iCyPhy

Deterministic Modeling of Uncertain Systems: Implications for Certification

Edward A. Lee

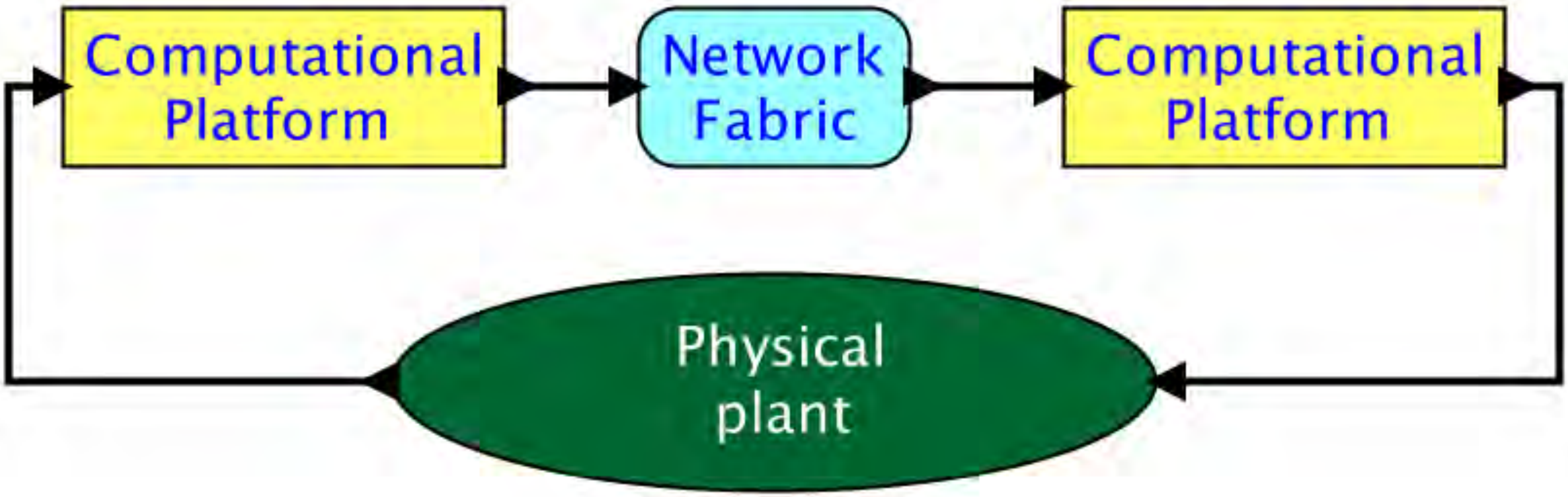
Professor in the Graduate School and
Robert S. Pepper Distinguished Professor Emeritus

Software Certification Consortium (SCC) Meeting

May 2-3, 2019, Annapolis, MD, USA

University of California at Berkeley

Determinism? Really?



Should we insist on determinism for complex systems, where Murphy's Law and physical reality will thwart us at every turn?

Determinism in Physics: Laplace’s Demon

Pierre Simon Laplace

Pierre-Simon Laplace (1749–1827).
Portrait by Joan-Baptiste Paulin Guérin, 1838

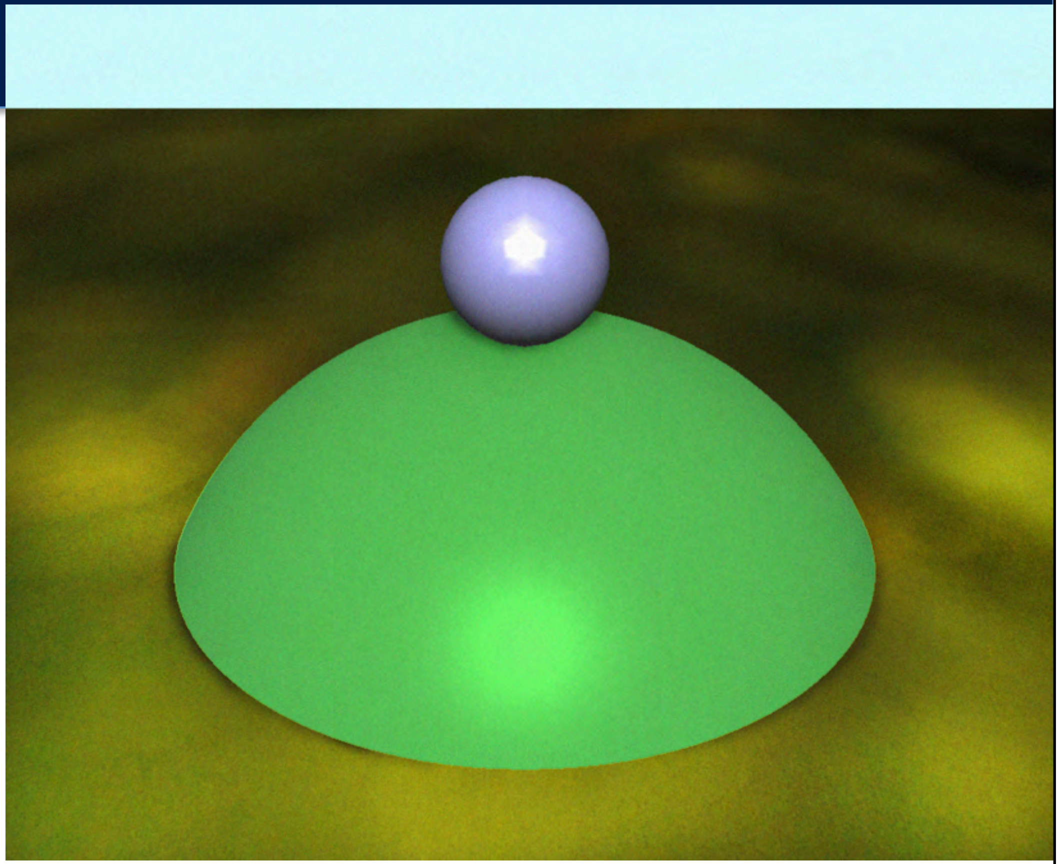
Did quantum physics dash this hope?

"At first, it seemed that these hopes for a complete determinism would be dashed by the discovery early in the 20th century that events like the decay of radioactive atoms seemed to take place at random. It was as if God was playing dice, in Einstein's phrase. But science snatched victory from the jaws of defeat by moving the goal posts and redefining what is meant by a complete knowledge of the universe."

(Stephen Hawking, 2002)

Norton's Hill

Even without quantum physics, Newtonian physics is not deterministic.



Metastable system that obeys all of Newton's laws but is nondeterministic.

Norton, J. D. (2007). Causation as Folk Science.
In Causation, Physics, and the Constitution of Reality
Oxford, Clarendon Press:

Laplace’s Demon cannot exist.

