# Miscomputation

## Learning to live with errors

HaPoC
2015

**Tomas Petricek**, University of Cambridge
http://tomasp.net | tomas@tomasp.net | @tomaspetricek

# Miscomputation

*If trials of three or four simple cases have been made, and are found to agree with the results given by the engine, it is scarcely possible that there can be any error (...).*

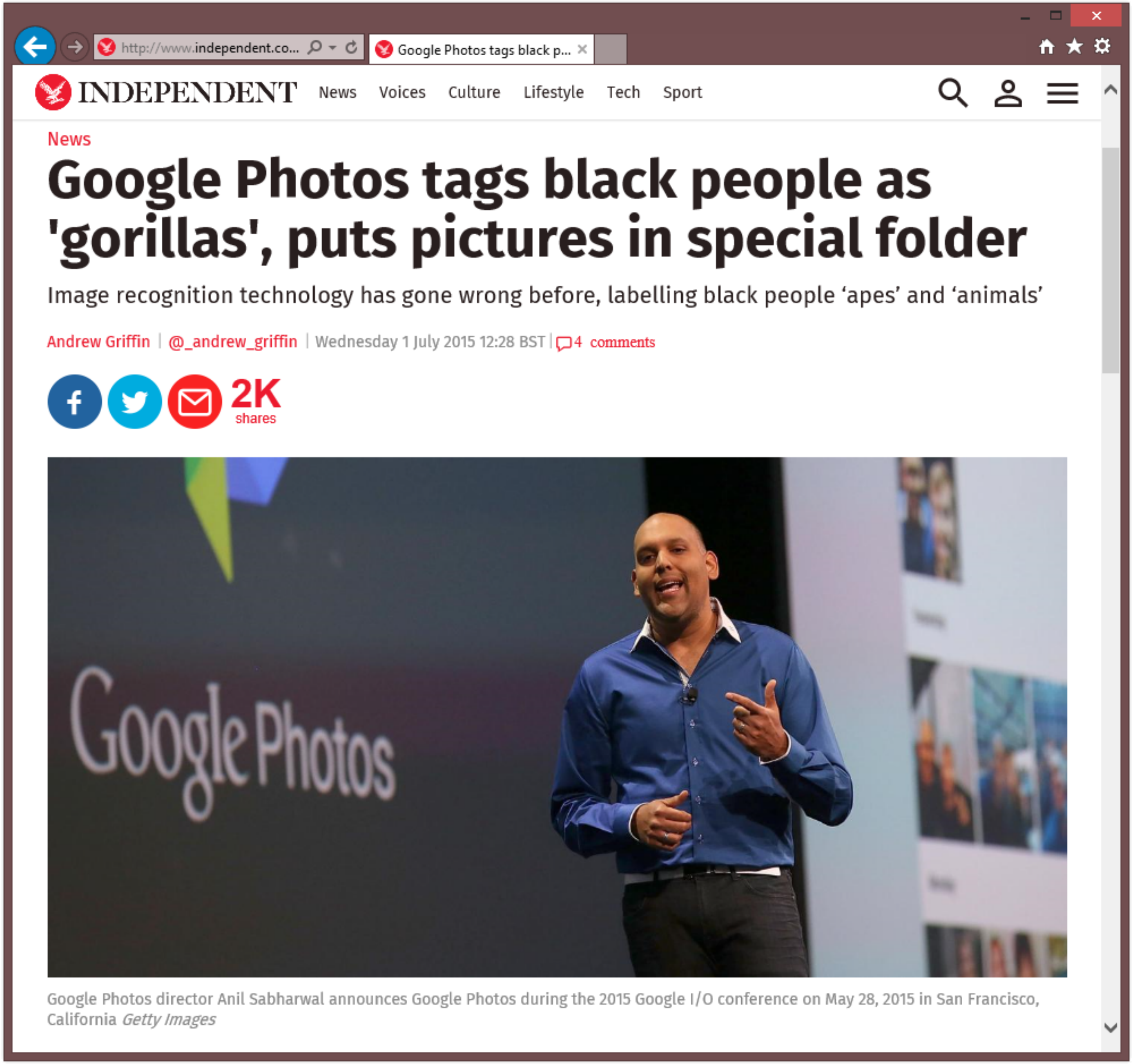
Charles Babbage, On the mathematical powers of the calculating engine (1837)

