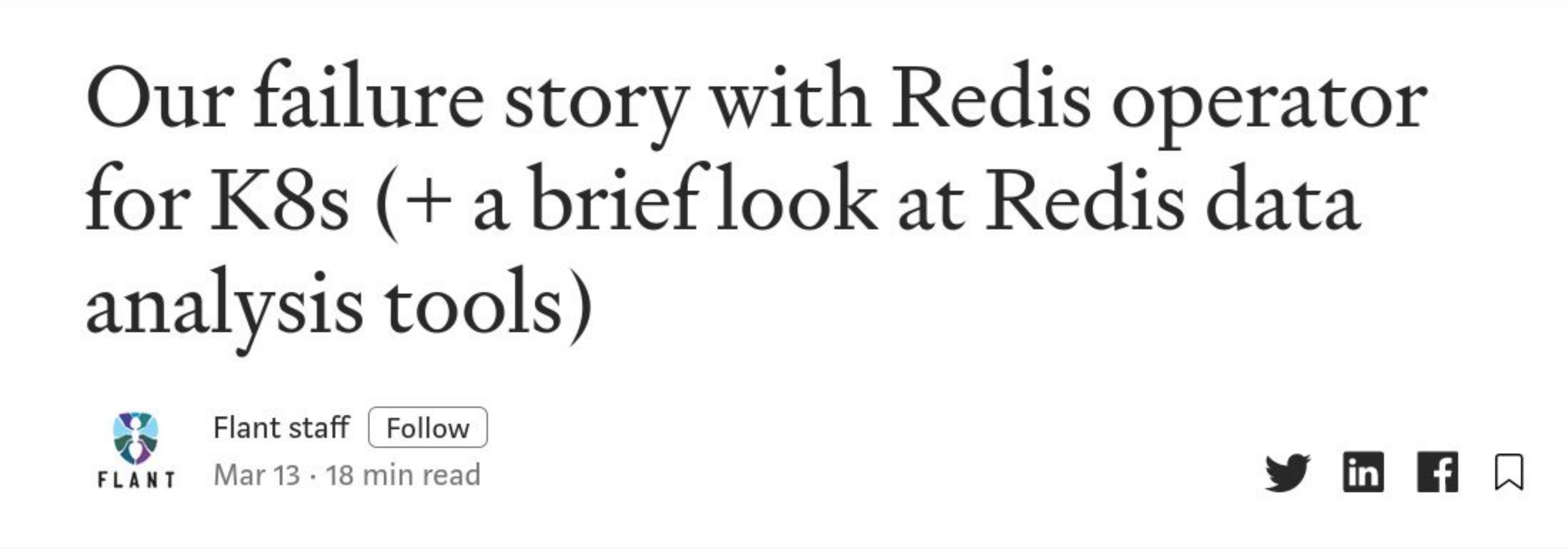
# Kubernetes Operators: Safety First Through Model Checkers

Neven Miculinic
grid.ai

## Motivation

With kubernetes building operators is easy...right?

## Motivation

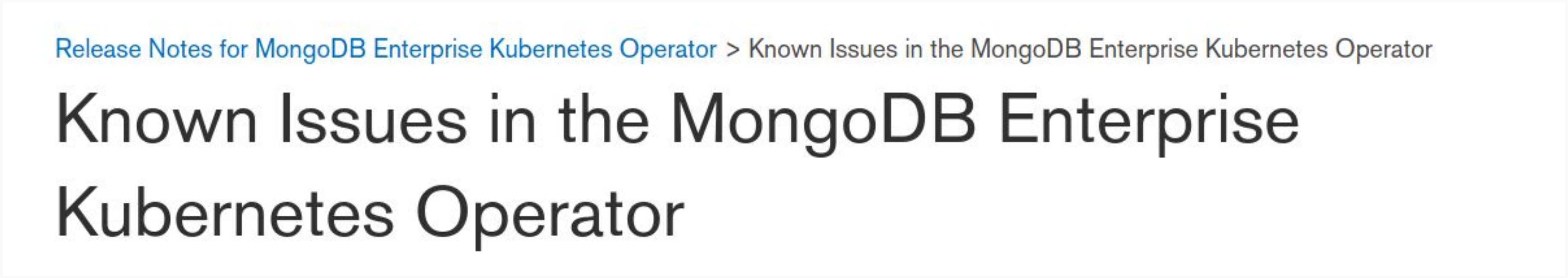


Our failure story with Redis operator for K8s (+ a brief look at Redis data analysis tools)

Flant staff Follow

Mar 13 · 18 min read

## Motivation



Release Notes for MongoDB Enterprise Kubernetes Operator > Known Issues in the MongoDB Enterprise Kubernetes Operator

Known Issues in the MongoDB Enterprise Kubernetes Operator

## Motivation - MongoDB operator

### Difficulties with Updates

In some cases, the Kubernetes Operator can **stop receiving change events**. As this problem is **hard to reproduce**, the recommended workaround is to delete the operator pod. Kubernetes starts the new Kubernetes Operator automatically and starts working correctly:

