LET'S BRING SYSTEM DYNAMICS BACK TO CS!

YOLO!

ABOUT ME

MARIANNE BELLOTTI - @BELLMAR

- 15+ years as a software engineer
- Lover of complex systems:
  - Legacy Modernization
  - Hybrid intelligence in defense
- 3.5 years doing rescue work on critical systems

WHAT IS SYSTEM DYNAMICS?

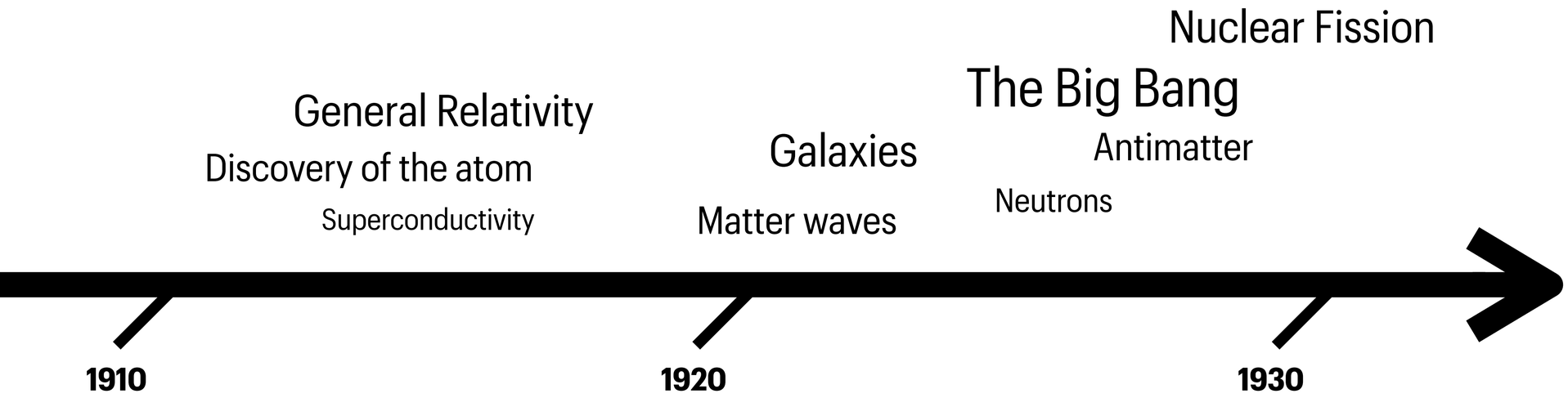
THE LOST MATHEMATICS OF PEOPLE

- Form of modeling systems
- Expressing complex systems in feedback loops
- Versus other approaches that focus on logic based rules
- Popular in policy circles, but originated at MIT with early computer scientists

THE MATHEMATICS OF PEOPLE

CYBERNETICS

- **Emerged at a time when physics was making huge progress defining the natural world**



WHAT IS SYSTEM DYNAMICS?

A Software Engineer's Guide to Cybernetics

Marianne Bellotti
Sep 13, 2020 · 7 min read ★

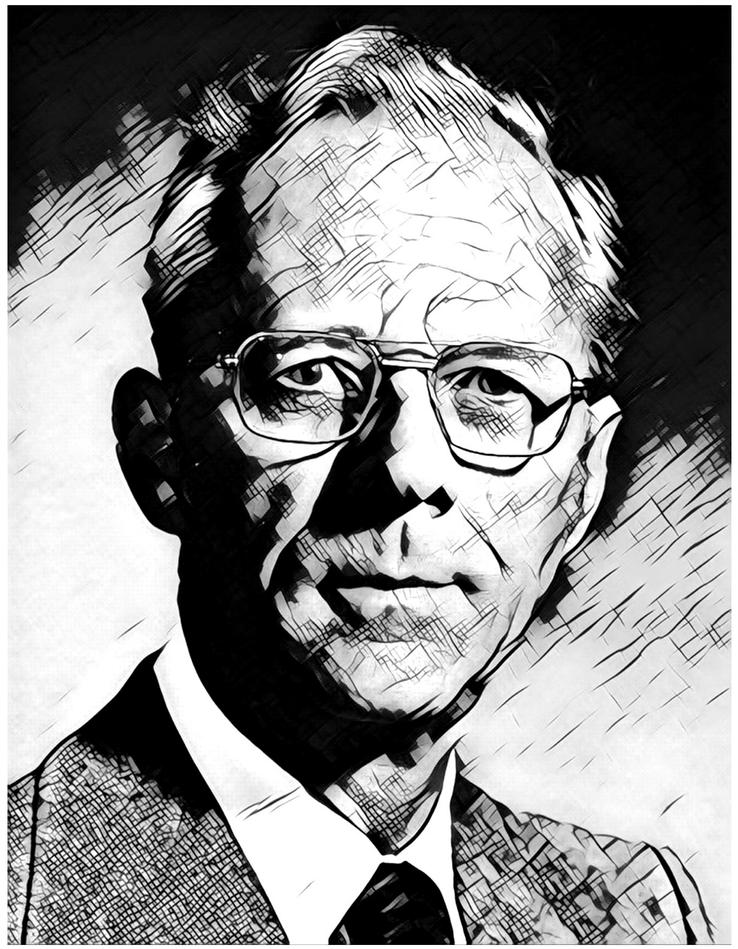
Before 'cyber' was a prefix for everything internet and computers, it was how mathematicians were going to conquer the world.

