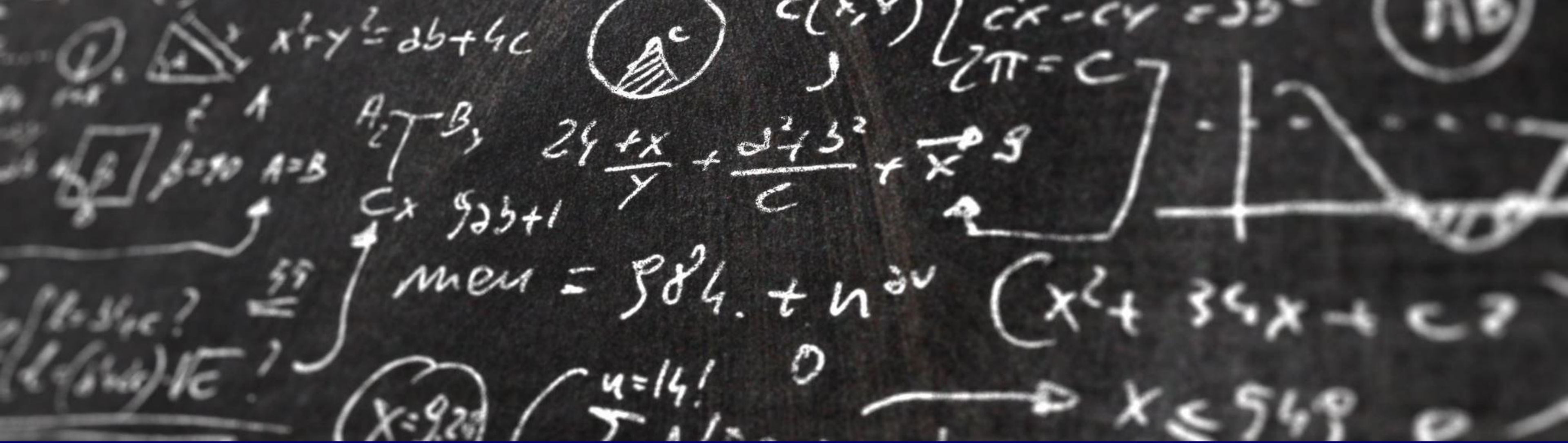
**Abstract** Software code review, i.e., the practice of having other team members critique changes to a software system, is a well-established best practice in both open source and proprietary software domains. Prior work has shown that formal code inspections tend to improve the quality of delivered software. However, the formal code inspection process mandates strict review criteria (e.g., in-person meetings and reviewer checklists) to ensure a base level of review quality, while the mod-

Empirical Software Engineering 2015

Welcome!
The conference will start at
1:00 pm CET
Welcome!
Welcome!
Not streaming



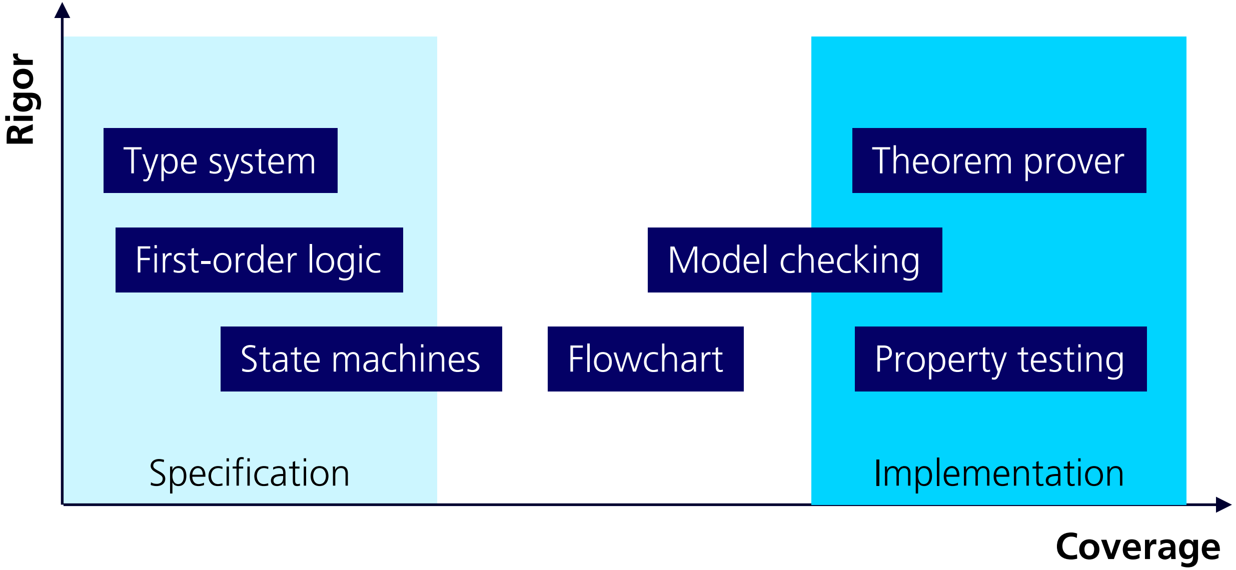
“Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence”