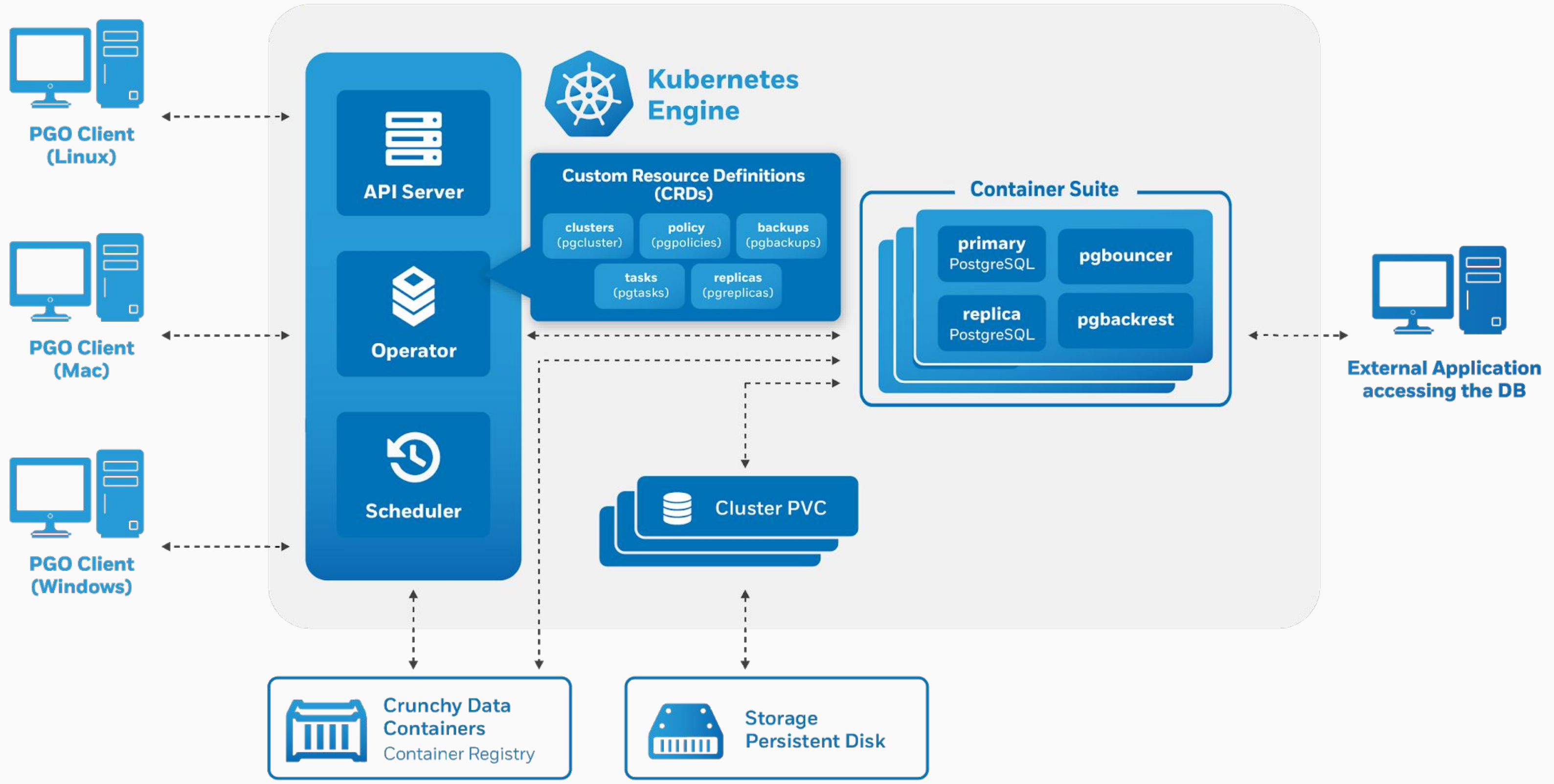
## Motivation - MongoDB operator

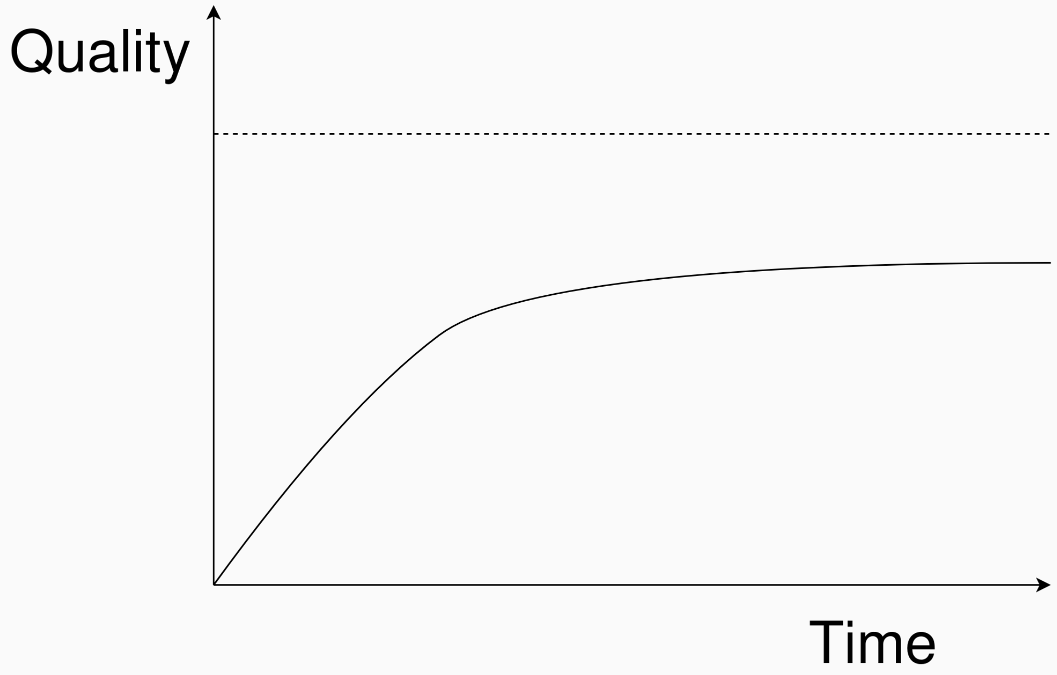
The Kubernetes Operator will not be able to apply the **following change** on a MongoDB Deployment **simultaneously**:

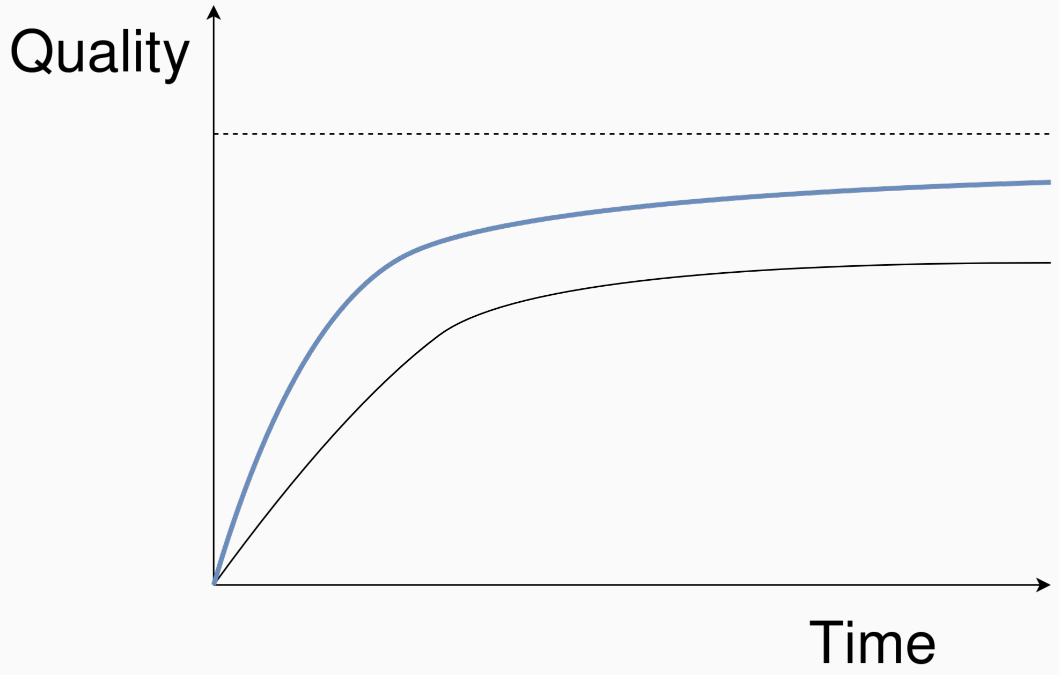
- The TLS configuration is disabled (security.tls.enabled: false)
- The number of members in a Replica Set is increased

If both operations are applied at the same time, the MongoDB Resource could go into a **unrecoverable state**.