- Computer scientists connected to cybernetics
  - John von Neumann: computer architecture
  - Warren McCulloch and Walter Pitt: neural networks
  - Claude Shannon: Boolean algebra on circuits
  - J.C.R. Licklider: ARPAnet
  - Alan Turing*: models for cell growth ->Turing machines

WHAT IS SYSTEM DYNAMICS?

FATHER OF SYSTEMS DYNAMICS, JAY WRIGHT FORRESTER

- Magnetic-core memory
- Developed forerunner of RAM
- CS Legend:
  - IEEE Medal of Honor (1972)
  - Howard N. Potts Medal
  - National Medal of Technology and Innovation (1989)
  - Computer History Museum Fellow

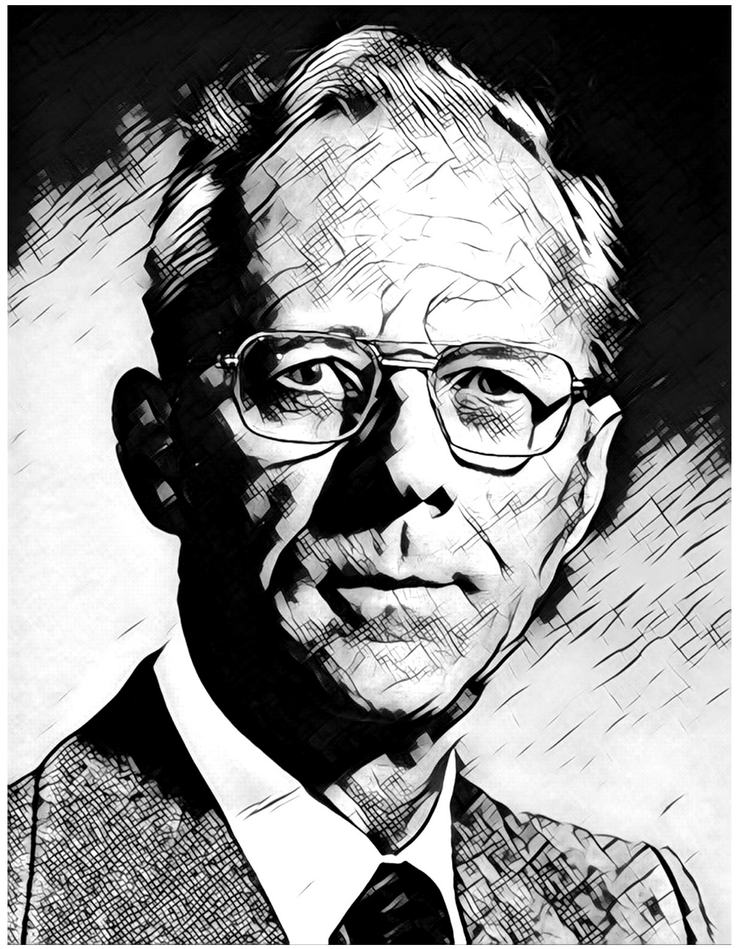


LET'S BRING SYSTEM DYNAMICS BACK TO CS!