In 2008, David Wolpert proved that Laplace’s demon cannot exist.

His proof relies on the observation that such a demon, were it to exist, would have to exist in the very physical world that it predicts.

David Wolpert

Determinism is a property of models, not things

$$x(t) = x(0) + \int_0^t v(\tau)d\tau$$

$$v(t) = v(0) + \frac{1}{m}\int_0^t F(\tau)d\tau,$$

Deterministic model

Deterministic system?

Recall from Yesterday: Two Uses of Models

- In *science*, the value of a model lies in how well its behavior matches that of the physical system.

- In *engineering*, the value of a *physical system* lies in how well its behavior matches that of the model.

A scientist asks, “Can I make a model for this thing?”
An engineer asks, “Can I make a thing for this model?”

Models vs. Reality

$$x(t) = x(0) + \int_0^t v(\tau) d\tau$$

$$v(t) = v(0) + \frac{1}{m} \int_0^t F(\tau) d\tau,$$

The model

The target (the thing being modeled).

In this example, the *modeling framework* is calculus and Newton's laws.

Fidelity is how well the model and its target match

A Model

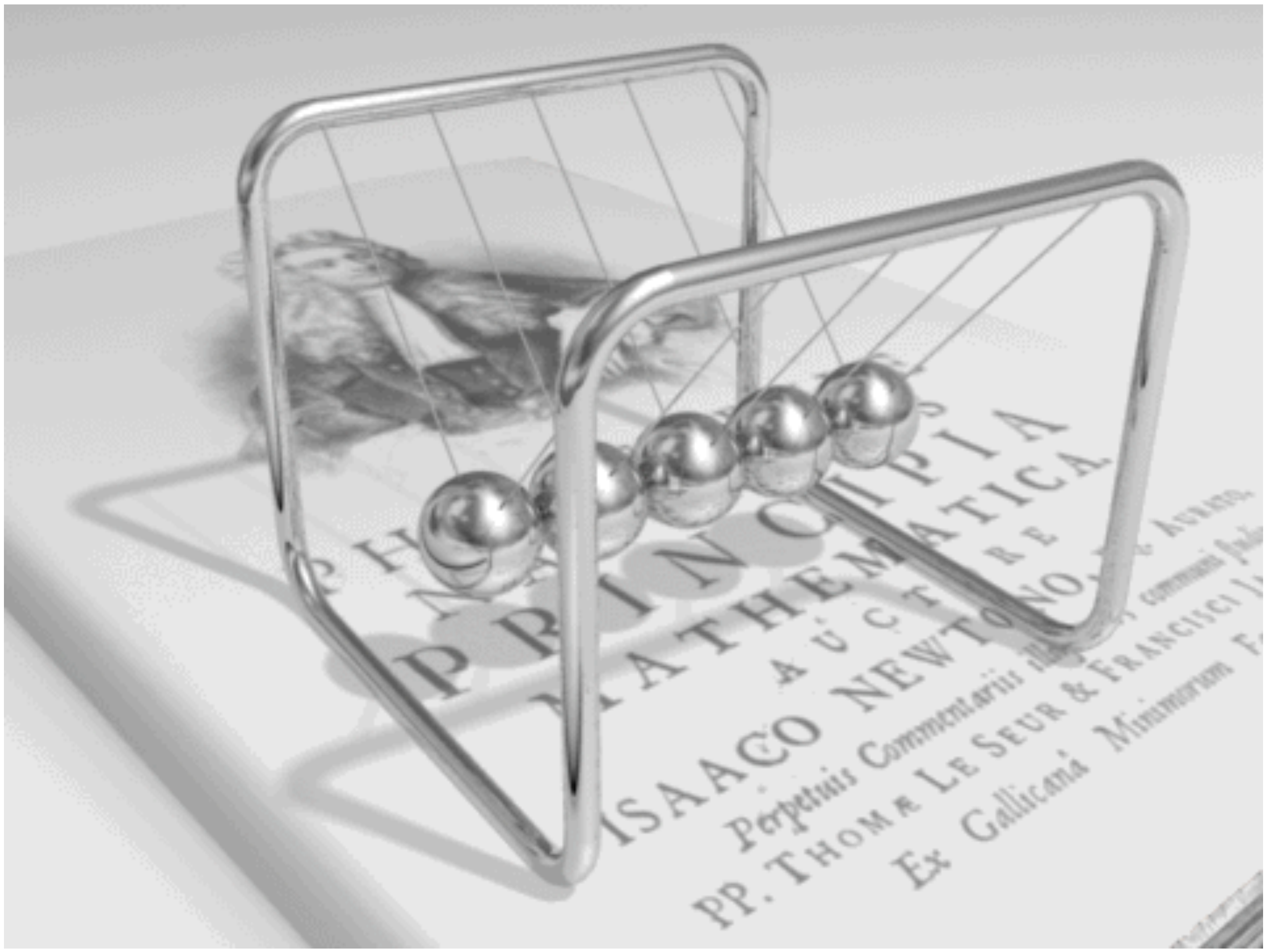
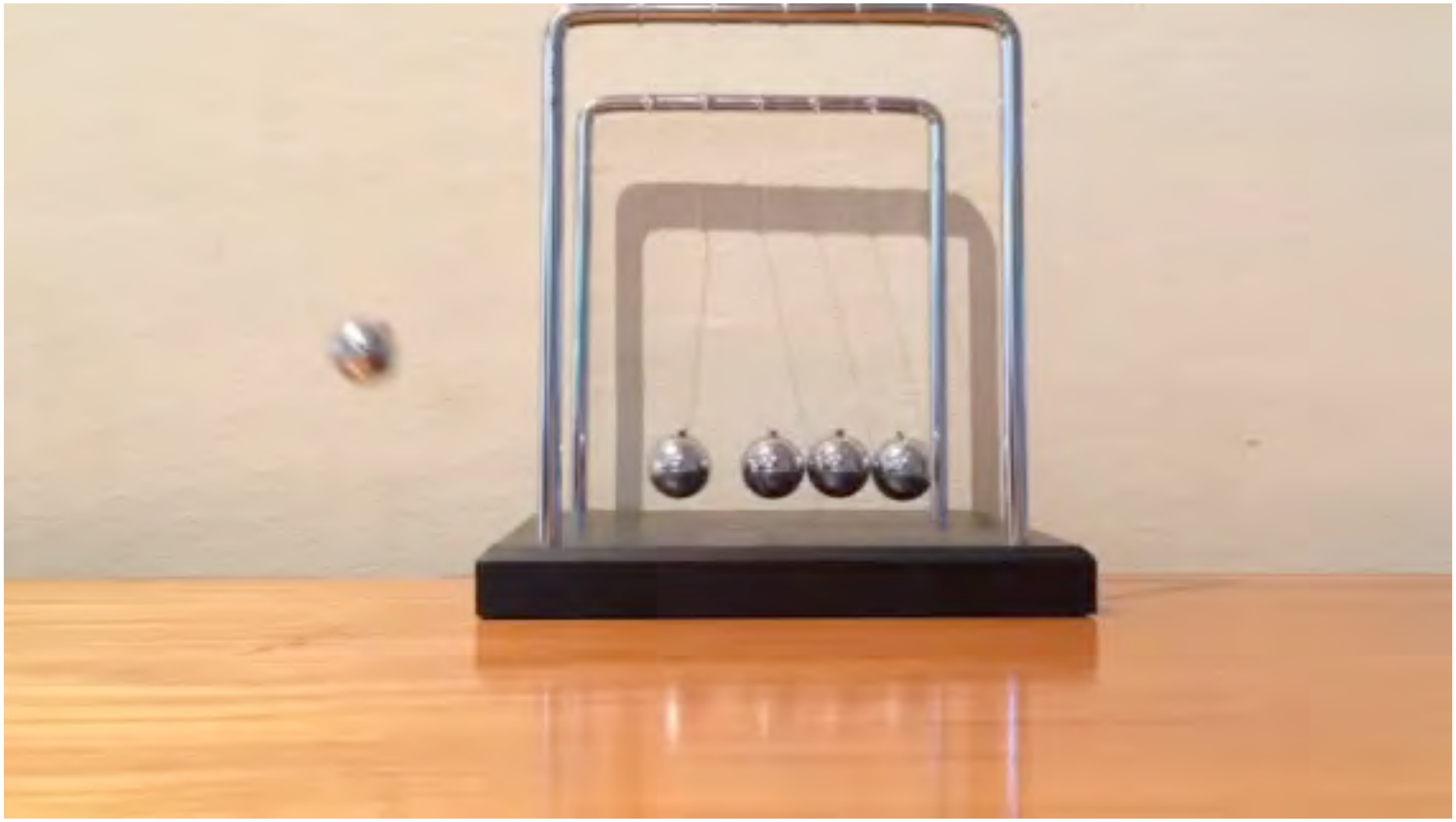