## Tools


PGO Client (Linux)
PGO Client (Mac)
PGO Client (Windows)
API Server
Operator
Scheduler
Kubernetes Engine
Custom Resource Definitions (CRDs)
clusters (pgcluster)
policy (pgpolicies)
backups (pgbackups)
tasks (pgtasks)
replicas (pgreplicas)
Container Suite
primary PostgreSQL
pgbouncer
replica PostgreSQL
pgbackrest
External Application accessing the DB
Cluster PVC
Crunchy Data Containers
Container Registry
Storage Persistent Disk

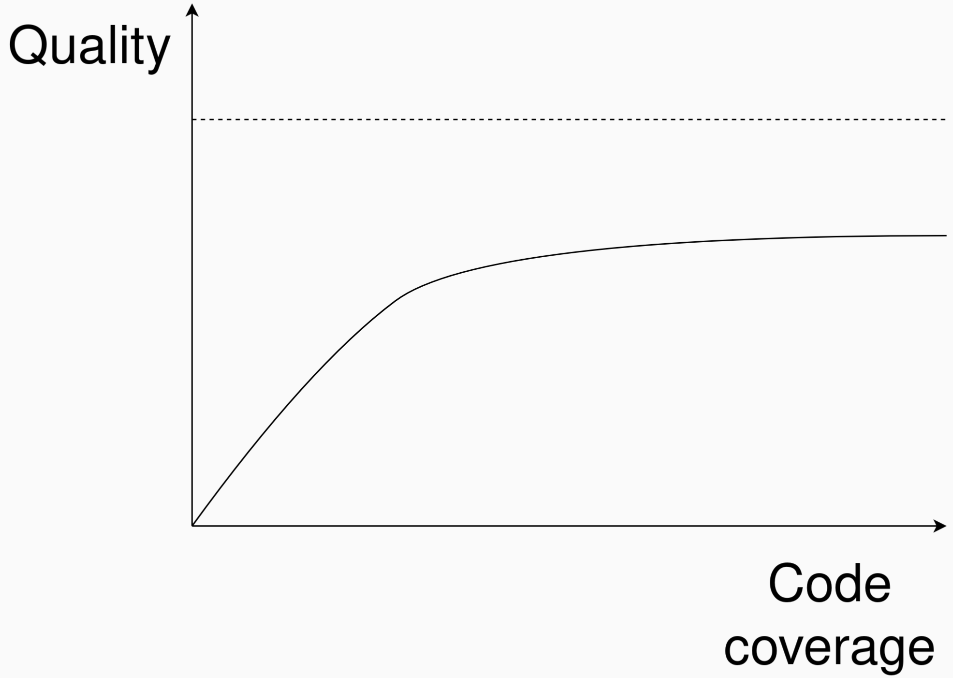

Quality
Time



Better:

- Tools
- Process
- Externalities

## Utility theory



- Unit tests
- Law of diminishing marginal utility

## Conventional approach

- Deep design review
- Static code analysis
- Stress testing
- Fault-injection testing
- Unit tests
- Code reviews
- ...

## A “New” tool appears

Model checkers:

- We specify the system model, its transitions functions, invariants, rules and properties we want checking
- The system searches the whole solution space for contradictions

## Wrong tool for the job

- Real time constraints
- It’s better expressing safety or liveness properties than real time constraints

## Industry usage -- AWS

Use of Formal Methods at Amazon Web Services

2014-09-29

## Industry usage -- AWS

Adoption at AWS
A crime thriller!

Starting
Chris Newcombe as C.N.
Tim Rath as T.R.

## Industry usage -- AWS

“To a first approximation, we can say that accidents are almost always the result of incorrect estimates of the likelihood of one or more things.” -C. Michael Holloway, NASA

## Industry usage -- AWS

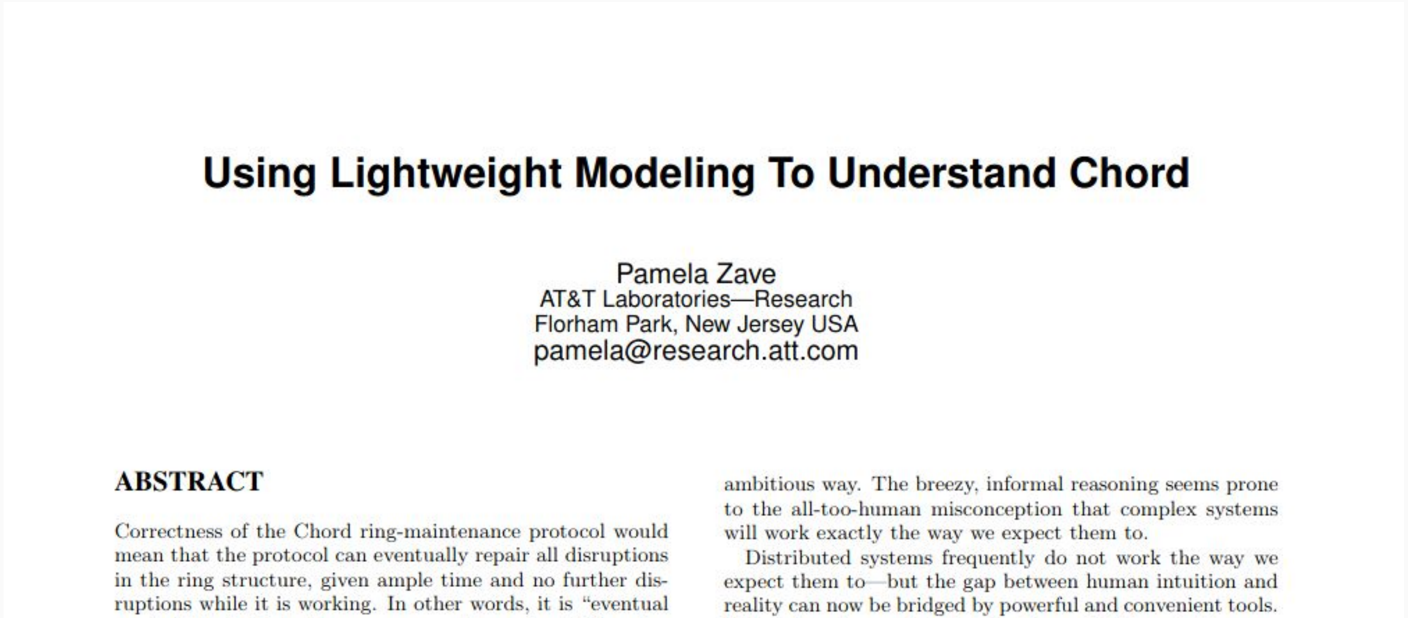
- C.N. was dissatisfied with the quality
- C.N. dismissed formal methods due to myths

## Misconceptions

- Takes long time to learn
- Only a small fraction of real-life problems fit this paradigm
- A low return on investment
- They are impractical

## Industry usage -- AWS

- A mystic paper appears
- ’10-year test of time’ award at SIGCOMM 2011



# Using Lightweight Modeling To Understand Chord

Pamela Zave
AT&T Laboratories—Research
Florham Park, New Jersey USA
pamela@research.att.com

## ABSTRACT

Correctness of the Chord ring-maintenance protocol would mean that the protocol can eventually repair all disruptions in the ring structure, given ample time and no further disruptions while it is working. In other words, it is “eventual

ambitious way. The breezy, informal reasoning seems prone to the all-too-human misconception that complex systems will work exactly the way we expect them to.