WHAT IS SYSTEM DYNAMICS?

THE LOST MATHEMATICS OF PEOPLE

- Involved in the cybernetics movement at MIT
- Moved to MIT Sloan in 1956
- Coined the term “system dynamics”
- Control theory for economics



LET’S BRING SYSTEM DYNAMICS BACK TO CS!

WHAT IS SYSTEM DYNAMICS?

THE LOST MATHEMATICS OF PEOPLE

- Involved in the cybern
  MIT
- Moved to MIT Sloan in
- Coined the term “system dynamics”
- Control theory for economics

LET’S BRING SYSTEM DYNAMICS BACK TO CS!

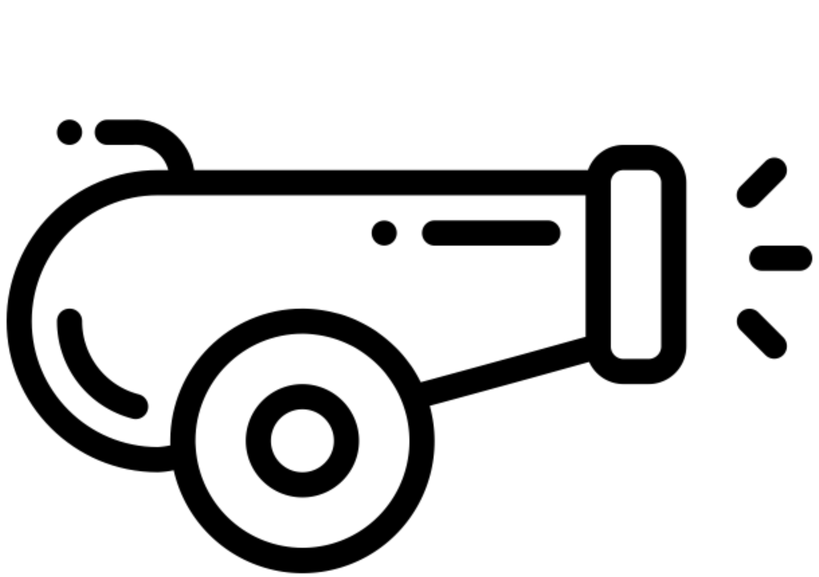
CONTROL THEORY

FEEDBACK LOOPS EVERYWHERE!

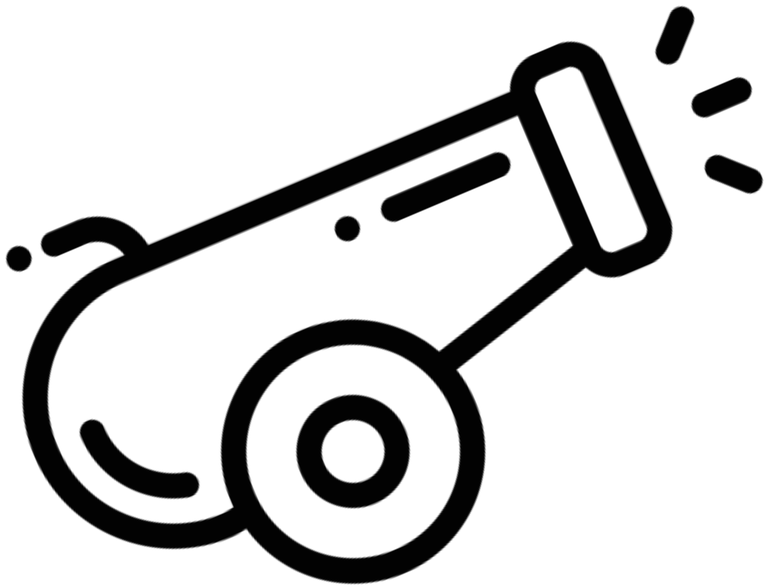
We always seem to have either too much or too little inventory! Build a computer to fix this please!

CONTROL THEORY

FEEDBACK LOOPS EVERYWHERE!



Missed Shot



Overcorrection

LET'S BRING SYSTEM DYNAMICS BACK TO CS!