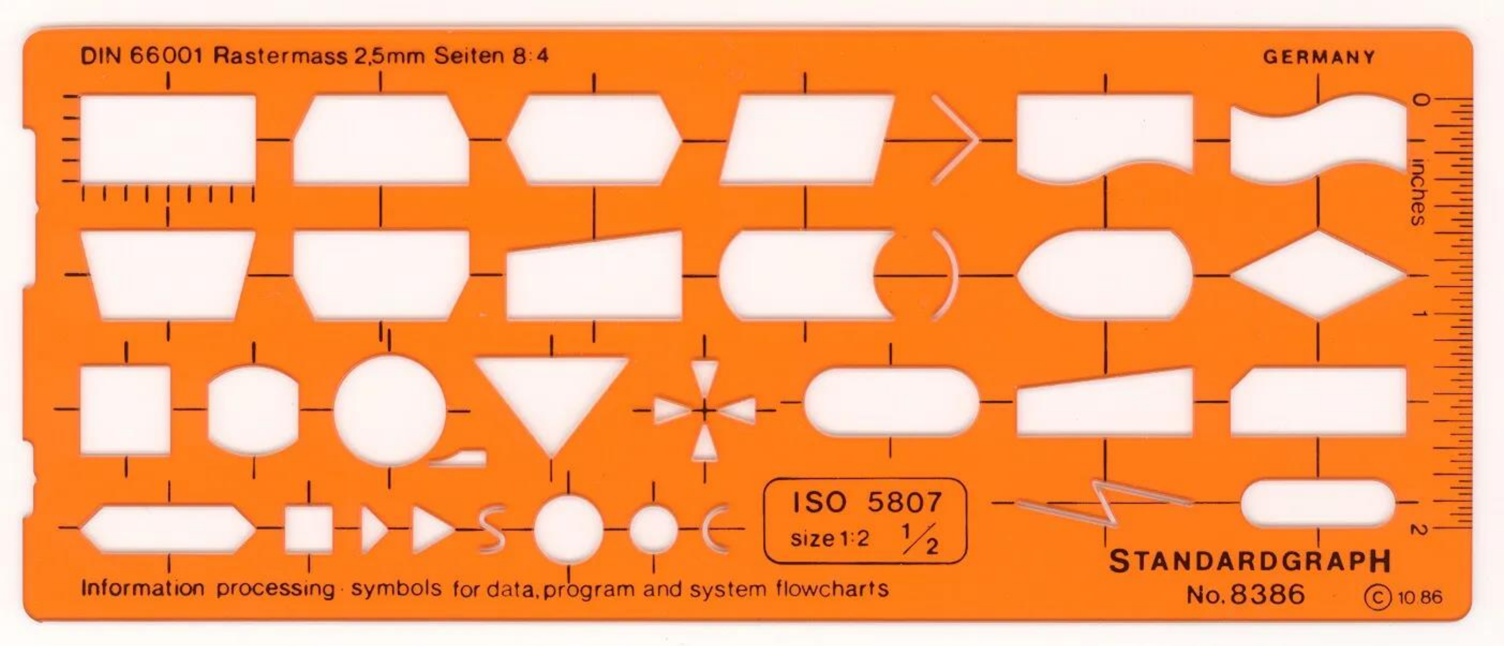
# Formal Methods

**“*Formal Methods* refers to mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems”**

# What are *Formal Methods?*

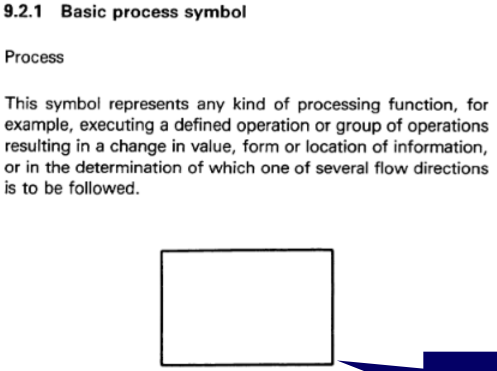


# ISO 5807 Flowchart


DIN 66001 Rastermass 2,5mm Seiten 8:4
GERMANY
0
inches
1
2
ISO 5807
size 1:2 1/2
STANDARDGRAPH
No. 8386
© 10.86
Information processing · symbols for data, program and system flowcharts

Giesecke+Devrient
Creating Confidence

# ISO 5807:1985

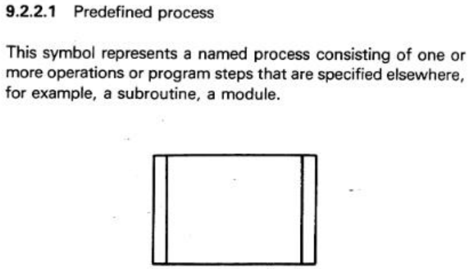


### 9.2.1 Basic process symbol

Process

This symbol represents any kind of processing function, for example, executing a defined operation or group of operations resulting in a change in value, form or location of information, or in the determination of which one of several flow directions is to be followed.

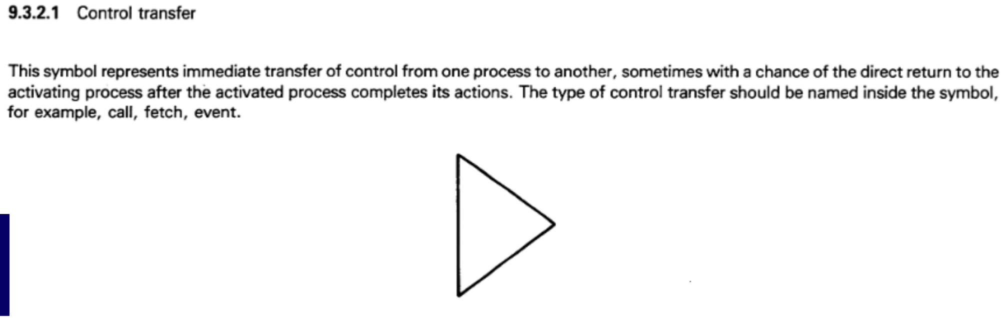
Syntax



### 9.2.2.1 Predefined process

This symbol represents a named process consisting of one or more operations or program steps that are specified elsewhere, for example, a subroutine, a module.

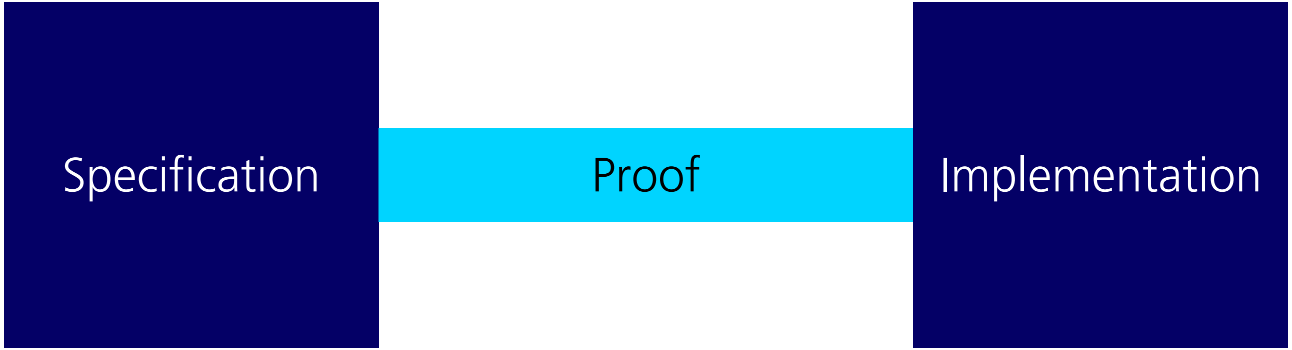
Semantics



### 9.3.2.1 Control transfer

This symbol represents immediate transfer of control from one process to another, sometimes with a chance of the direct return to the activating process after the activated process completes its actions. The type of control transfer should be named inside the symbol, for example, call, fetch, event.

