

An aerial photograph of a city, likely Zurich, showing a river (Limmat) flowing through the center. On the left, a bridge crosses the river. The right side of the image is dominated by a large, historic building with a prominent dome, which is the main building of ETH Zurich. The surrounding area includes other city buildings, trees, and a yellow construction crane in the lower right.

A case for models in operating systems

Timothy Roscoe (Mothy)
Dept. of Computer Science, ETH Zurich

Previously on ...

ACM SIGOPS Strategic Workshop (SOSP 2025)

⇒ OSCAR Workshop (EuroSys 2025)



Systems Research is averse to systematization

- Papers are preferred that **improve systems** but **avoid systematizing knowledge**.
 - Close to plain old good engineering
 - Companies should do this themselves, rather than firing people.
 - AI has just made this so much more of an issue.
- This is leaving a lot of potential on the table
 - Less reinvention of the wheel and more significant progress
 - Less reliance on specific products or industry standards
 - Less tacit knowledge means more effective education

A paucity of systematic knowledge?

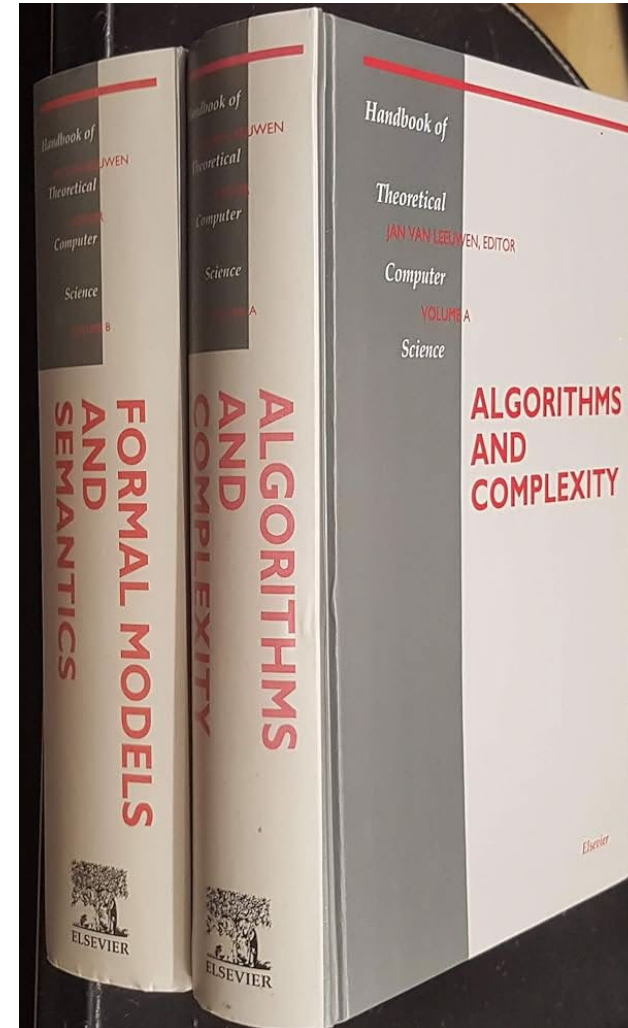
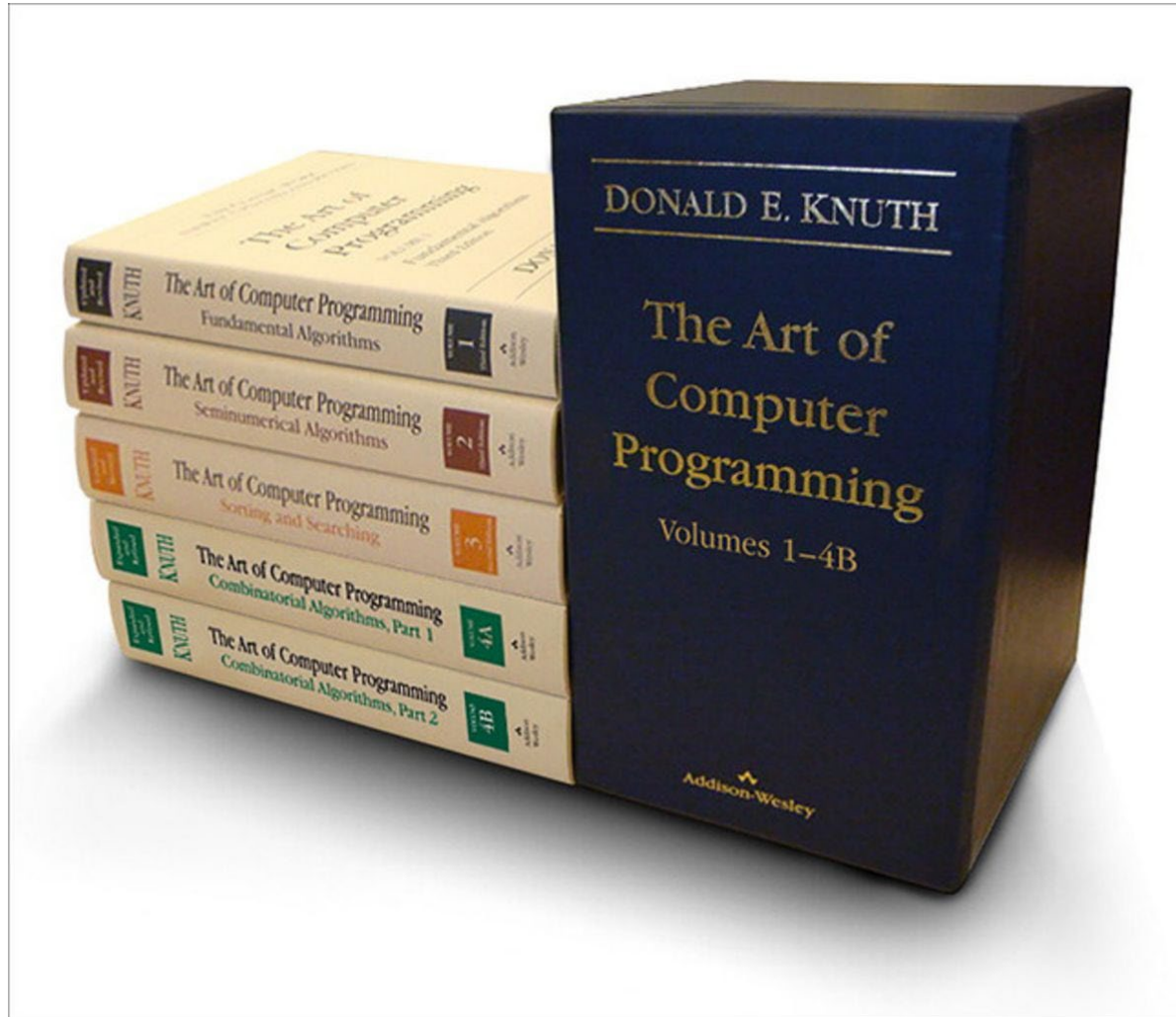
- Most Systems papers (academic and industrial) **recast** our tacit knowledge in some "new" setting
⇒ limited impact beyond the system itself
- We create **too many systems** but **identify too few principles**.