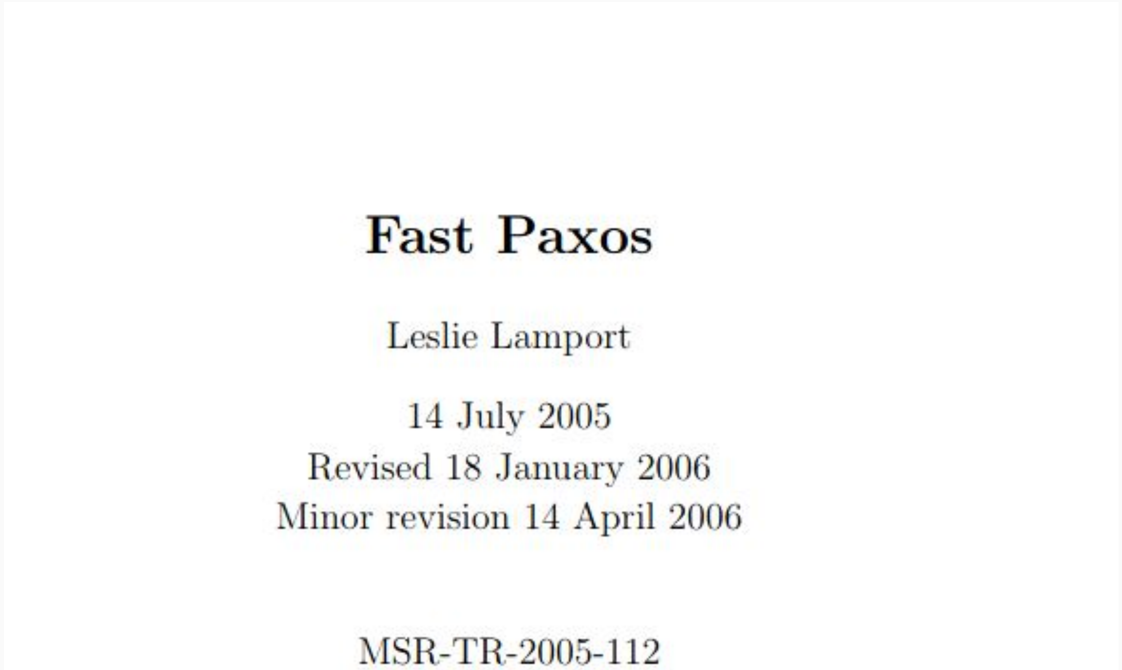
Distributed systems frequently do not work the way we expect them to—but the gap between human intuition and reality can now be bridged by powerful and convenient tools.

## Industry usage -- AWS

- C.N. investigates the first tool, a unsung hero called Alloy
- Limited expressivity

## Industry usage -- AWS

- C.N. looked further for the holy grail



Fast Paxos

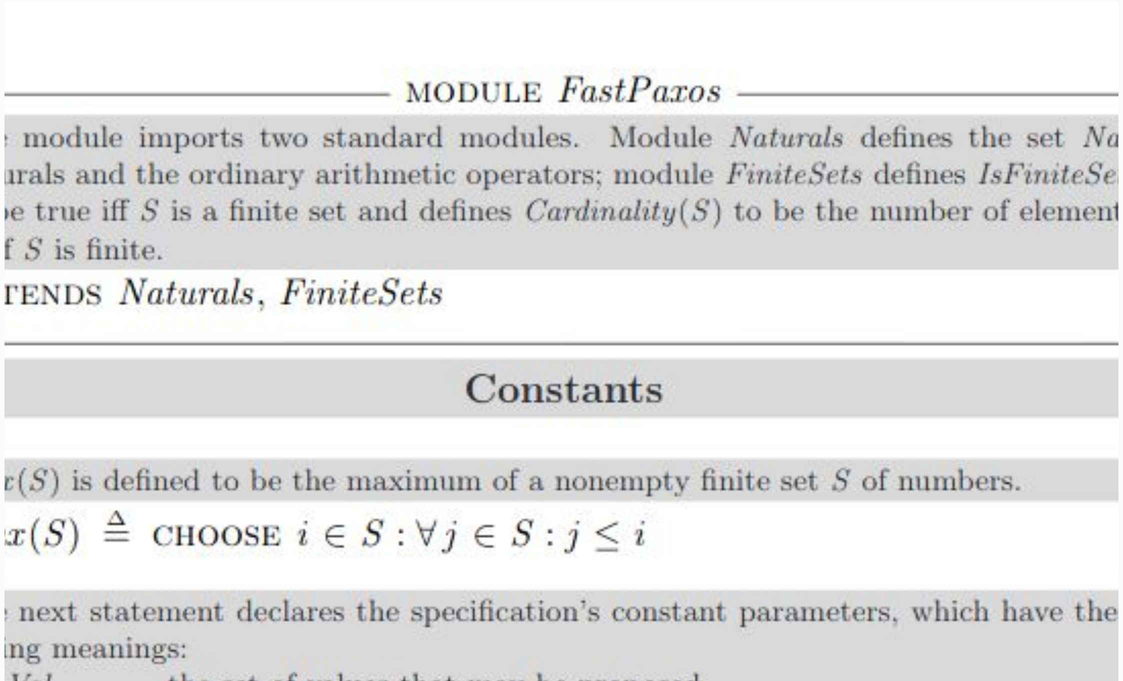
Leslie Lamport

14 July 2005
Revised 18 January 2006
Minor revision 14 April 2006

MSR-TR-2005-112

## Industry usage -- AWS

- And found the TLA+ spec at the end...



MODULE *FastPaxos*

module imports two standard modules. Module *Naturals* defines the set *Na*
urals and the ordinary arithmetic operators; module *FiniteSets* defines *IsFiniteSe*
e true iff $S$ is a finite set and defines *Cardinality*$(S)$ to be the number of element
f $S$ is finite.

TENDS *Naturals*, *FiniteSets*

**Constants**

$x(S)$ is defined to be the maximum of a nonempty finite set $S$ of numbers.