# What is verification?

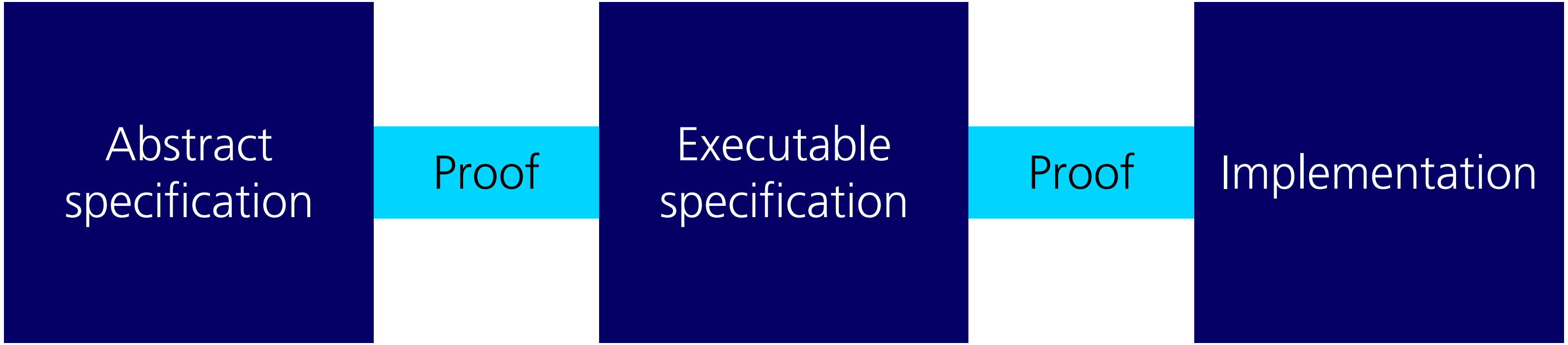


Specification

Proof

Implementation

# What is verification?



Abstract specification

Proof

Executable specification

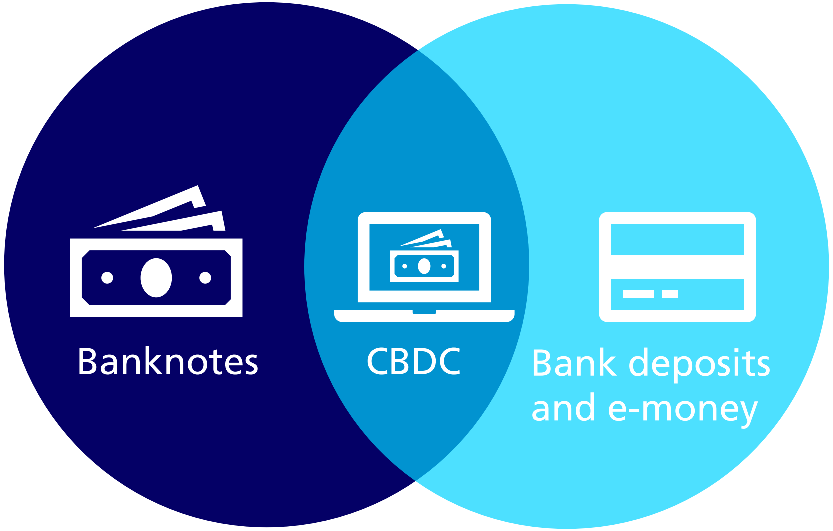
Proof

Implementation

# Formal Methods in practice

# Central Bank Digital Currency

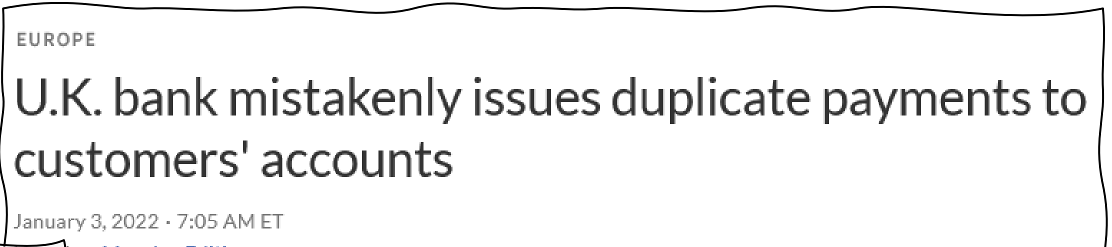
Issued by the central bank



Digital money

# Our customers

- central banks
- commercial/retail banks
- payment service providers

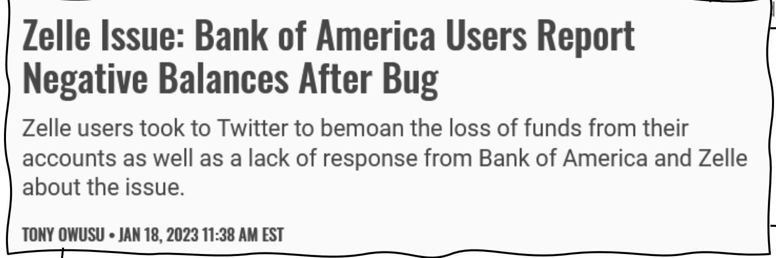


EUROPE

## U.K. bank mistakenly issues duplicate payments to customers' accounts

January 3, 2022 · 7:05 AM ET

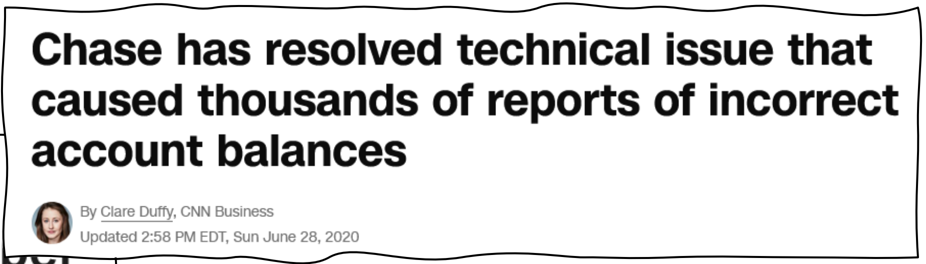
on Morning Edition



## Zelle Issue: Bank of America Users Report Negative Balances After Bug

Zelle users took to Twitter to bemoan the loss of funds from their accounts as well as a lack of response from Bank of America and Zelle about the issue.