Result: there's **too much related work** to absorb

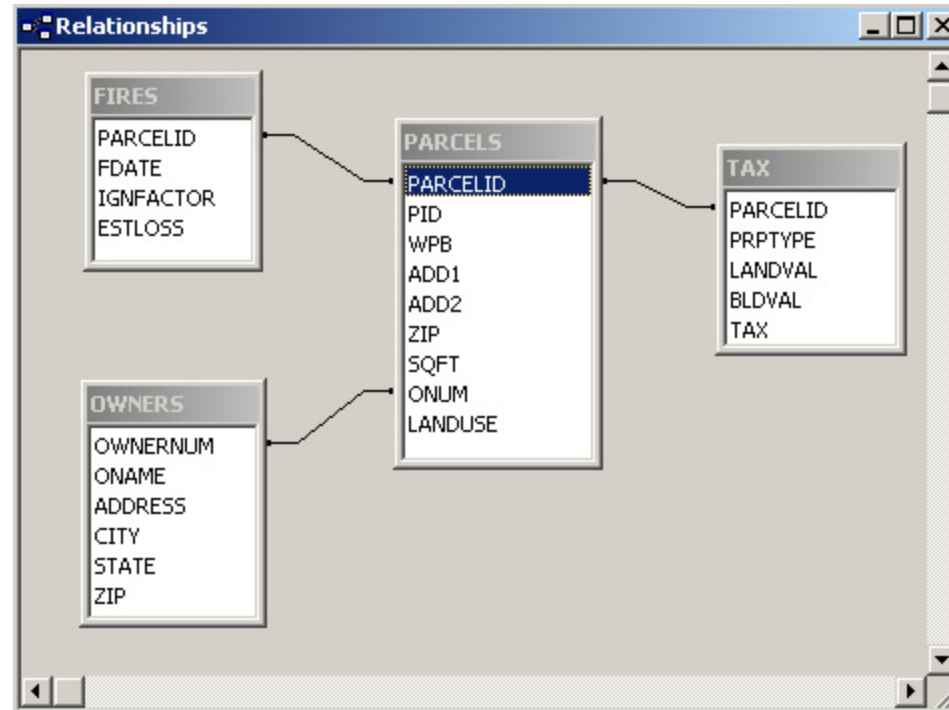
- No summary of what we have learned from all of it.
- No systematic framework in which to fit it.