$x(S) \triangleq$ CHOOSE $i \in S : \forall j \in S : j \leq i$

next statement declares the specification's constant parameters, which have the
ing meanings:

*Val* the set of values that may be proposed

## Industry usage -- AWS

## Industry usage -- AWS

## Industry usage -- AWS

## Industry usage -- AWS

DynamoDB

amazon web services™

## Industry usage -- AWS

- T.R. enters the story with DynamoDB
- Performed classical testing approach
- A couple of weeks with TLA+
- Found a bug in his design requiring 35 high level steps + 2 more bugs

## Industry usage -- AWS

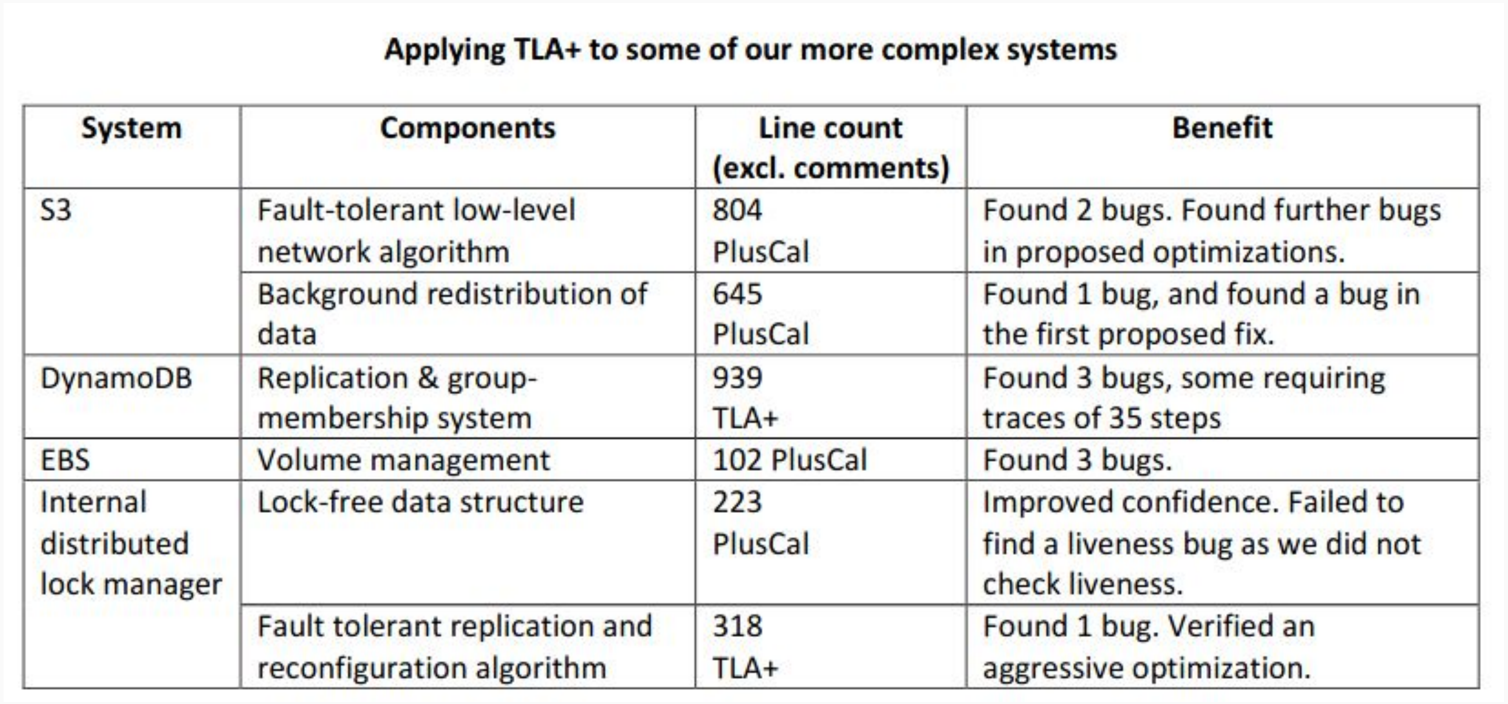
## Industry usage -- AWS

“Debugging designs”

## Industry usage -- AWS

“Exhaustively testable pseudo-code”

# Industry usage -- AWS

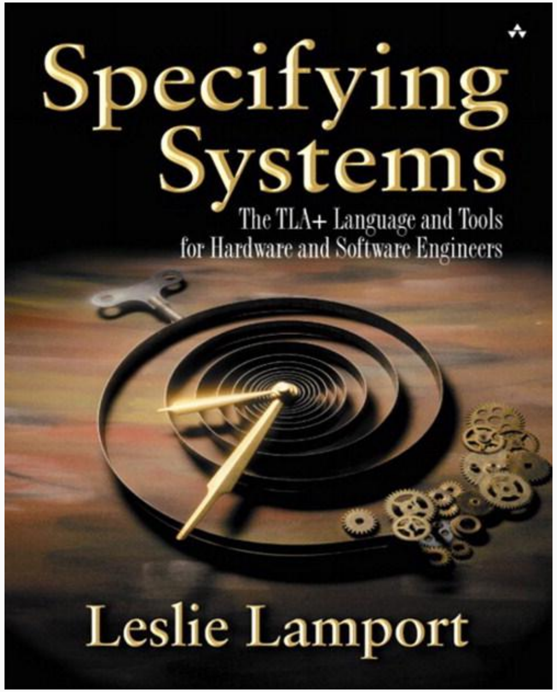


**Applying TLA+ to some of our more complex systems**

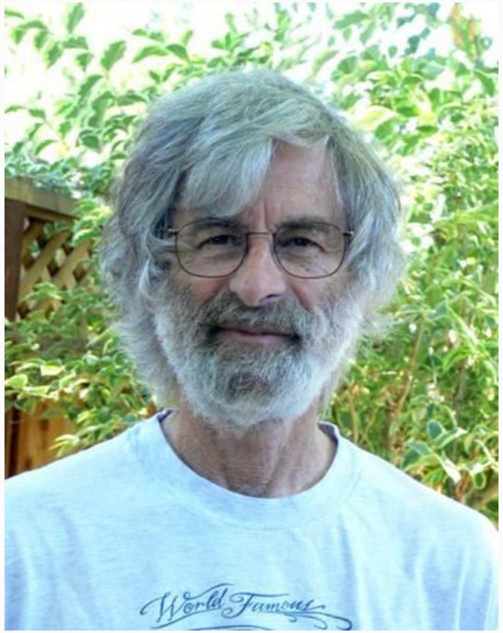
| System | Components | Line count (excl. comments) | Benefit |
|---|---|---|---|
| S3 | Fault-tolerant low-level network algorithm | 804 PlusCal | Found 2 bugs. Found further bugs in proposed optimizations. |
| | Background redistribution of data | 645 PlusCal | Found 1 bug, and found a bug in the first proposed fix. |
| DynamoDB | Replication & group-membership system | 939 TLA+ | Found 3 bugs, some requiring traces of 35 steps |
| EBS | Volume management | 102 PlusCal | Found 3 bugs. |
| Internal distributed lock manager | Lock-free data structure | 223 PlusCal | Improved confidence. Failed to find a liveness bug as we did not check liveness. |
| | Fault tolerant replication and reconfiguration algorithm | 318 TLA+ | Found 1 bug. Verified an aggressive optimization. |