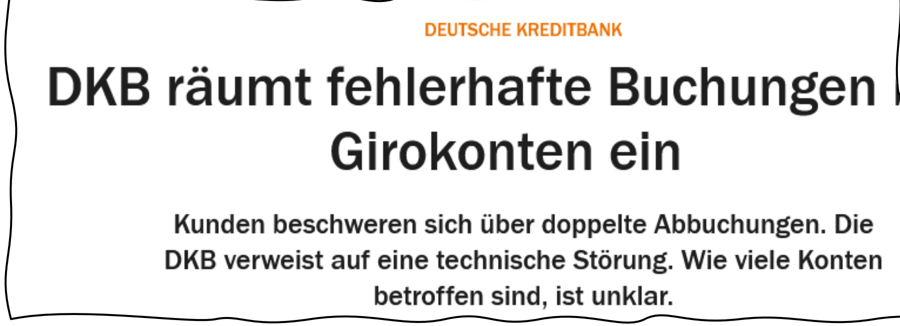
TONY OWUSU • JAN 18, 2023 11:38 AM EST



## Chase has resolved technical issue that caused thousands of reports of incorrect account balances

By Clare Duffy, CNN Business

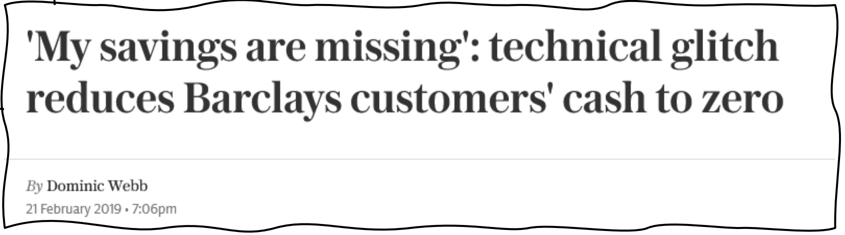
Updated 2:58 PM EDT, Sun June 28, 2020



DEUTSCHE KREDITBANK

## DKB räumt fehlerhafte Buchungen bei Girokonten ein

Kunden beschweren sich über doppelte Abbuchungen. Die DKB verweist auf eine technische Störung. Wie viele Konten betroffen sind, ist unklar.



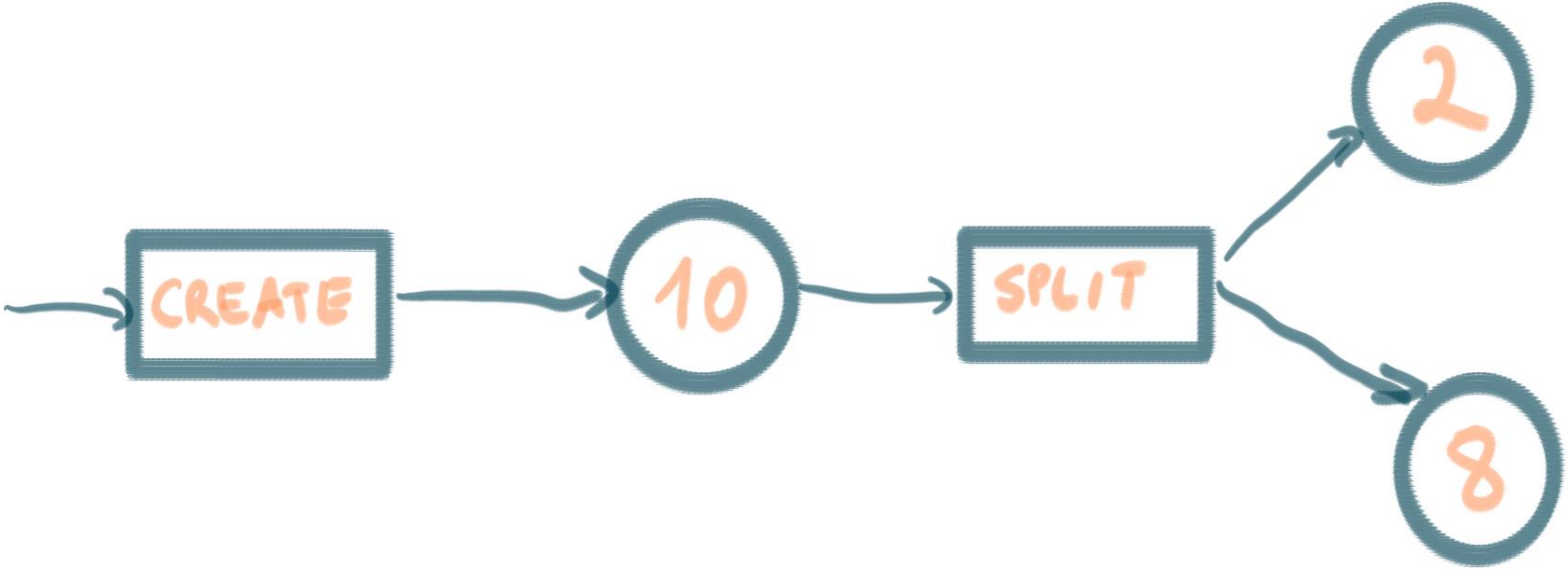
## 'My savings are missing': technical glitch reduces Barclays customers' cash to zero