Image by Dominique Toussaint, GNU Free Documentation License, Version 1.2 or later.

A Physical Realization

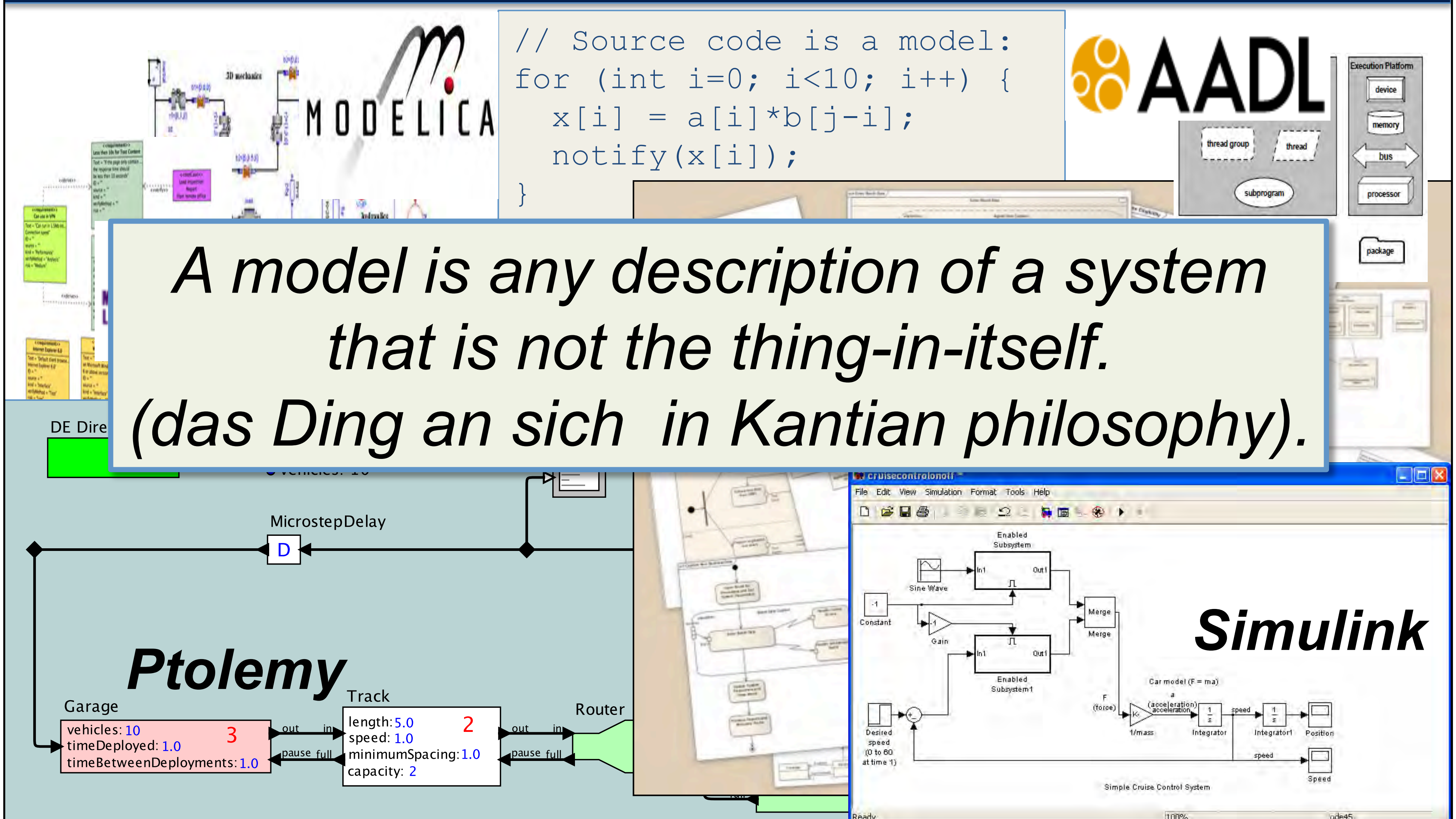


Model Fidelity

- To a *scientist*, the model is flawed.
- To an *engineer*, the realization is flawed.

To a realist, both are flawed…

What is a model?