*Errors in coding were only gradually recognized to be a significant problem: a typical early comment was that of Miller [circa 1949], who wrote that such errors, along with hardware faults, could be "expected, in time, to become infrequent".*

Mark Priestley, Science of Operations (2011)

# Examples of miscomputations

- Invalid specification
- Invalid implementation
- No specification available
- Physical error condition



News

# Google Photos tags black people as 'gorillas', puts pictures in special folder

Image recognition technology has gone wrong before, labelling black people 'apes' and 'animals'

Andrew Griffin | @_andrew_griffin | Wednesday 1 July 2015 12:28 BST | 4 comments

Google Photos director Anil Sabharwal announces Google Photos during the 2015 Google I/O conference on May 28, 2015 in San Francisco, California *Getty Images*

# Living with errors

Coq

# Errors as a curse

# Algol research programme (1960s)

*One of the goals (..) was to utilize the resources of logic to increase the confidence (..) in the correctness of a program. As McCarthy had put it, "[instead] of debugging a program, one should prove that it meets its specifications (...)".*

Mark Priestley, Science of Operations (2011)

# Dependently typed programming (2010s)

Tries to make proof a part of programming practice

> *[T]oday most people who write software (...) assume that the costs of formal program verification outweigh the benefits. The purpose of this book is to convince you that the technology of program verification is mature enough today (...).*

Adam Chlipala, Certified Programming
with Dependent Types (2013)

# Error as a curse

Dream for the last 50 years

Common point of view in academic programming language community

Mixed success in practice

# Errors as progress

# How Did Software Get So Reliable Without Proof?

C.A.R. Hoare

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

# Engineering approach

Solid engineering practices are often good enough.

Testing software is one such practice.

- **Tests** to rule out errors
- **Tests** to rule out regression errors
- **Tests** as a specification

# Test-Driven Development (TDD)

*[In TDD] we drive development with automated tests (...). We*

1. *write new code only if an automated test has failed*
2. *eliminate duplication.*

*These are two simple rules, but they generate complex individual and group behavior (...).*

Kent Beck, Test-Driven
Development by Example (2003)

# Test-Driven Development (TDD)

1. Introduce controlled isolated miscomputation
2. Eliminate miscomputation by writing more code

TDD incorporates miscomputation
as a part of the development cycle!

Tests serve as specification and documentation

# Errors as the unavoidable

# Erlang programming langauge

Created by Ericsson for telecomunications

Distributed long-running reliable systems

# Miscomputations in Erlang

- ***exceptions** occur when the run-time does not know what to do.*
- ***errors** occur when the programmer doesn't know what to do.*

Joe Armstrong, Programming reliable systems (2003)

Errors are expected. Specification does not cover all cases.

# Handling errors in Erlang

*What kind of code must the programmer write when they find an error? The philosophy is let some other process fix the error, but what does this mean for their code? The answer is **let it crash**.*

Joe Armstrong, Programming reliable systems (2003)

# Errors as the unavoidable

Miscomputation is a normal part of execution.

(Should we still call it miscomputation?)

# Errors as an inspiration

# Smalltalk ecosystem (1970s)

*[Smalltalk approach] to the design of programming languages [is] quite different from what was familiar in the Algol [programme].*

*Programming was not thought of as the task of constructing a linguistic entity, but rather as a process of working interactively with the semantic representation of the program, using text as one possible interface.*

Mark Priestley, Science of Operations (2011)

# Computation as interaction

Live coding environments for performing music

*In musical genres that are not notated so closely (...), there are no wrong notes – only notes that are more or less appropriate to the performance.*

Alan Blackwell and Nick Collins, The Programming Language as a Musical Instrument (2005

# Miscomputation in live coding

*[Live coders] may well prefer to accept the results of an imperfect execution. [They] might perhaps compensate for an unexpected result by manual intervention, or even accept the result as a serendipitous alternative to the original note.*

Alan Blackwell and Nick Collins, The Programming
Language as a Musical Instrument (2005)

# Errors as an inspiration

Make miscomputations more apparent.

Enable quick human intervention.

Not limited to live coded art performances!

# Summary

# Summary

**Not all miscomputation is bad**

(Is it still a miscomputation when it's expected?)

**Different research programmes**

(No approach is better in general)

http://tomasp.net | tomas@tomasp.net | @tomaspetricek