## Industry usage -- Microsoft

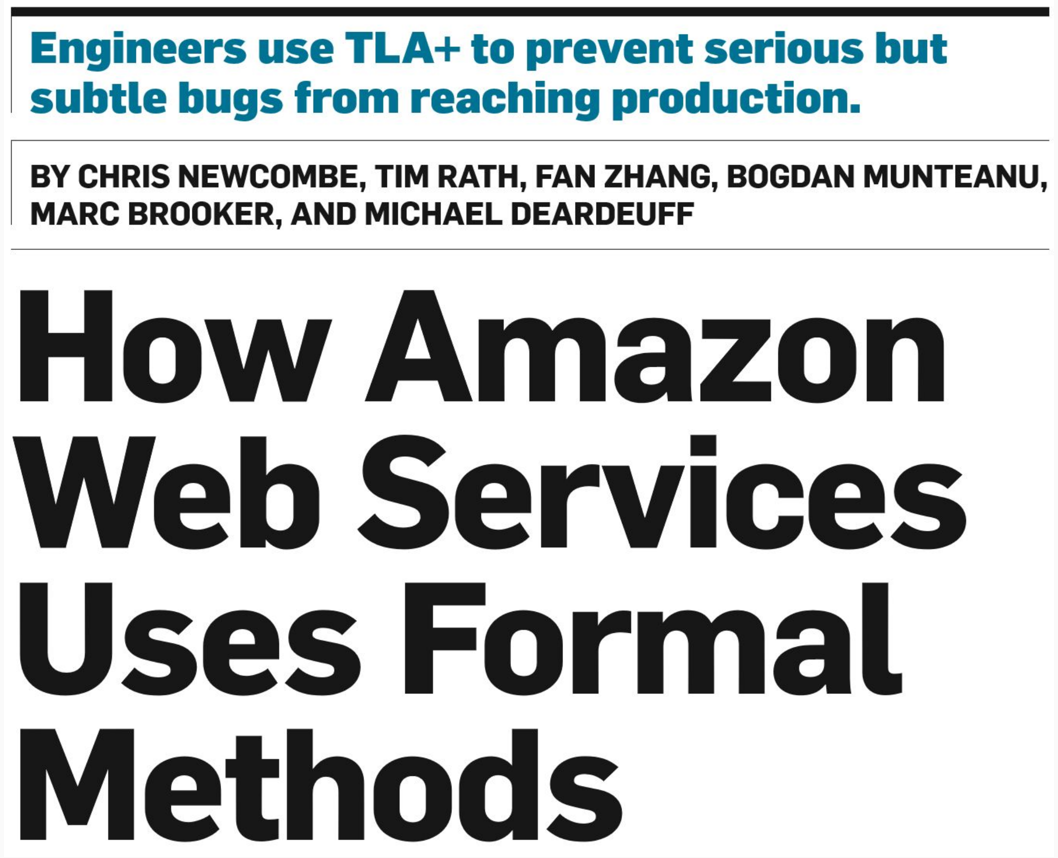
- Leslie Lamport & Microsoft Research


Specifying Systems
The TLA+ Language and Tools for Hardware and Software Engineers
Leslie Lamport

2002



## Industry usage -- Microsoft


Engineers use TLA+ to prevent serious but subtle bugs from reaching production.

BY CHRIS NEWCOMBE, TIM RATH, FAN ZHANG, BOGDAN MUNTEANU, MARC BROOKER, AND MICHAEL DEARDEUFF

How Amazon Web Services Uses Formal Methods

- April 2015
- December 26th email

## Industry usage -- Microsoft

- Service Fabric
- Azure Batch
- Azure Storage
- Azure Networking
- Azure IoT Hub

- uncovered a safety violation even in our most confident implementation

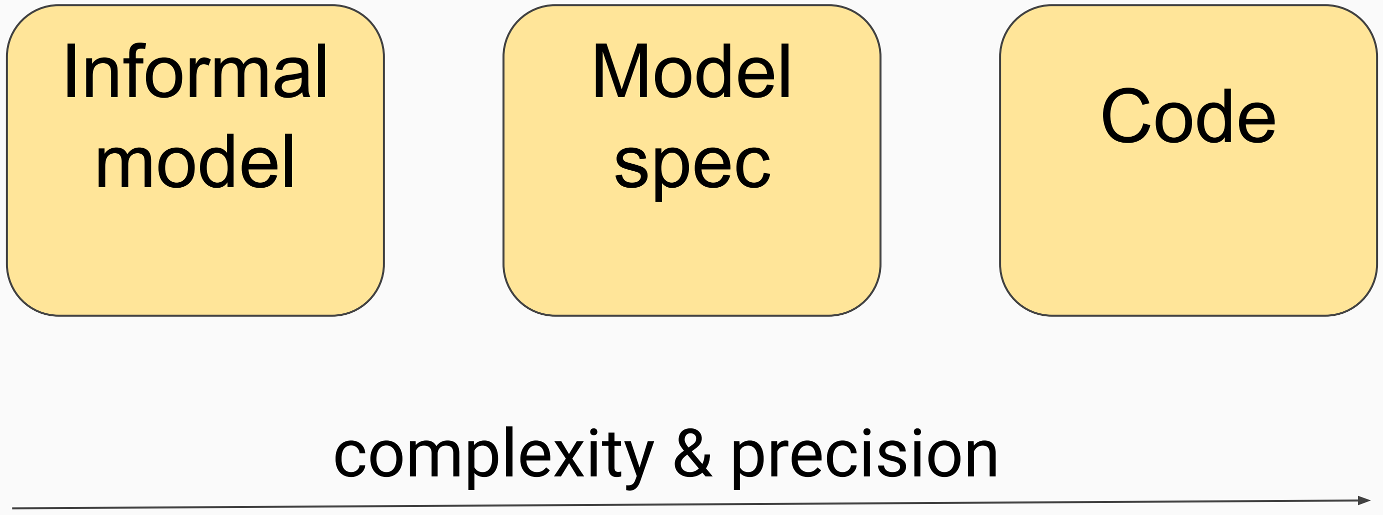
## Industry usage

- Intel
- Microsoft
    - XBOX
    - Azure
- AWS
- OpenComRTOS → 10x code reduction with TLA+ help
- ElasticSearch 7.0+
- MongoDB
- ...