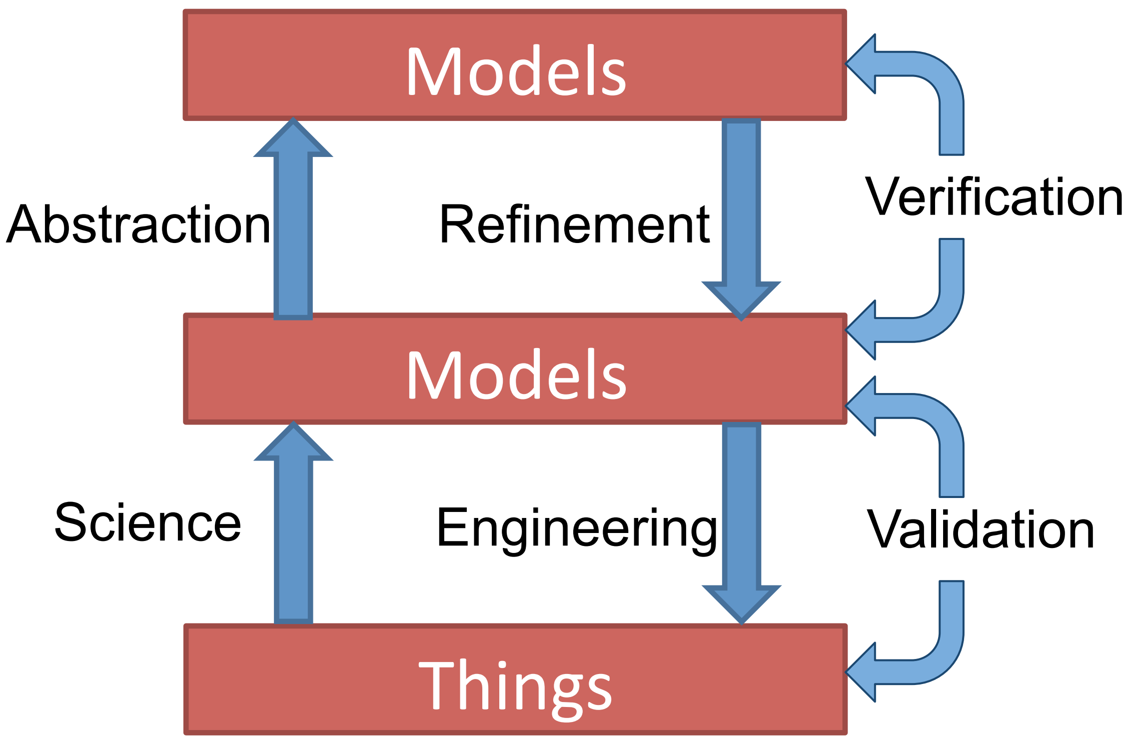


A model is any description of a system that is not the thing-in-itself.
(das Ding an sich in Kantian philosophy).

The Framework

Per Boehm:

- Am I building the right product? (validation)
- Am I building the product right? (verification)



Automation is most useful for the Verification question.

Determinism as a Property of Models

A model is *deterministic* if, given the initial *state* and the *inputs*, the model defines exactly one *behavior*.

Our most valuable models are *deterministic*.

Software as a Model

```
void foo(int32_t x) {
      if (x > 1000) {
            x = 1000;
      }
      if (x > 0) {
            x = x + 1000;
            if (x < 0) {
                  panic();
            }
      }
}
```

This program defines exactly one behavior, given the input x.

The modeling framework defines *state*, *input*, and *behavior*.

The physical system has many properties not represented in the model (e.g. timing).

Architecture as a Model

Physical System

Image: Wikimedia Commons

Model

Integer Register-Register Operations

RISC-V defines several arithmetic R-type operations. All operations read the *rs1* and *rs2* registers as source operands and write the result into register *rd*. The *funct* field selects the type of operation.

31 — 27	26 — 22	21 — 17	16 — 7	6 — 0
rd	rs1	rs2	funct10	opcode
5	5	5	10	7
dest	src1	src2	ADD/SUB/SLT/SLTU	OP
dest	src1	src2	AND/OR/XOR	OP
dest	src1	src2	SLL/SRL/SRA	OP
dest	src1	src2	ADDW/SUBW	OP-32
dest	src1	src2	SLLW/SRLW/SRAW	OP-32

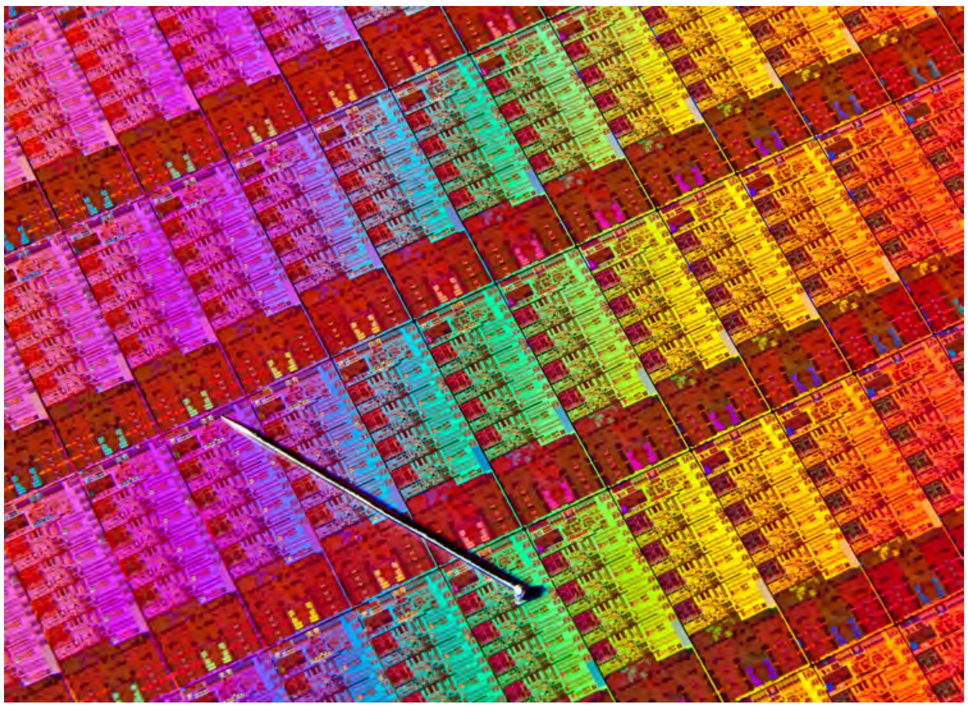
Waterman, et al., The RISC-V Instruction Set Manual, UCB/EECS-2011-62, 2011