**What if we tried to
systematize our
knowledge?**

Other fields have done this

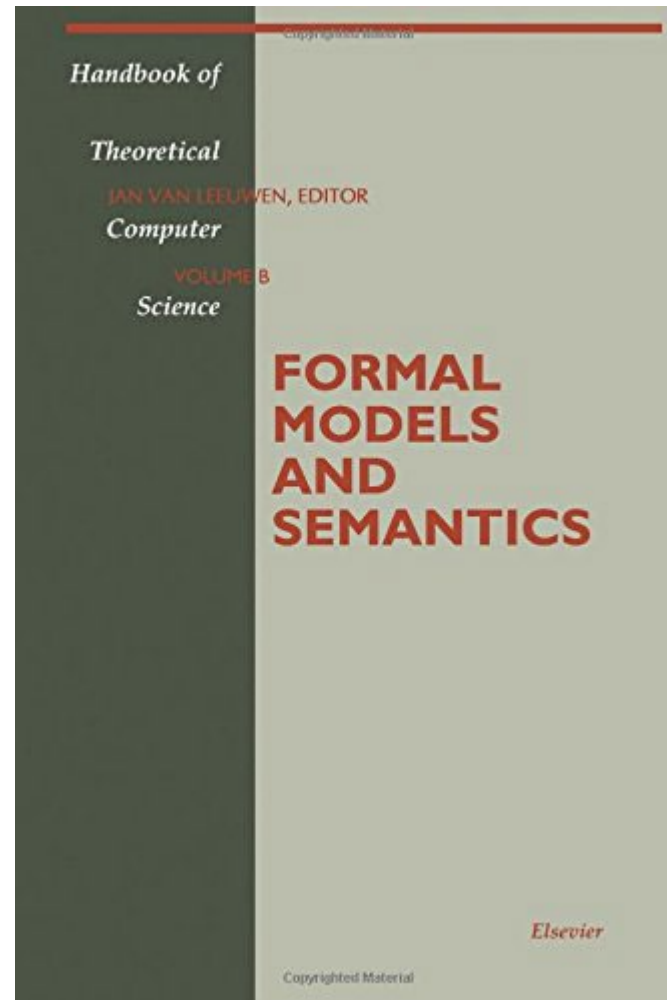


Databases – since way before Windows 95



https://web.mit.edu/11.521/www/lectures/lecture10/lec_data_design.html

Programming Languages (!)



So, what to do?

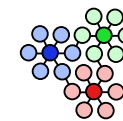
What's our goal? Let's start with a mission.

*To put **knowledge** about building computer systems
on an ever-more **structured, systematic** footing.*

So what can be done?

Some options:

- ~~Give up and go all-in on automating systems research with AI~~
- Change the publishing system to incentivize community
- Build shared research **artifacts**
- Write, and value, **survey** papers
- Create better **textbooks**
- Build **reference models**