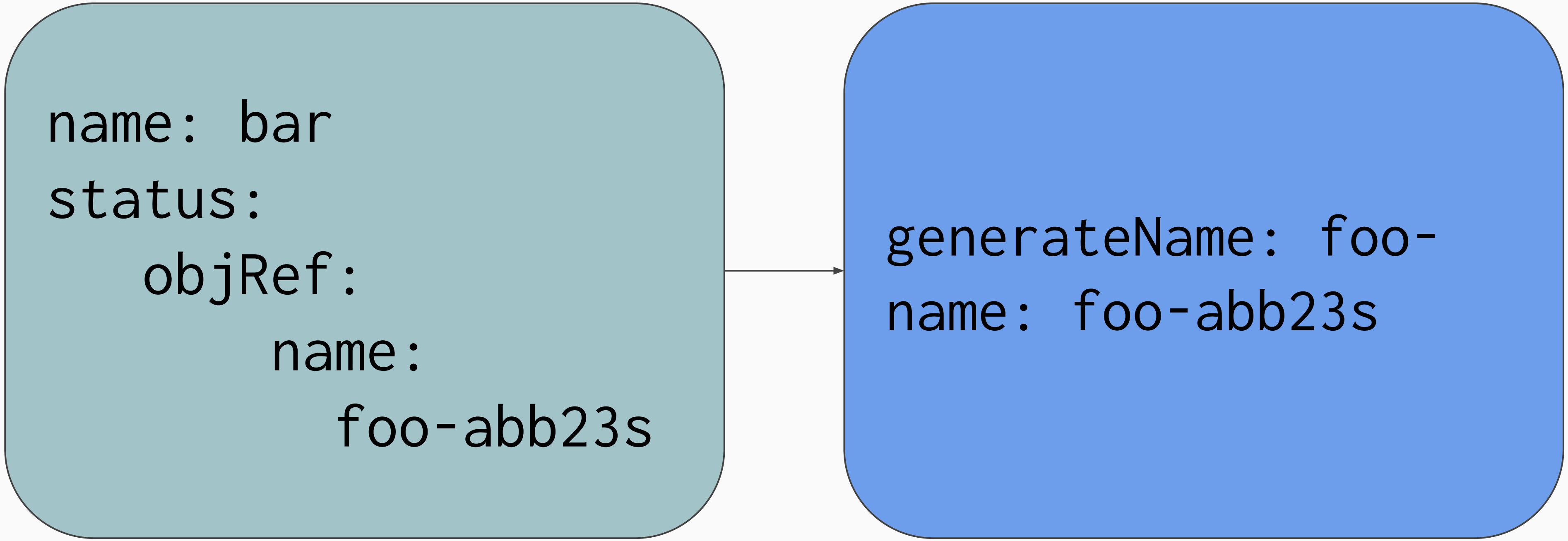
## Benefits

- Improved design quality
- Less bugs
- Performance optimizations
- improved time-to-market
- Documentation

## Benefits



# Example TLA+



```
name: bar
status:
   objRef:
      name:
         foo-abb23s
```

```
generateName: foo-
name: foo-abb23s
```

## Example TLA+ (PlusCAL)

```
---- MODULE ns ----
EXTENDS TLC,Integers,Sequences
```

## Example TLA+ (PlusCAL)

```
(*--algorithm ns
variables
  fooObjs = <<>>,
  barObj = [name |-> "foo", created
|-> FALSE],
```

## Example TLA+ (PlusCAL)

- Safety properties
- Liveness properties

## Example TLA+ (PlusCAL)

```
define
  LimitedFooObjs == Len(fooObjs) <= 1
end define;
```

## Example TLA+ (PlusCAL)

```
begin
  Start: while ~barObj.created do
      ...
  end while;
  assert(Len(fooObjs) = 1);
end algorithm;*)
```

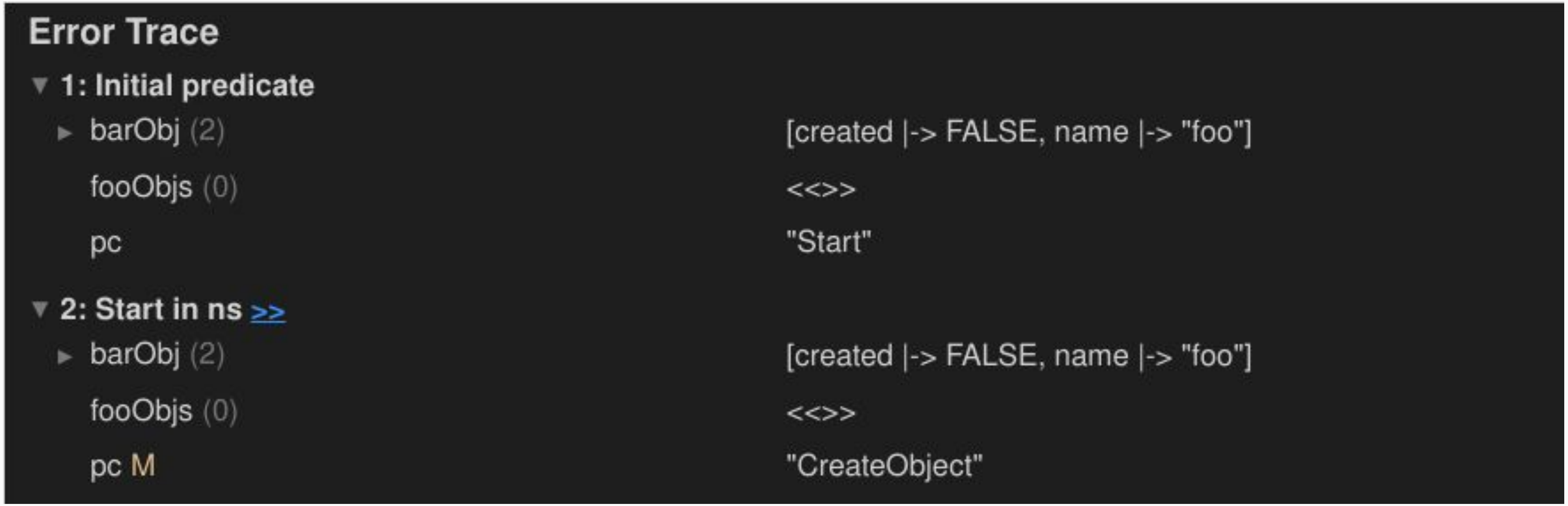
## Example TLA+ (PlusCAL)

```
either
    CreateObject: fooObjs := Append(fooObjs,
            [generateName |-> barObj.name \o "-"]);
    either
        MarkCreated: barObj.created := TRUE;
    or
        RebootDuringMarkingCreation: skip;
    end either;
or
    RebootDuringCreation: skip;
end either;
```