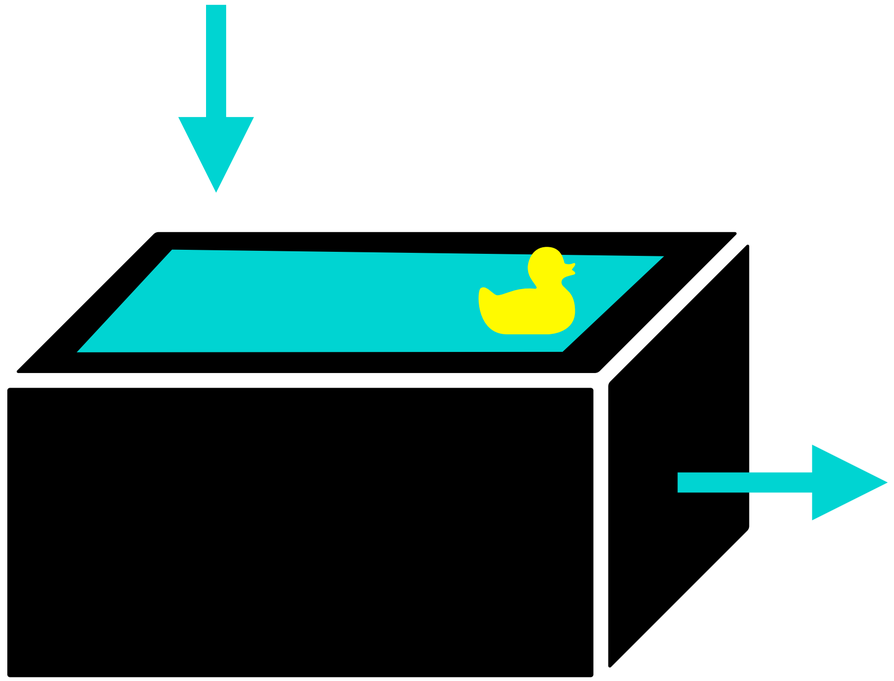
CONTROL THEORY

- Environmental forces respond to changes and need to be factored in:
- Counteracting: limit, restrict the change
- Intensifying: amplify the change

WHAT IS SYSTEM DYNAMICS?

STOCKS AND FLOWS

- Two primary abstractions
  - Stocks: pools of resources
  - Flows: rates of change on those resources



MODELING IN SOFTWARE

PROOFS & OTHER FORMAL METHODS

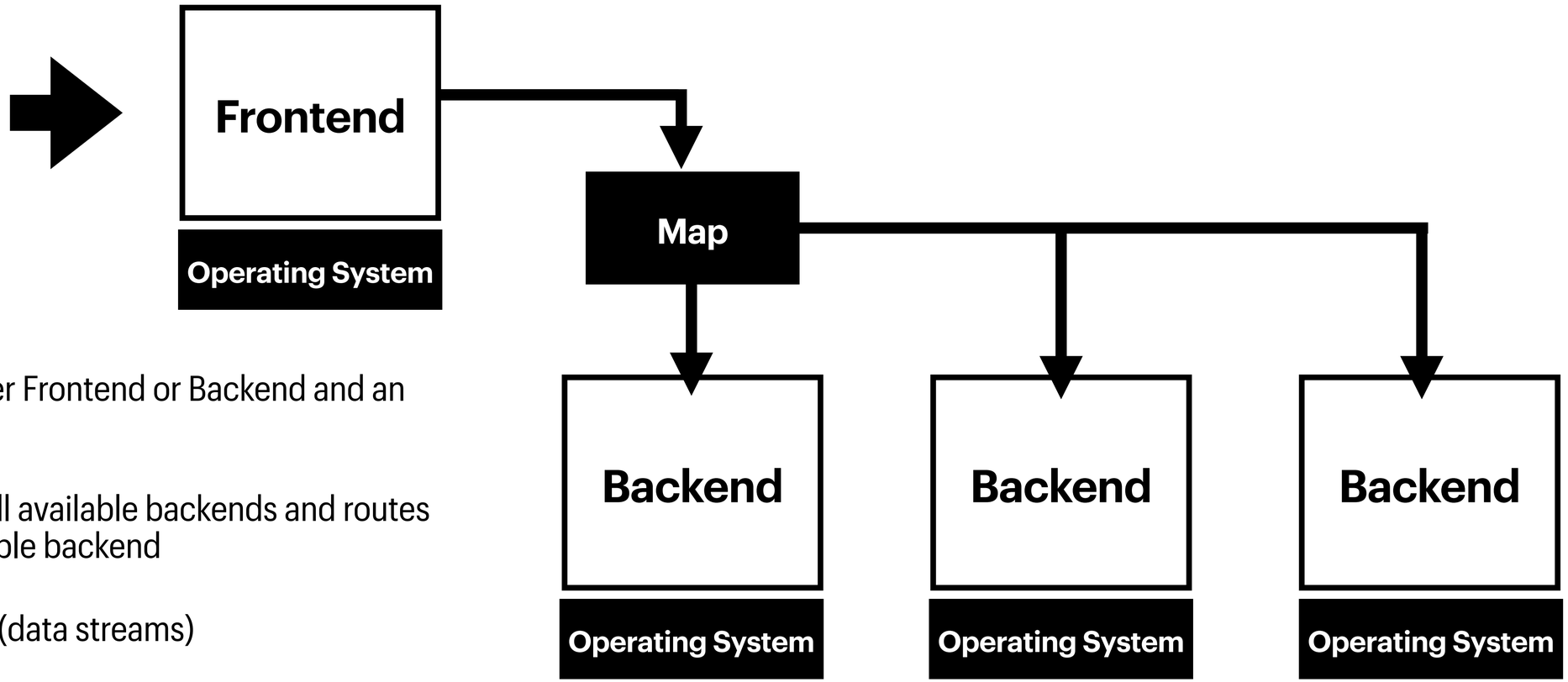
- Despite realizing huge benefits from system dynamics type modeling and research, this is not how computer scientists think about systems today.
- Purely anecdotal, i.e. threat modeling
- Formal Methods:
  - Mathematical correctness
  - Coq (proofs), TLA+ (concurrency), Alloy (relational)