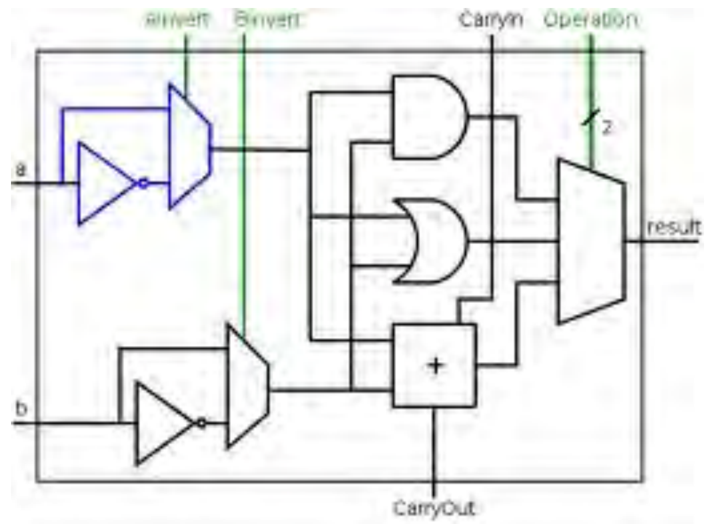
Instruction Set Architectures (ISAs) are deterministic models

Digital Circuits as Models

Physical System

Model





Synchronous digital logic
is a deterministic model

Physical Dynamics as a Model

Physical System

Image: Wikimedia Commons

Model

$$\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \mathbf{F}(\tau)d\tau$$

Differential Equations
are deterministic models

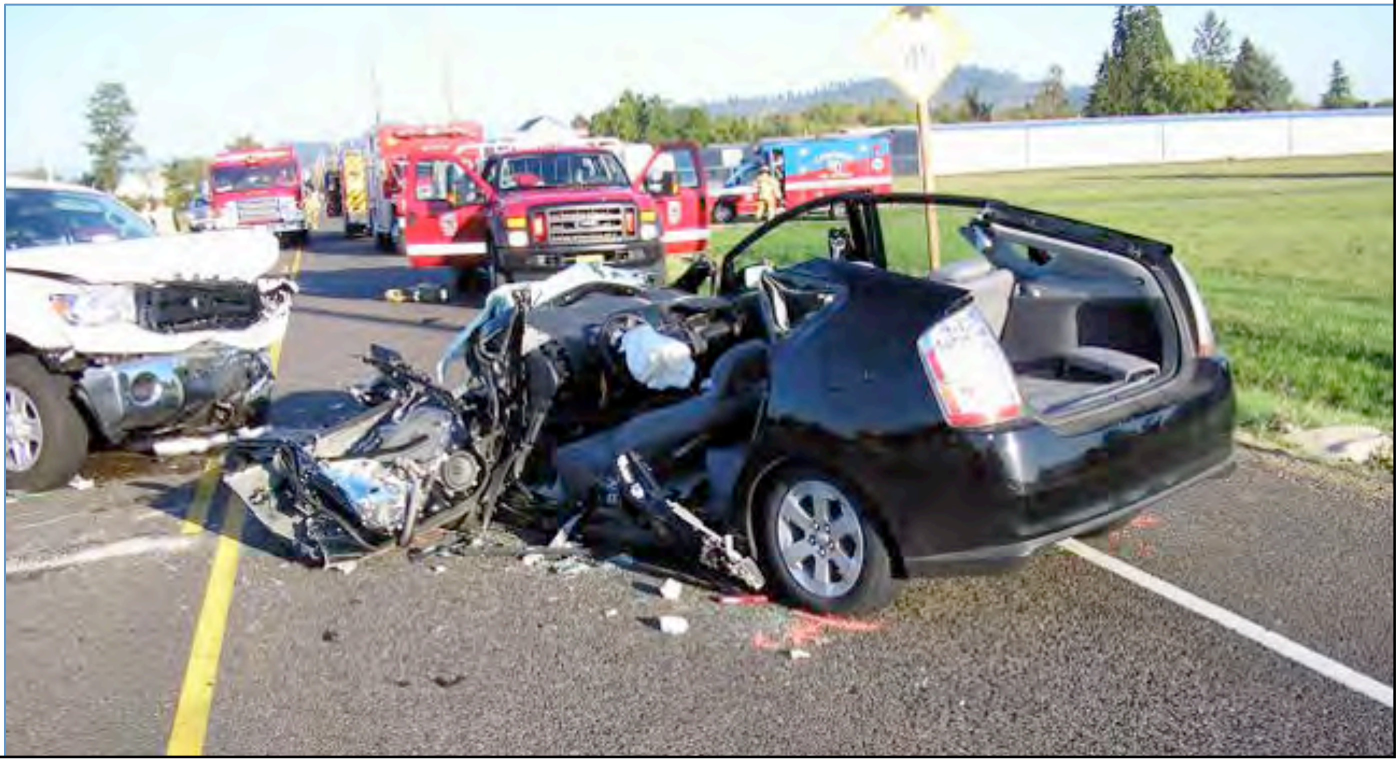
Why have deterministic models proven so valuable?

- They enable testing.
  - Known inputs => known outputs
- Analysis is more tractable.
  - Math: Boolean algebra, calculus, etc.
- Simulation is more useful.
  - One input yields one trace.
- Verification scales better.
  - Much smaller state space.
- More certifiable?

“Toyota” Style of Design

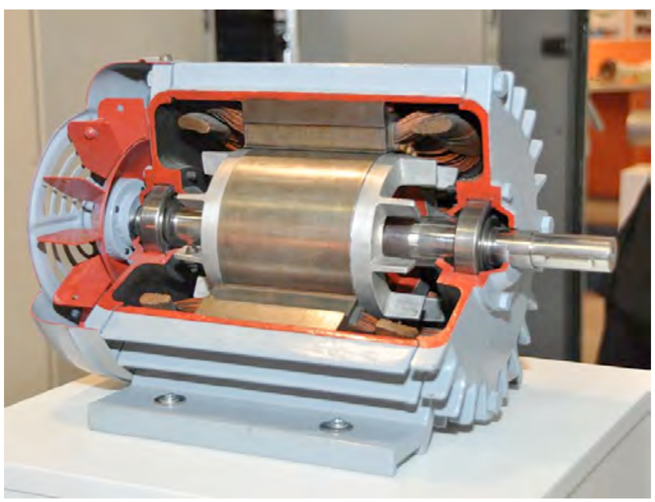
NASA's Toyota Study Released by Dept. of Transportation released in 2011 found that Toyota software was “untestable.”

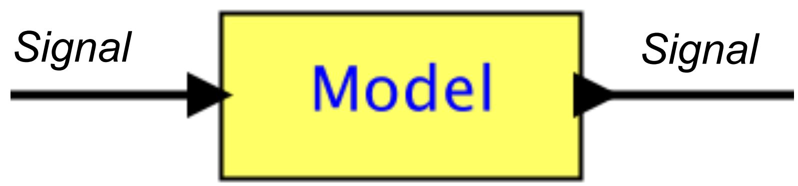
Possible victim of unintended acceleration.