emulab



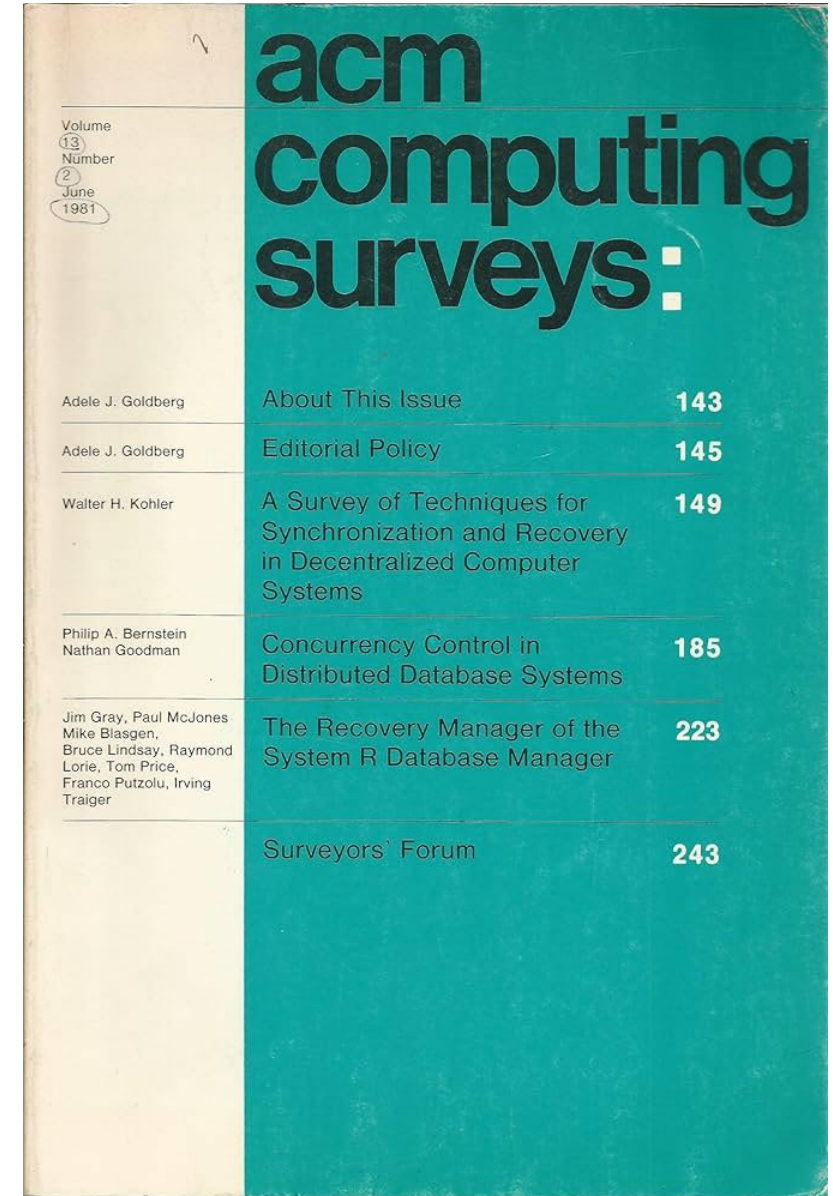
PLANETLAB

An open platform for developing, deploying, and accessing planetary-scale services

acm computing surveys:	
Volume 12, Number 2, June 2017	
Adria J. Goldberg	About This Issue 143
Adria J. Goldberg	Editorial Policy 145
Walter H. Kuster	A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems 149
Philip A. Bernstein Noah Goodman	Concurrency Control in Distributed Database Systems 185
Jim Gray, Paul M. Jones Mike Shapiro Bruce Lindsay, Raymond Luthi, Tim Price, Franco Pizzuti, Irving Traiger	The Recovery Manager of the System R Database Manager 223
	Surveyors' Forum 243