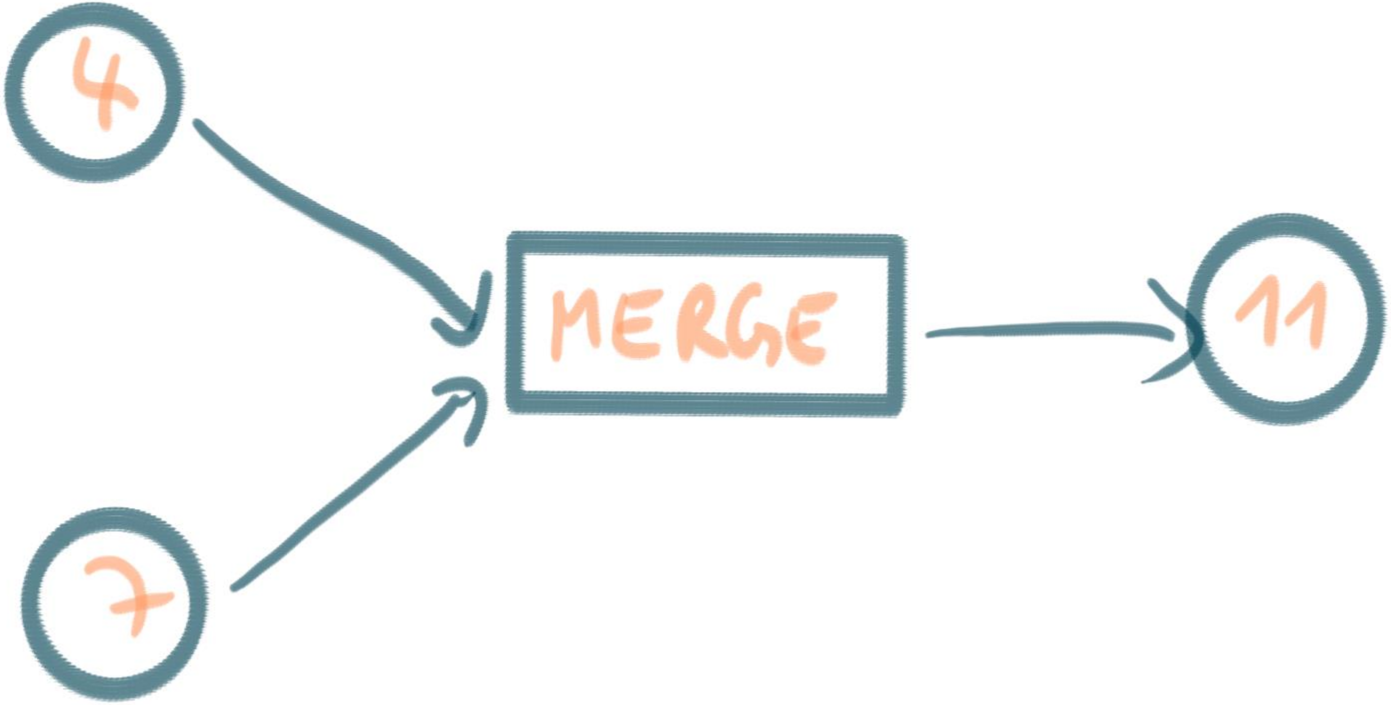
*By* Dominic Webb

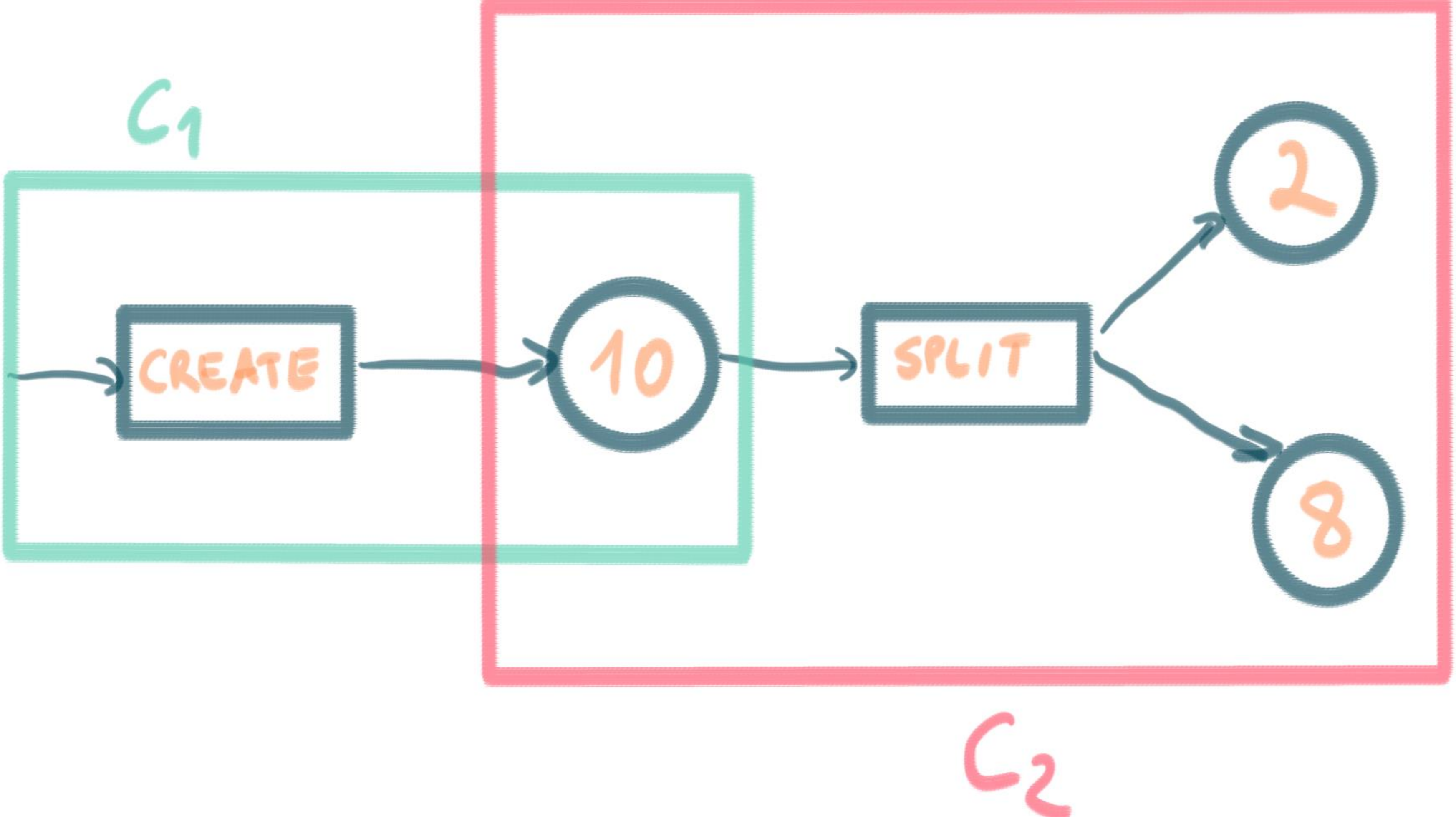
21 February 2019 • 7:06pm

# How money is represented in G+D Filia®

20€
Secret random number

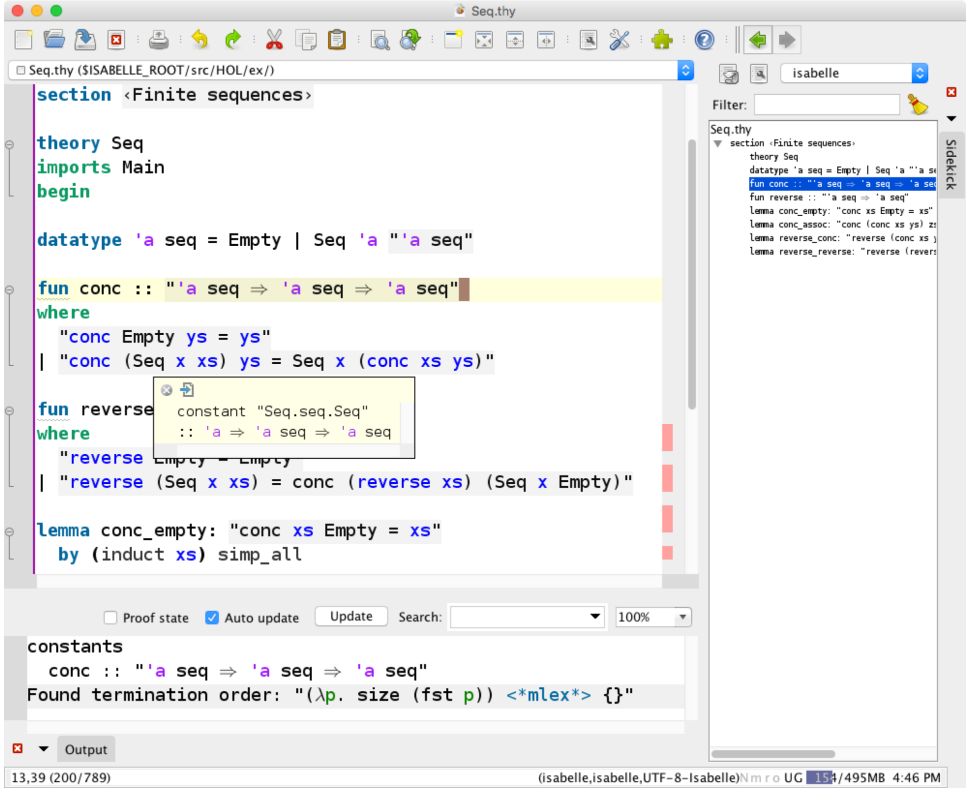

CREATE
10
SPLIT
2
8


4
7
MERGE
11


$C_1$
CREATE
10
SPLIT
2
8
$C_2$

$$\underbrace{\sum \text{utxo}}_{\text{circulation}} = \sum \text{CREATE} - \sum \text{DESTROY}$$

# Isabelle to the rescue!



Seq.thy

Seq.thy ($ISABELLE_ROOT/src/HOL/ex/)

```
section ‹Finite sequences›

theory Seq
imports Main
begin

datatype 'a seq = Empty | Seq 'a "'a seq"

fun conc :: "'a seq ⇒ 'a seq ⇒ 'a seq"
where
  "conc Empty ys = ys"
| "conc (Seq x xs) ys = Seq x (conc xs ys)"

fun reverse
where
  "reverse Empty = Empty"
| "reverse (Seq x xs) = conc (reverse xs) (Seq x Empty)"

lemma conc_empty: "conc xs Empty = xs"
  by (induct xs) simp_all
```

```
constant "Seq.seq.Seq"
:: 'a ⇒ 'a seq ⇒ 'a seq
```

☐ Proof state ☑ Auto update Update Search: 100%

```
constants
  conc :: "'a seq ⇒ 'a seq ⇒ 'a seq"