DISTRIBUTED SYSTEMS

PROBLEMS IN THE CLOUD

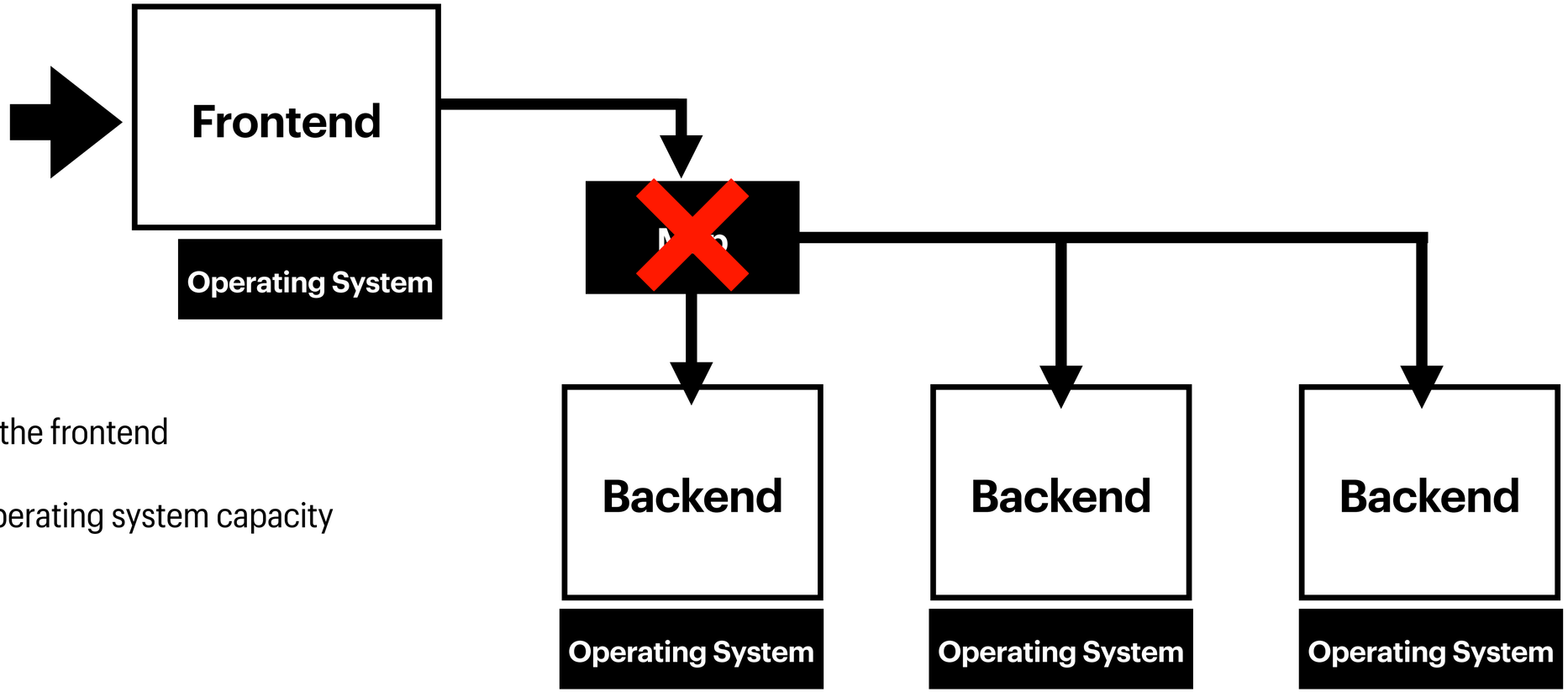
- Mathematical correctness focuses on impossible states
- But distributed systems tend to fail due to *undesirable* states.
  - Feedback loops create a context that triggers unexpected behavior