CPS combinations of deterministic models are nondeterministic

```
void initTimer(void) {
    SysTickPeriodSet(SysCtlClockGet() / 1000);
    SysTickEnable();
    SysTickIntEnable();
}
volatile uint timer_count = 0;
void ISR(void) {
    if(timer_count != 0) {
        timer_count--;
    }
}
int main(void) {
    SysTickIntRegister(&ISR);
    .. // other init
    timer_count = 2000;
    initTimer();
    while(timer_count != 0) {
        ... code to run for 2 seconds
    }
    ... // other code
}
```





$$\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \mathbf{F}(\tau)d\tau$$

Image: Wikimedia Commons

Parts Stockpiling

Could this explain why aircraft manufacturers stockpile the microprocessors they expect to need for the entire production and maintenance run of an aircraft?

Determinism?
What about resilience? Adaptability?

Deterministic models do not eliminate the need for robust, fault-tolerant designs.

In fact, they *enable* such designs, because they make it much clearer what it means to have a fault!

Existence proofs that useful deterministic models for CPS exist

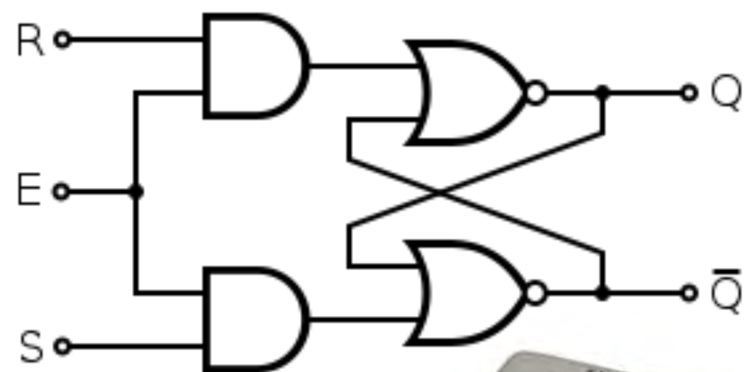
Deterministic models for CPS with faithful implementations exist:

- PTIDES: distributed real-time software
  - http://chess.eecs.berkeley.edu/ptides
- PRET: time-deterministic architectures
  - http://chess.eecs.berkeley.edu/pret

Together, these technologies give a programming model for distributed and concurrent real-time systems that is deterministic and has controlled timing.

The hardware out of which we build computers is capable of delivering “correct” computations *and* precise timing…

Synchronous digital logic delivers precise, repeatable timing.



… but the overlaying software abstractions discard timing.

```
// Perform the convolution.
for (int i=0; i<10; i++) {
  x[i] = a[i]*b[j-i];
  // Notify listeners.
  notify(x[i]);
}
```

PRET Machines – Giving software the capabilities its hardware already has.

- **PRE**cision-**T**imed processors = **PRET**
- **P**redictable, **RE**peatable **T**iming = **PRET**
- **P**erformance *with* **RE**peatable **T**iming = **PRET**

http://chess.eecs.berkeley.edu/pret

```
// Perform the convolution.
for (int i=0; i<10; i++) {
  x[i] = a[i]*b[j-i];
  // Notify listeners.
  notify(x[i]);
}
```

Computing

+

With time

= PRET

Current Work: Lingua Franca: Actors Done Right

A polyglot meta-language for deterministic, concurrent, time-sensitive systems.

Invited: Actors Revisited for Time-Critical Systems

Marten Lohstroh
UC Berkeley, USA

Martin Schoeberl
TU Denmark, Denmark

Andrés Goens
TU Dresden, Germany

Armin Wasicek
Avast, USA

Christopher Gill
Washington Univ., St. Louis, USA

Marjan Sirjani
Mälardalen Univ., Sweden

Edward A. Lee
UC Berkeley, USA

ABSTRACT

Programming time-critical systems is notoriously difficult. In this paper we propose an actor-oriented programming model with a semantic notion of time and a deterministic coordination semantics based on discrete events to exercise precise control over both the computational and timing aspects of the system behavior.

2 ACTORS

The actor model was introduced by Hewitt [6] in the early 70s. Since then, the use of actors has proliferated in programming languages [1, 2], coordination languages [14, 15], distributed systems [7, 11], and simulation and verification engines [13, 17]. Actors have much in common with objects—a paradigm focused on reducing code replication and increasing modularity via data

To Appear, Design Automation Conference (DAC), June, 2019.

But…

Determinism has its limits.

- Complexity
- Uncertainty
- Chaos
- Incompleteness

Complexity

- Some systems are too complex for deterministic models.
- Nondeterministic abstractions become useful.

“Iron wing” model of an Airbus A350.

But…

Determinism has its limits.

- Complexity
- Uncertainty
- Chaos
- Incompleteness

Uncertainty

- We can't construct deterministic models of what we don't know.
- For this, nondeterminism is useful.
- Bayesian probability (which is mostly due to Laplace) quantifies uncertainty.

Portrait of Reverend Thomas Bayes (1701 - 1761) that is probably not actually him.

But…

Determinism has its limits.

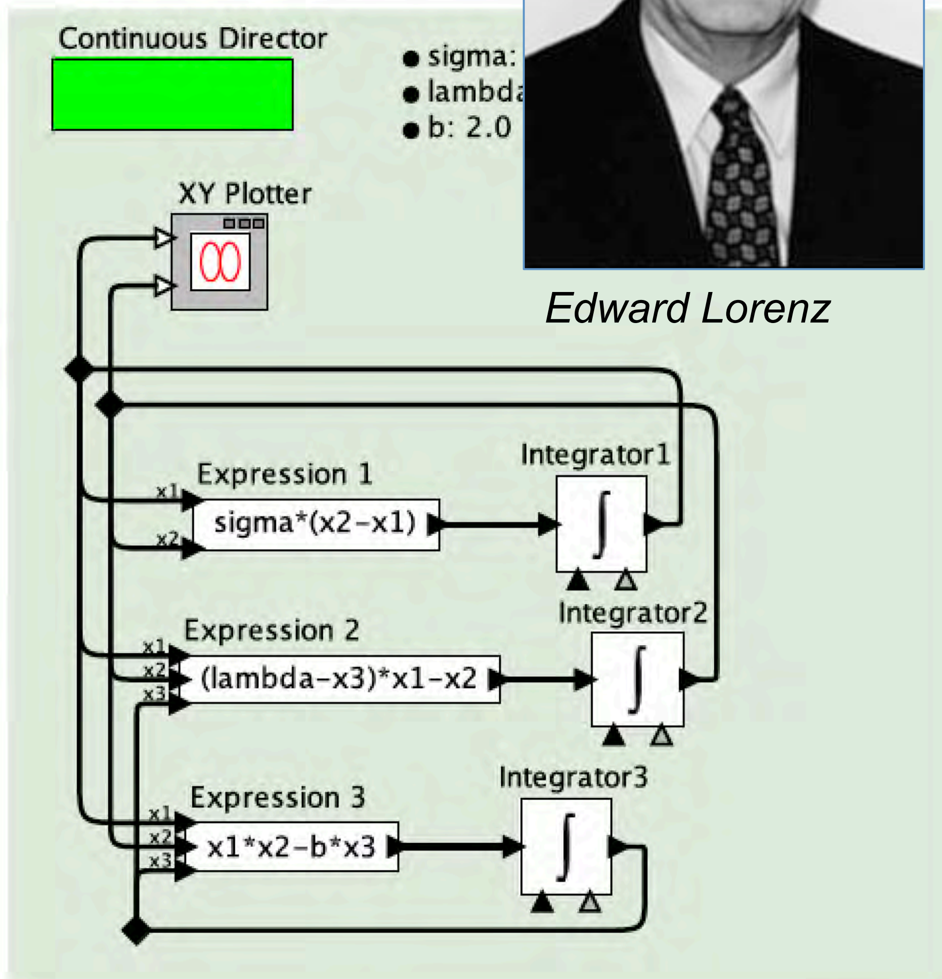
- Complexity
- Uncertainty
- Chaos
- Incompleteness