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
```

Output

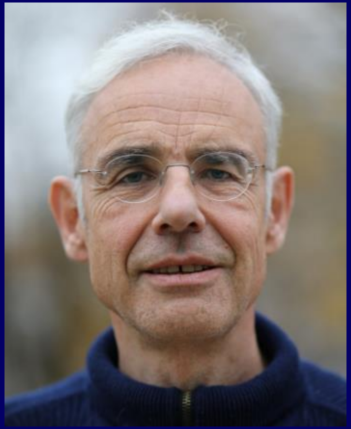
13,39 (200/789) (isabelle,isabelle,UTF-8-Isabelle)N m r o UG 154/495MB 4:46 PM

isabelle

Filter:

```
Seq.thy
▼ section ‹Finite sequences›
    theory Seq
    datatype 'a seq = Empty | Seq 'a "'a se
    fun conc :: "'a seq ⇒ 'a seq ⇒ 'a seq
    fun reverse :: "'a seq ⇒ 'a seq"
    lemma conc_empty: "conc xs Empty = xs"
    lemma conc_assoc: "conc (conc xs ys) zs
    lemma reverse_conc: "reverse (conc xs y
    lemma reverse_reverse: "reverse (revers
```

Sidekick



“Isabelle/HOL = Functional Programming + Logic”

# G+D Filia® in Isabelle/HOL

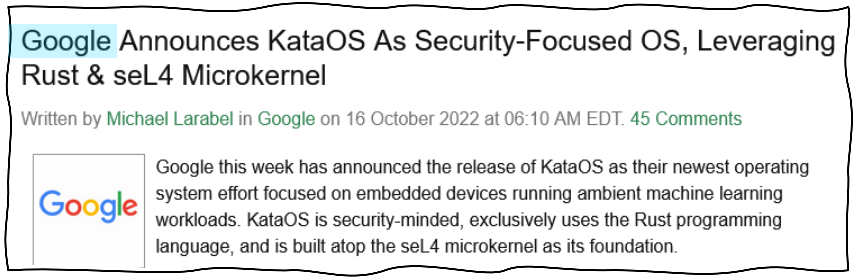
- mathematical model of “coins” and their evolution
- graph-theoretic considerations
- high-level correctness properties
- reference implementation (executable in Scala)

# Example: Money in circulation

```
definition graph_balance :: nat where
‹graph_balance = (ΣN ∈ unspent. value N)›

lemma graph_balance_alt_def:
  ‹graph_balance = ¦(Σc ∈ graph. value_difference c)¦›
```