AWS KINESIS (2019)



- Servers have applications either Frontend or Backend and an operating system
- Frontend generates a map of all available backends and routes incoming requests to an available backend
- Backends process the request (data streams)

AWS KINESIS (2019)



- AWS increases the memory of the frontend
- Frontend now overloads the operating system capacity
- Frontend crash corrupts map

LET'S BRING SYSTEM DYNAMICS BACK TO CS!

AWS KINESIS

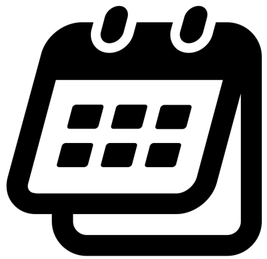
COULD WE HAVE FOUND THIS PROBLEM WITH FORMAL VERIFICATION?

- In theory: Yes, we can write a proof that shows the frontend can exhaust the capacity of the underlining operating system.
- In practice: why would we consider the operating system in scope?
- The real problem is feedback loops between operating system state, frontend state and backend state

OTHER EXAMPLES

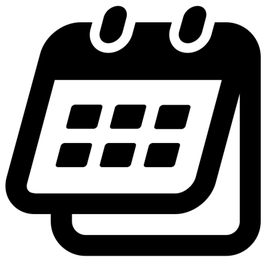
SOME BAD QUARTERS IN 2019

MARCH



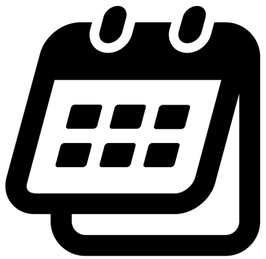
- Facebook
- Instagram

APRIL



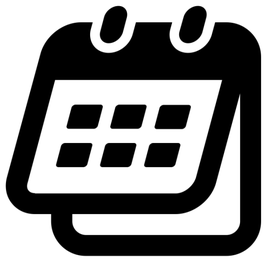
- AeroData

MAY



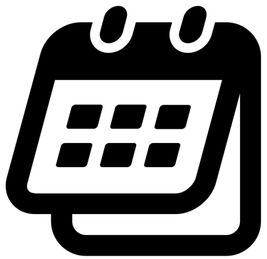
- Microsoft
- Salesforce

JUNE



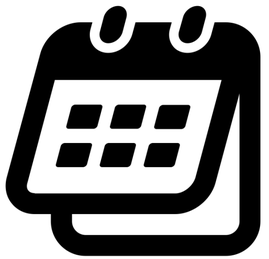
- Google
- Verizon
- Slack

JULY



- Cloudflare
- Facebook
- Twitter
- Apple
- Twitter again!

