150 Years of Observability

First Iron Bridge
1780 – Coalbrookdale

Mary Poppendieck
mary@poppendieck.com
www.poppendieck.com

Photograph © Tom Poppendieck, Taken in 1989

Copyright © 2020 Poppendieck LLC
1934: De Havilland DH.86 Express – Miss HOBART
October 19, 1934: Disappeared in the Bass Strait near Melbourne

Wood and Canvas Construction
1952: De Havilland DH-106 Comet 1 – Yoke Peter

January 10, 1954: Crashed into the sea near Elba Island on takeoff from Rome

First Jet Aircraft, First Pressurized Cabin
A Classic Case of Forensic Engineering


Cause: Metal fatigue starting at the corners of the square windows.
Flight Data Recorder

1958 Flight Memory Unit

David Warren
Principal Research Scientist, Aeronautical Research Laboratories, Melbourne

Image Credit: Bill Schofield

Image Credit: BBC
“Insatiable Death Awaiting Inevitable Accident” by John Tenniel, published in *Punch* in 1891.

Clearly, Nineteenth century iron railroad bridges were developing cracks and failing at unacceptable rates.

From: *To Engineer is Human* by Henry Petroski, 1982.
“Fifty to ninety percent of all structural failures are believed to be the result of crack growth.”

“Often structural failures develop slowly enough to be noticed...and corrective measures can be taken.”

“We can [prevent] material failures, ...but not failures of the mind.”
I35 Bridge, Minneapolis
August 1, 2007
What is the Equivalent of Metal Fatigue in Software?
Operator Fatigue

NUCLEAR ACCIDENT AT THREE MILE ISLAND

On March 28, 1979, and for several days thereafter -- as a result of technical malfunctions and human error -- Three Mile Island’s Unit 2 Nuclear Generating Station was the scene of the nation’s worst commercial nuclear accident. Radiation was released, a part of the nuclear core was damaged, and thousands of residents evacuated the area. Events here would cause basic changes throughout the world’s nuclear power industry.
How Incidents Turn Into Accidents

“Occasionally two or more failures come together in unexpected ways and defeat the safety devices.

If the system is also tightly coupled, these failures can cascade faster than any safety device or operator can cope with them.

If the accident brings down a significant part of the system and the system has catastrophic potential, we will have a catastrophe.”
1. A false ‘leak’ signal shut down the secondary cooling system (which draws heat out of the primary cooling system).

2. Emergency feedwater pumps engaged to cool the primary system, but valves in the pipes – which are always open – had been accidentally left closed. It took 8 minutes to discover this.

3. The uncooled core heated up and the steam generator boiled dry, so the reactor shut down and control rods dropped into the core to stop the chain reaction. But with no cooling, pressure built up.

4. A relief valve opened to relieve pressure – and remained stuck open – causing primary coolant to gush out for over 2 hours. The ‘valve open’ indicator failed, so operators thought it was closed.

5. High pressure injection of cold water started, but operators soon cut it back. Why? They had no indicator of water level in the core, and they guessed (incorrectly) that it was getting dangerously high.

Was the Cause: 1) Operator Error or 2) An Incomprehensible System?
Understanding Controllability & Observability*

Combine \( r \) and \( y \) to get actuator commands

Impart control forces and energy

Measure states and produce the output

You need to be able to influence the system

Controllable

And know it's changing

Observable

*MATLAB Tech Talks by Brian Douglas

Controllable
There exist control signals which allow the system to reach any state in a finite amount of time.

Observable
All states of the system can be known from system outputs.

The Three Mile Island Reactor was barely Controllable, and certainly not Observable.

It’s 40 Years Later – What has Changed?
Complex Interactions, Tight Coupling, Catastrophic Potential, Incomprehensibility

March 10, 2019 – Crashed after takeoff. MCAS software implicated.

Image Credit: LLBG Spotter - Ethiopian Airlines ET-AVJ takeoff from TLV, CC BY-SA 2.0
Six Decades of Software Engineering

Complexity Observability

Increasing complexity drives our systems just beyond our ability to understand them.
1967 – My First Job: Telephone Network Automation

Bell Telephone Laboratories: No. 2 ESS
Electronic Switching System for Community Exchanges

Design goal:
Cost-to-Install and Reliability equivalent to existing electromechanical systems

=> Reliability goal:
Maximum 2 hours downtime in 40 years

3 Minutes / Year!
No 2 ESS
Switching System

**Fault Tolerance: Duplication**

1: Detect Mismatch.
2: Determine and isolate faulty unit.
3: Run diagnostics to isolate fault to a small number of components.
4: Print out repair notification.
5: Resume normal operation upon repair.

Fig. 1 — No. 2 ESS Block Diagram,
On January 15, 1990, a routine failover triggered a cascading outage that shut down half of the US phone network for 9 hours.

We’ve seen this many times: Failures Cascade Across a Network
1970’s: ARPANET Complexity Grows

1971: ARPANET

Each distinct network stands on its own, no internal changes required.
No information retained by the gateways and routers.
No global control at the operations level.
Communications on a best-effort basis.

Copyright©2020 Poppendieck.LLC

1974: TCP/IP

Vinton Cerf and Robert Kahn

Make hosts responsible for data exchange

Fault Handling: Host Responsibility

Copyright©2020 Poppendieck.LLC
1970’s

PARC Inventions

Interactive Computing:
Personal Computers
Desktop Publishing
Laser Printing
Smalltalk
Ethernet
Mouse

Graphical User Interface:
Bitmapped displays
WYSIWYG editing
Overlapping Windows
Menus and Icons

[ XEROX commercialized only ONE of these.]

Photograph Courtesy of Xerox PARC
1970’s & 80’s – My Job: Engineering Systems

Process Control System circa 1980

- Operator Station
- Recipe Storage
- Historical Records

Mini-Computer

PLC (Digital Control)

Switches, Timers

PID Controller

Events

Setpoints

Speed, Flow, Thickness

Motors, Pumps, Heaters, Coolers

Fault Handling: Local Controllers

Observability: Operator Station Historical Records

Event-Driven Object-Oriented Edge Computing

Copyright © 2020 Poppendieck LLC
1990’s – Relentlessly Increasing Complexity

Fault Tolerant

Defect Free

The Internet vs. Heavy Process
Fault Tolerance: Replication Isolation

Fault Handling: Local Responsibility
2010's

Fault Tolerant  
Fault Handling: Local Responsibility  
Observability: Operator  
Observability: Historical Data

Automated Deployment, Loose Coupling, Observability, Machine Learning.
Questions for Software Engineers in 2020

- How do we know our automated deployment is safe and secure?
- How do we mitigate the complexities introduced by microservices?
- Does observability make our systems more reliable, resilient, and safe?
- Do the learning sets used for AI introduce bias into our systems?
- Does our software control systems with catastrophic potential?
- Do we ensure the safety of the systems controlled by our software?
- Are suitable feedback loops and forensic data used for learning?
- Do we accept responsibility for the consequences of our systems?

Have we learned from the colossal failure of the MCAS System?
THANK You!

Mary Poppendieck
mary@Poppendieck.com
www.Poppendieck.com

Poulnabrone dolmen
County Clare, Ireland

Photograph © Tom Poppendieck

Copyright ©2020 Poppendieck LLC