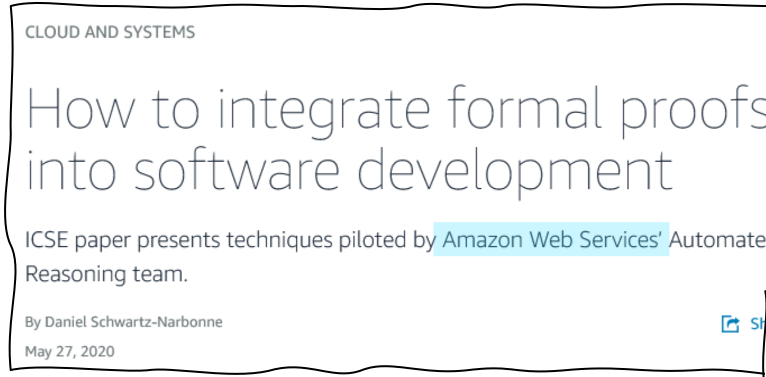
# It’s not just us



## Google Announces KataOS As Security-Focused OS, Leveraging Rust & seL4 Microkernel

Written by Michael Larabel in Google on 16 October 2022 at 06:10 AM EDT. 45 Comments

Google this week has announced the release of KataOS as their newest operating system effort focused on embedded devices running ambient machine learning workloads. KataOS is security-minded, exclusively uses the Rust programming language, and is built atop the seL4 microkernel as its foundation.



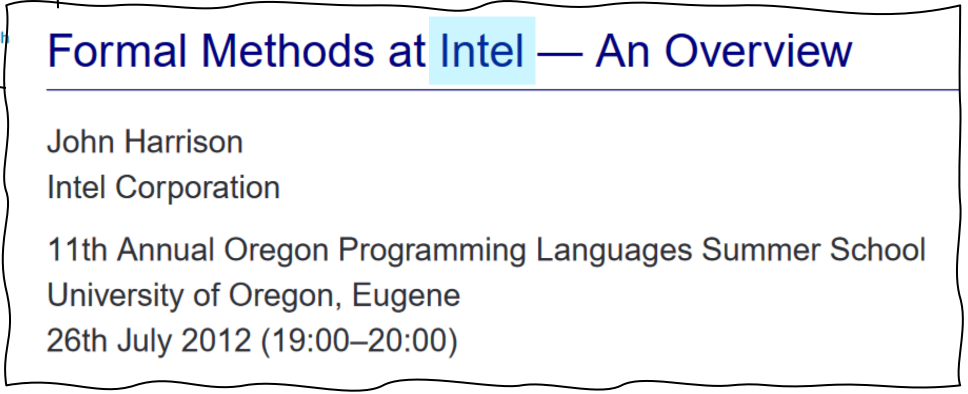
CLOUD AND SYSTEMS

## How to integrate formal proofs into software development

ICSE paper presents techniques piloted by Amazon Web Services’ Automated Reasoning team.

By Daniel Schwartz-Narbonne

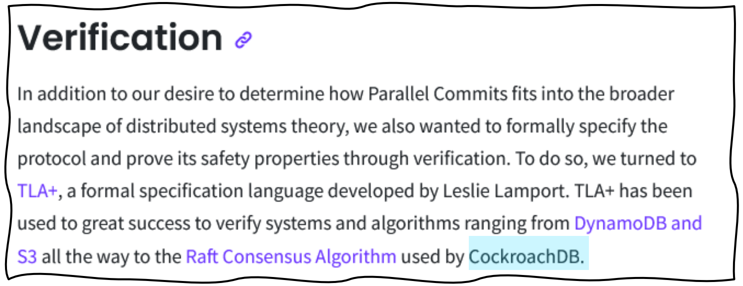
May 27, 2020



## Formal Methods at Intel — An Overview

John Harrison
Intel Corporation

11th Annual Oregon Programming Languages Summer School
University of Oregon, Eugene
26th July 2012 (19:00–20:00)



## Verification

In addition to our desire to determine how Parallel Commits fits into the broader landscape of distributed systems theory, we also wanted to formally specify the protocol and prove its safety properties through verification. To do so, we turned to TLA+, a formal specification language developed by Leslie Lamport. TLA+ has been used to great success to verify systems and algorithms ranging from DynamoDB and S3 all the way to the Raft Consensus Algorithm used by CockroachDB.

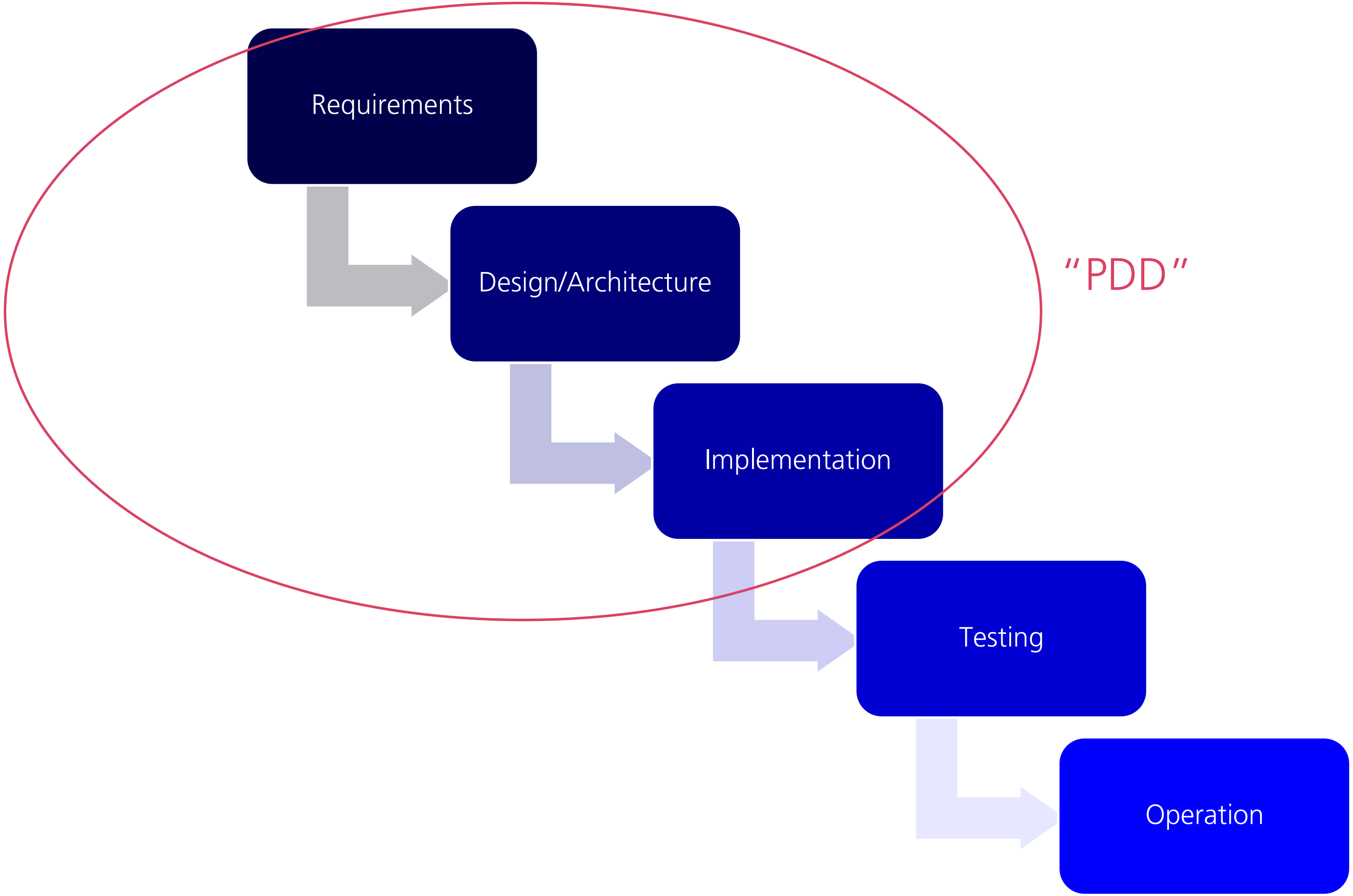
# Proof-Driven Development (PDD)