AUGUST



- Comcast

SYSTEMS OF SYSTEMS

NOT “HUMAN ERROR”

- The largest cause of these types of failures is “configuration change”
- But these changes are stable within their subsystem
- Modern computer systems chain several systems together, creating feedback loops
- Tight coupling makes it difficult to stop failure from cascading through the system

FORMAL METHODS

PROBLEMS IN THE CLOUD

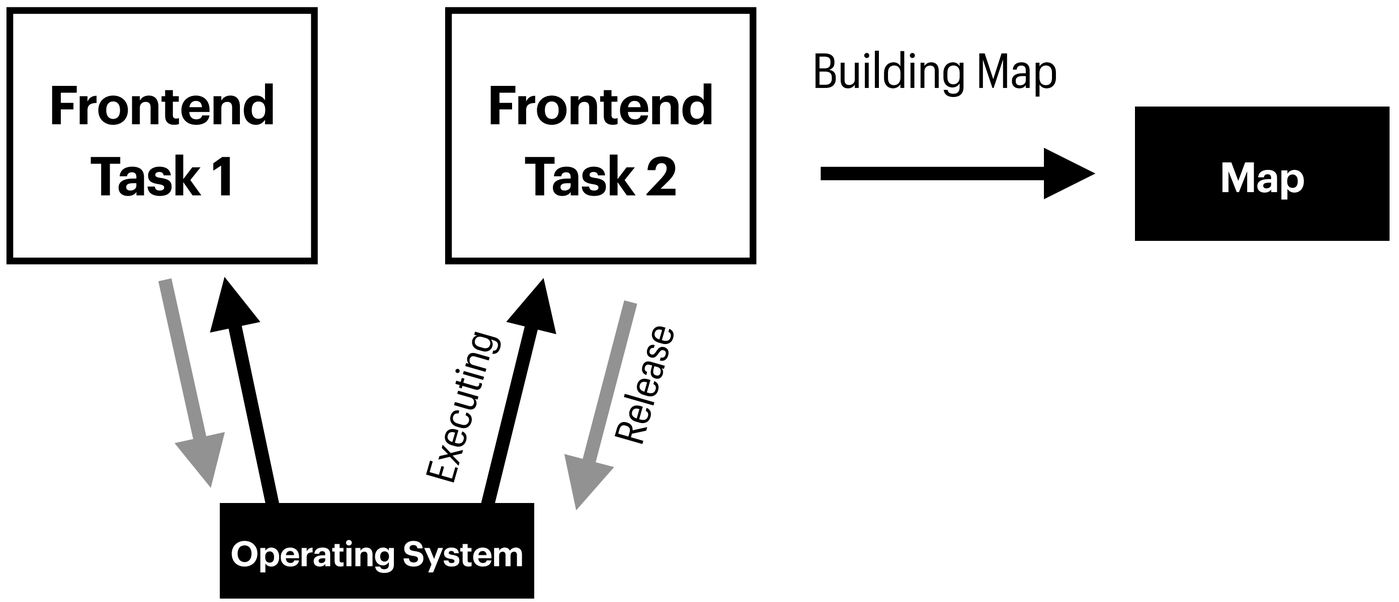
- Mathematical correctness focuses on impossible states
- But distributed systems tend to fail due to *undesirable* states.
  - Feedback loops create a context that triggers unexpected behavior
  - Ratio of complexity to tight coupling