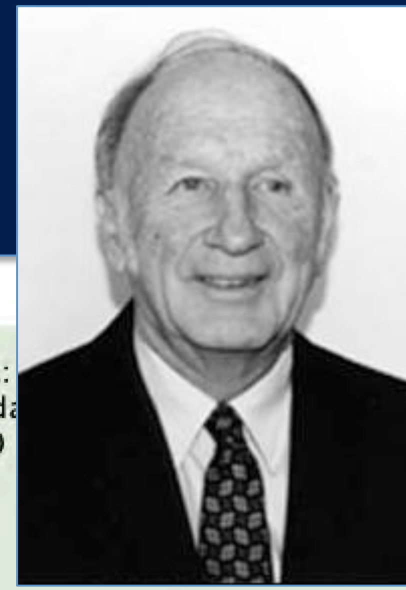
Determinism does not imply predictability

Lorenz attractor:

$$
\begin{aligned}
\dot{x}_1(t) &= \sigma(x_2(t) - x_1(t)) \\
\dot{x}_2(t) &= (\lambda - x_3(t))x_1(t) - x_2(t) \\
\dot{x}_3(t) &= x_1(t)x_2(t) - bx_3(t)
\end{aligned}
$$

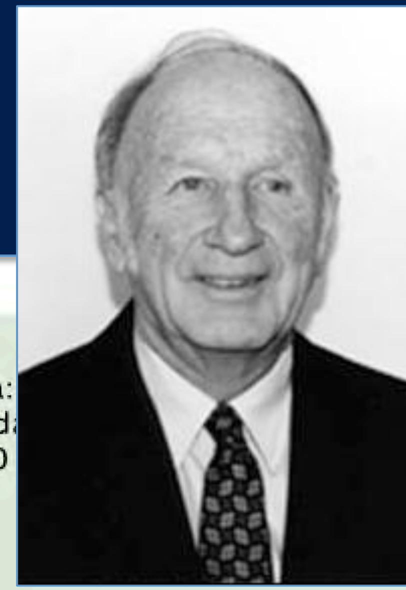
This is a chaotic system, so arbitrarily small perturbations have arbitrarily large consequences.





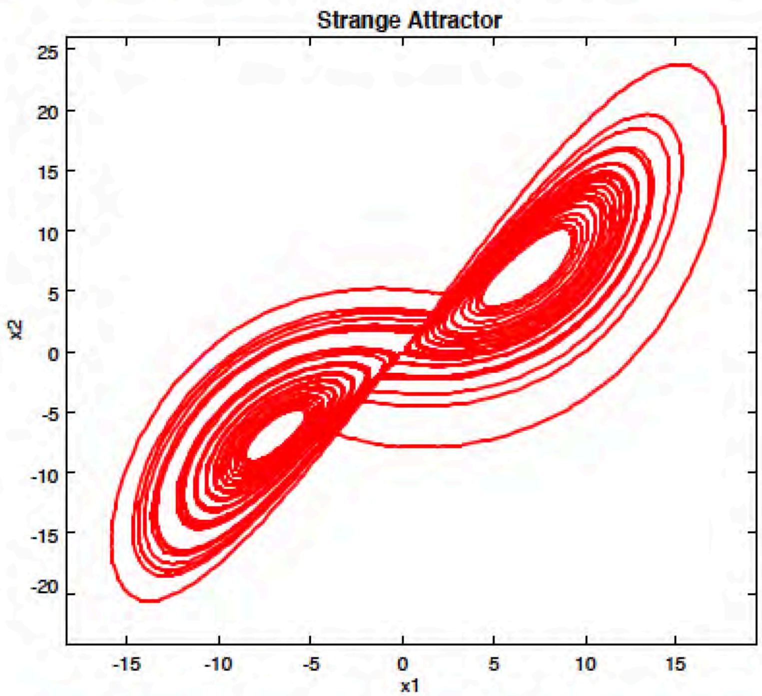
Edward Lorenz

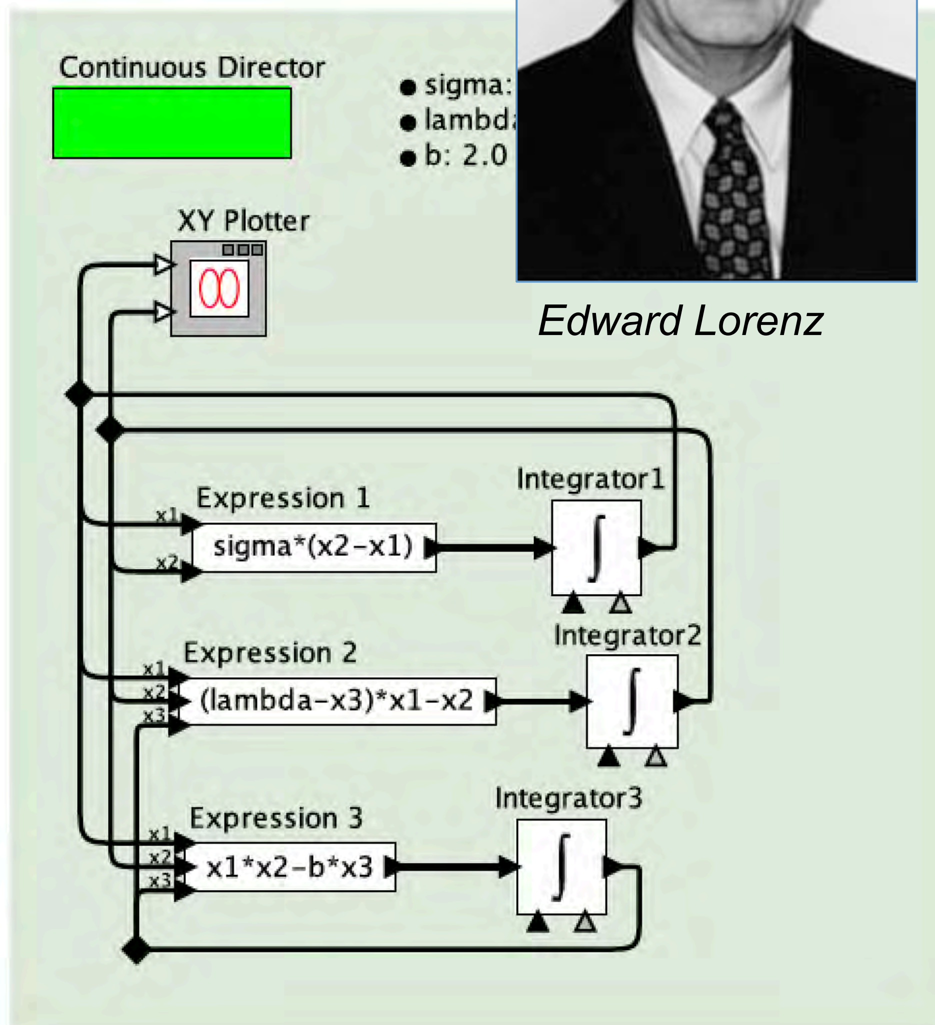
Determinism does not imply predictability