ELP 1215
ELP-1215
SP-SÃO CARLOS
MEIL
São Carlos Ambiental

AIRFRANCE
AMERICAN EXPRESS
FLYINGBLUE
2009
Hold Near Reader

# Designing a new feature

- Can the feature work correctly?
- Are there any undesirable feature interactions?
- How can we implement the feature?

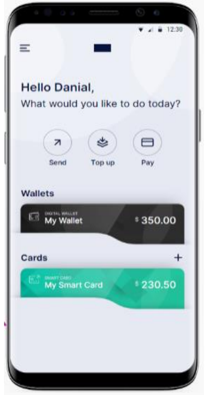

Requirements
Design/Architecture
Implementation
Testing
Operation
“PDD”

# PDD works for us

- we found some flaws in our initial design of a feature
  - … including a feature interaction bug
- after iterative improvement, the feature is now better than an alternative design
- changed the internal (simpler) data model, but we established a mapping
- feature has been shipped to production

# Roadmap

Giesecke+Devrient
Creating Confidence


Hello Danial,

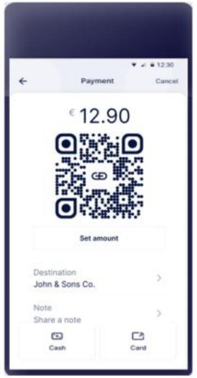
What would you like to do today?

Wallets

DIGITAL WALLET My Wallet $ 350.00

Cards +

SMART CARD My Smart Card $ 230.50



€ 12.90

Set amount

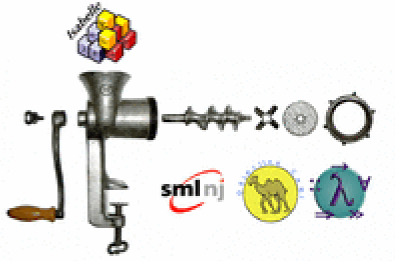
Destination John & Sons Co.

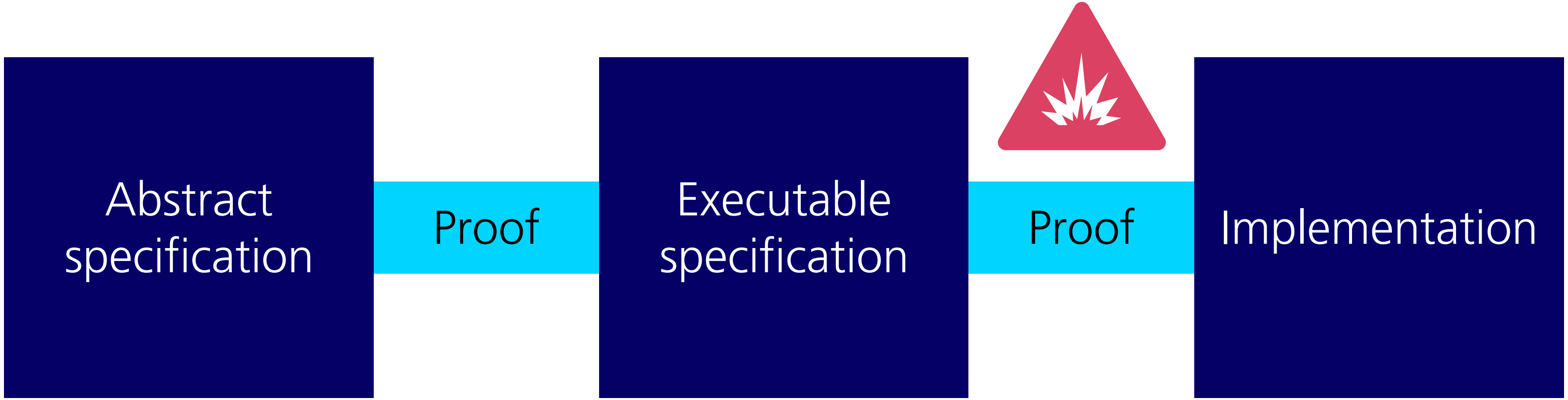
Note Share a note

# There’s always more to do …

- expanding the scope of our formalization
- adding model checking to our toolbox
- closing the gap between executable specification and implementation

# Closing the gap





Abstract specification

Proof

Executable specification

Proof

Implementation

# Questions?
# Answers!

Lars Hupel

https://lars.hupel.info

lars.hupel@gi-de.com

Giesecke+Devrient
Creating Confidence

# Image sources

- Edsger W. Dijskstra: Hamilton Richards, CC-BY-SA 3.0, https://commons.wikimedia.org/w/index.php?title=File:Edsger_Wybe_Dijkstra.jpg&oldid=710250942
- César A. Muñoz: https://shemesh.larc.nasa.gov/people/cam/
- BPMN: Mikelo Skarabo, CC-BY-SA 4.0, https://commons.wikimedia.org/w/index.php?title=File:BPMN-AProcessWithNormalFlow.svg&oldid=734511959