BENEFITS

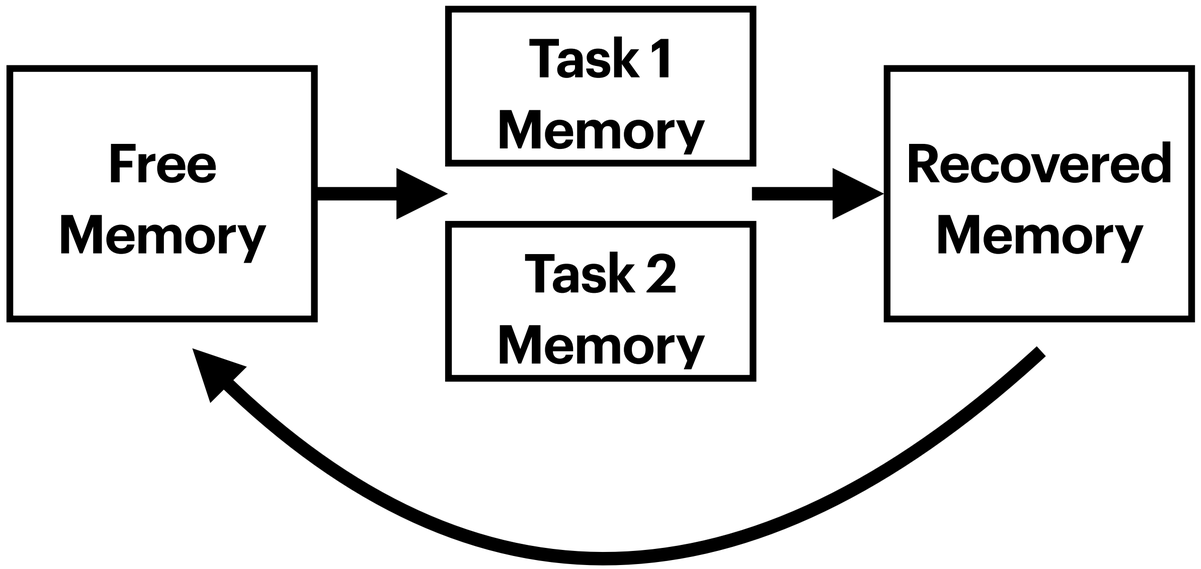
SPECS FOR DISTRIBUTIVE SYSTEMS

- Stock/Flow models are good at expressing coupling
    - Changes within a single flow —> tightly coupled
    - Flows separated by a stock —> loosely coupled
- Like threat modeling, system dynamics can help us reason about risks

FLOW OF RESOURCES



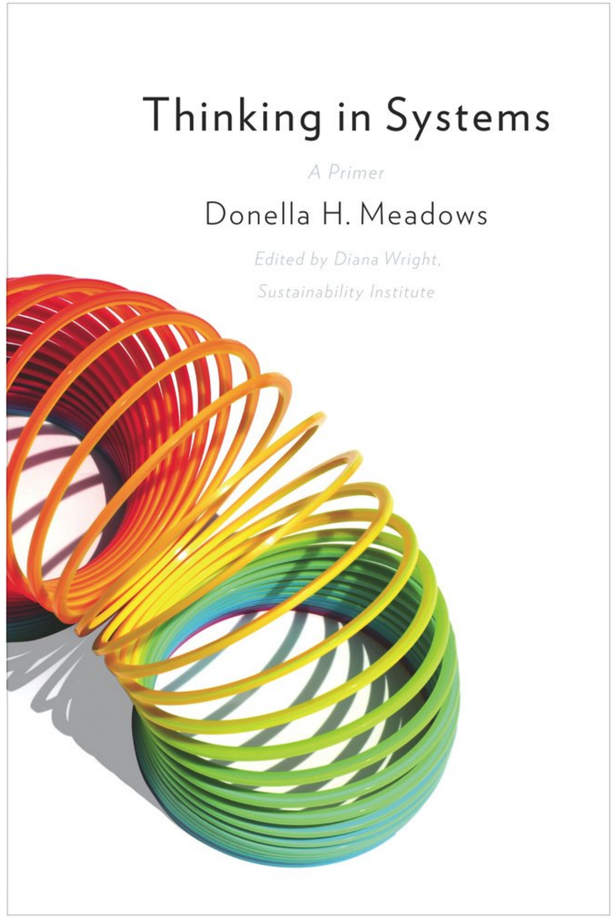
FLOW OF RESOURCES



LEARN

APPLYING SYSTEM DYNAMIC MODELS TO COMPUTER SYSTEMS

- *Thinking in Systems*, Donella Meadows
- System Dynamics Society (systemdynamics.org)
- Loopy (ncase.me/loopy) simulations of simple models



TOOLS

APPLYING SYSTEM DYNAMIC MODELS TO COMPUTER SYSTEMS

- Insight Maker (insightmaker.com)
- Simlin (simlin.com)
- PySD (pysd.readthedocs.io)
- PowerSim, Vensim, iThink and STELLA

FAULT

APPLYING SYSTEM DYNAMIC MODELS TO COMPUTER SYSTEMS

- Model checking for system dynamic models
- SMT solver under the hood
- Uses types to eliminate common model mistakes
- Doing a podcast about my experiences—>

LET'S BRING SYSTEM DYNAMICS BACK TO CS!

Let’s demo!

Keep in touch @bellmar

THANK YOU!

HTTPS://WWW.GITHUB.COM/FAULT-LANG