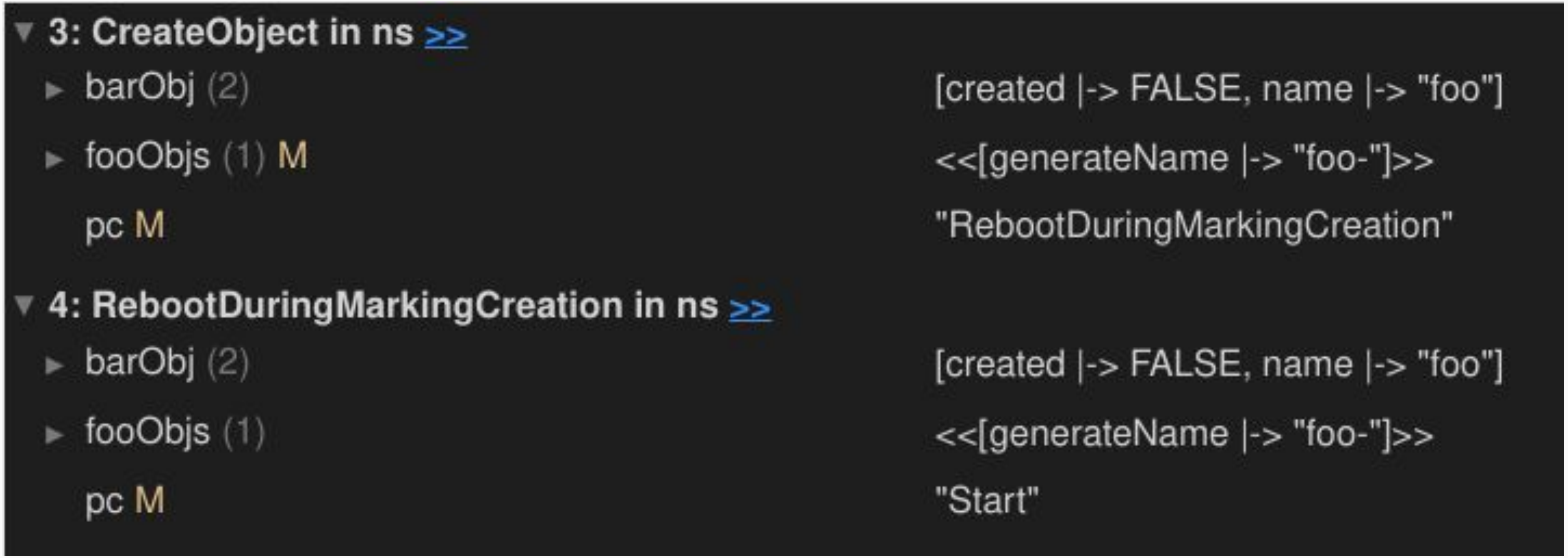
## Example TLA+ (PlusCAL)

```
either
    CreateObject: fooObjs := Append(fooObjs,
            [generateName |-> barObj.name \o "-"]);
    either
        MarkCreated: barObj.created := TRUE;
    or
        RebootDuringMarkingCreation: skip;
    end either;
or
    RebootDuringCreation: skip;
end either;
```

## Example TLA+ (PlusCAL)

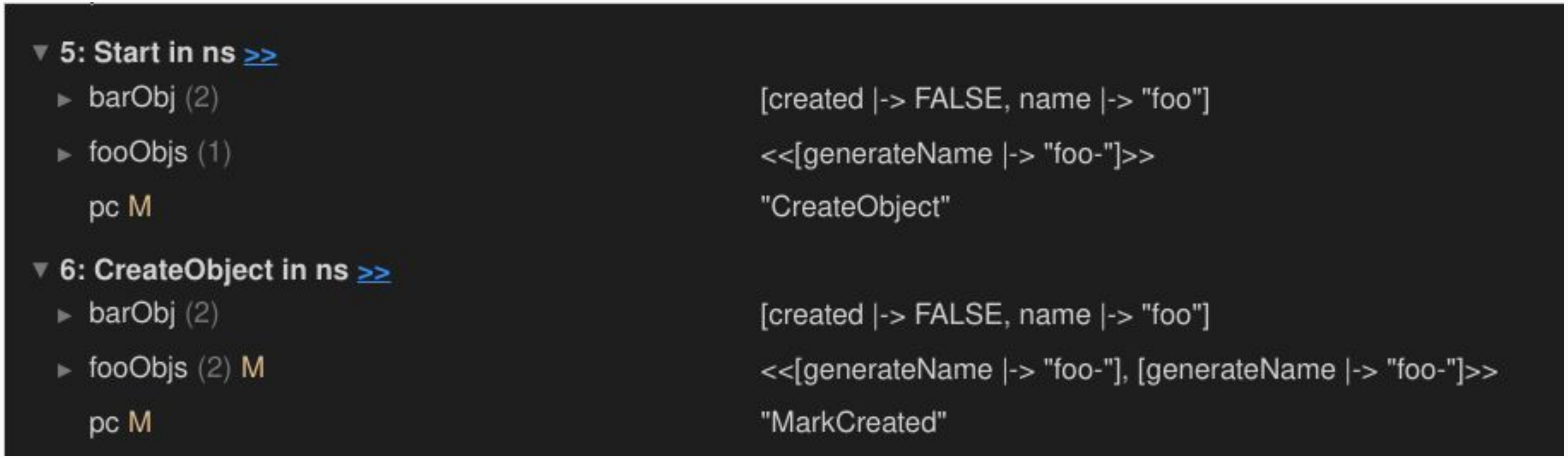

```
Error Trace
▼ 1: Initial predicate
  ▸ barObj (2)                [created |-> FALSE, name |-> "foo"]
    fooObjs (0)               <<>>
    pc                        "Start"
▼ 2: Start in ns >>
  ▸ barObj (2)                [created |-> FALSE, name |-> "foo"]
    fooObjs (0)               <<>>
    pc M                      "CreateObject"
```

## Example TLA+ (PlusCAL)



```
▼ 3: CreateObject in ns >>
  ► barObj (2)          [created |-> FALSE, name |-> "foo"]
  ► fooObjs (1) M       <<[generateName |-> "foo-"]>>
    pc M                "RebootDuringMarkingCreation"
▼ 4: RebootDuringMarkingCreation in ns >>
  ► barObj (2)          [created |-> FALSE, name |-> "foo"]
  ► fooObjs (1)         <<[generateName |-> "foo-"]>>
    pc M                "Start"
```

## Example TLA+ (PlusCAL)



```
▼ 5: Start in ns >>
  ▸ barObj (2)          [created |-> FALSE, name |-> "foo"]
  ▸ fooObjs (1)         <<[generateName |-> "foo-"]>>
    pc M                "CreateObject"
▼ 6: CreateObject in ns >>
  ▸ barObj (2)          [created |-> FALSE, name |-> "foo"]
  ▸ fooObjs (2) M       <<[generateName |-> "foo-"], [generateName |-> "foo-"]>>
    pc M                "MarkCreated"
```

## Thanks!

Q&A

Neven Miculinic

neven.miculinic@gmail.com

## References

Use of Formal Methods at Amazon Web Services Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, Michael Deardeu
Why Amazon Chose TLA +
Industrial use of TLA+ (Lamport)
Our failure story with Redis operator for K8s (+ a brief look at Redis data analysis tools)
Known Issues in the MongoDB Enterprise Kubernetes Operator — MongoDB Kubernetes Operator 1.4
How Amazon Web Services Uses Formal Methods | April 2015 | Communications of the ACM
TLA+ Conf: Program
(PDF) A Brief History of Formal Methods