Edward Lorenz

Plot of x_1 vs. x_2:





The position of a point is not meaningfully predictable even though the system is deterministic.

Determinism does not imply predictability

Deterministic real-time scheduling results in chaos.

[Thiele and Kumar, EMSOFT 2015]

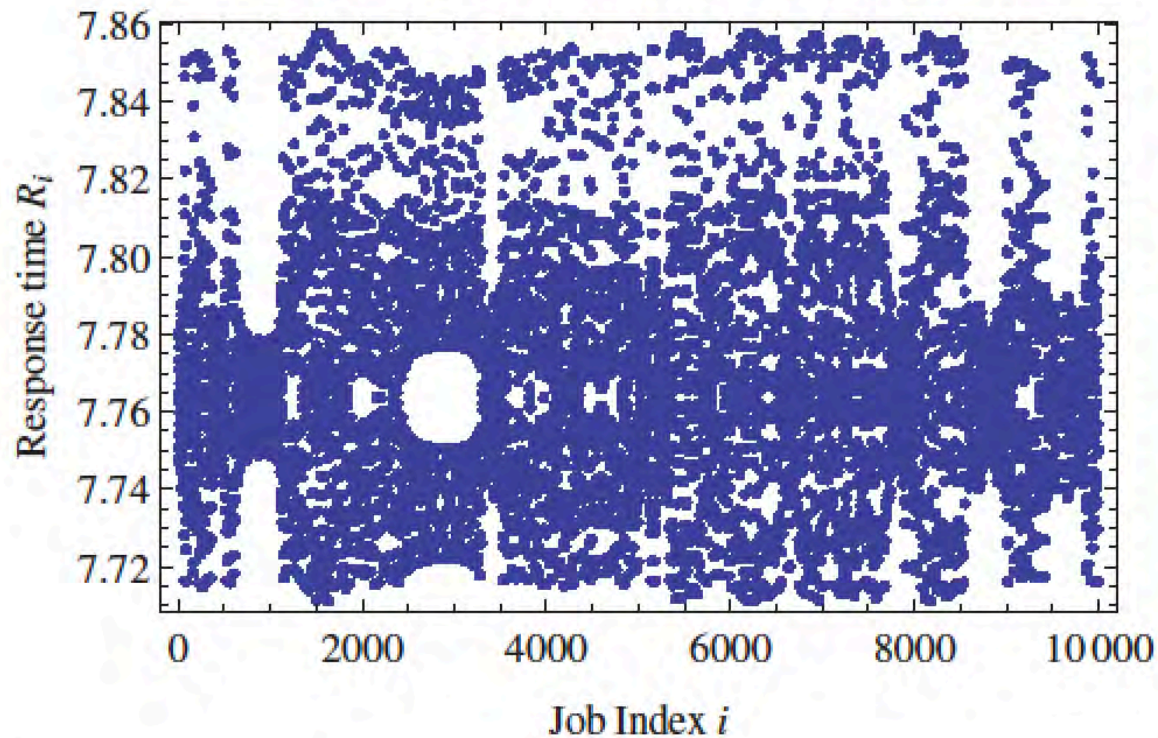


Fig. 15. Response time across jobs for the multi-resource scheduler with $R_s(i-1) = 7.76$ and $R_s(i-2) = 7.74$.

But…

Determinism has its limits.

- Complexity
- Uncertainty
- Chaos
- Incompleteness

Incompleteness of Determinism

Any set of deterministic models rich enough to encompass Newton's laws plus discrete transitions is incomplete.

Lee, *Fundamental Limits of Cyber-Physical Systems Modeling*, ACM Tr. on CPS, Vol. 1, No. 1, November 2016

Fundamental Limits of Cyber-Physical Systems Modeling

EDWARD A. LEE, EECS Department, UC Berkeley

This article examines the role of modeling in the engineering of cyber-physical systems. It argues that the role that models play in engineering is different from the role they play in science, and that this difference should direct us to use a different class of models, where simplicity and clarity of semantics dominate over accuracy and detail. I argue that determinism in models used for engineering is a valuable property and should be preserved whenever possible, regardless of whether the system being modeled is deterministic. I then identify three classes of fundamental limits on modeling, specifically chaotic behavior, the inability of computers to numerically handle a continuum, and the incompleteness of determinism. The last of these has profound consequences.

CCS Concepts: • **Theory of computation** → **Timed and hybrid models**; • **Computing methodologies** → *Modeling methodologies*; • **Software and its engineering** → Domain specific languages

Additional Key Words and Phrases: Chaos, continuums, completeness

ACM Reference Format:
Edward A. Lee. 2016. Fundamental limits of cyber-physical systems modeling. ACM Trans. Cyber-Phys. Syst. 1, 1, Article 3 (November 2016), 26 pages.
DOI: http://dx.doi.org/10.1145/2912149

Summary

- Deterministic models are extremely useful.
- Combining of our best deterministic cyber models and physical models today yields nondeterministic models.
- But deterministic models with faithful implementations exist (in research) for cyber-physical systems.
- Deterministic models aren't always possible or practical due to complexity, unknowns, chaos, and incompleteness.
- Determinism is a powerful modeling tool.
  Use it if you can. Back off only when you can't.

Conclusion

Models play a central role in reasoning about and designing engineered systems.

Determinism is a valuable and subtle property of models.

http://PlatoAndTheNerd.org