Survey papers

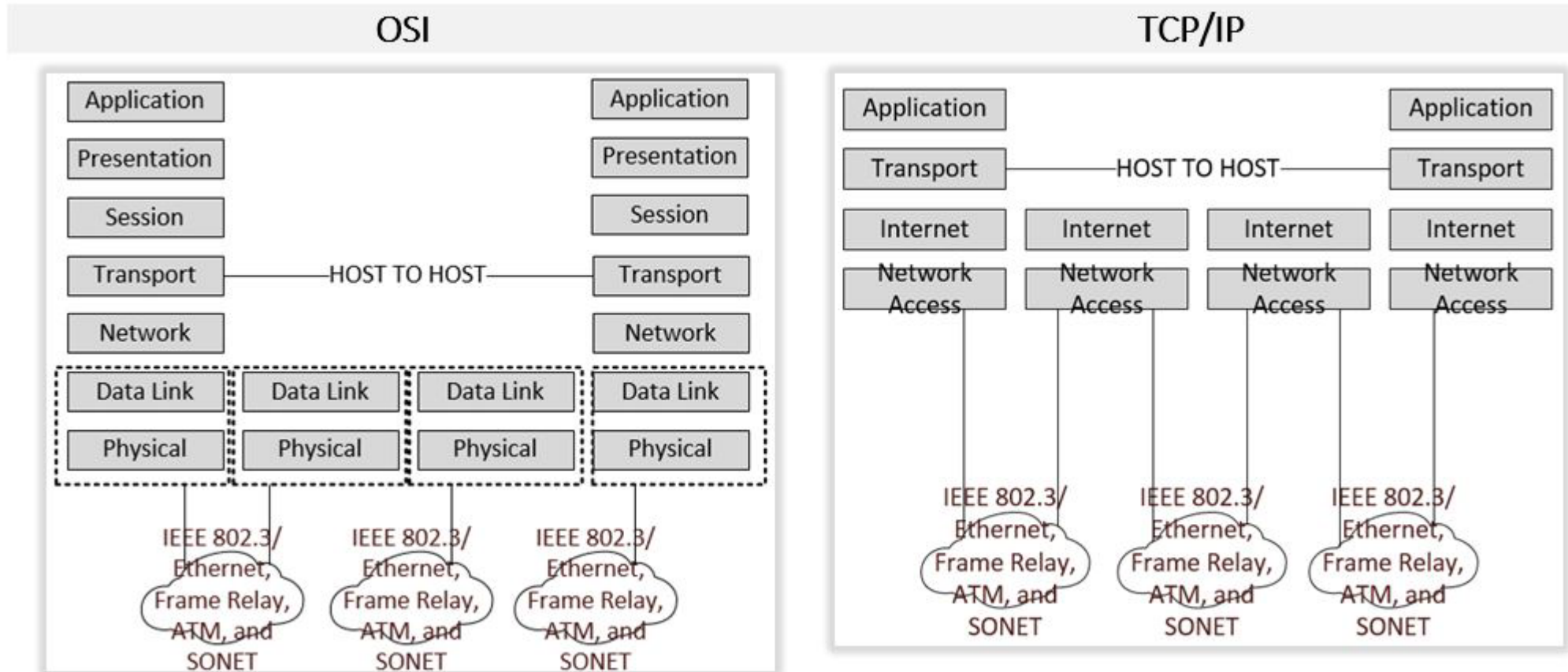
- We stopped reading (or valuing) ACS. Why?
 - Search engines? Now mostly slop.
 - Wilful ignorance? Where's the novelty?
 - More a symptom of the problem...
- A great survey paper defines a **taxonomy** or **conceptual framework**.
 - Unlikely that an LLM could do an original job
 - Useful to the author as a search engine?
- Often the starting point for a **reference model**



Volume 13 Number 2 June 1981	
Adele J. Goldberg	About This Issue 143
Adele J. Goldberg	Editorial Policy 145
Walter H. Kohler	A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems 149
Philip A. Bernstein Nathan Goodman	Concurrency Control in Distributed Database Systems 185
Jim Gray, Paul McJones Mike Blasgen, Bruce Lindsay, Raymond Lorie, Tom Price, Franco Putzolu, Irving Traiger	The Recovery Manager of the System R Database Manager 223
	Surveyors' Forum 243

Reference Models

We don't like reference models in Systems.



<https://commons.wikimedia.org/wiki/File:TCP-IP-ISO-REFERENCE-MODEL-COMPARISON-DIFFERENCE.jpg>

Rather different: X.900 (RM-ODP)

- Distributed Computing not networking
 - More comprehensive
 - Highly influential pre-web services
 - Still lots of good ideas from there
- Tool for thinking about design of complex distributed systems
 - Abstractions that help
 - Analytical tools
- Déjà-vu is a sign of a hidden RM...



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

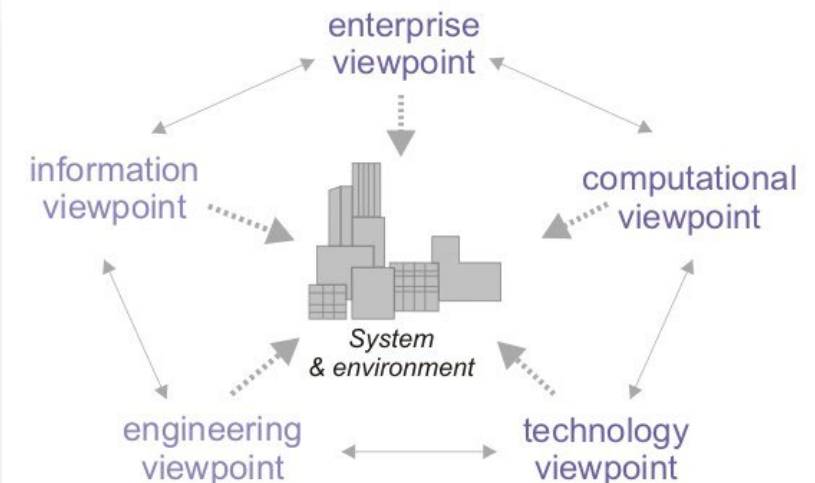
X.901

(08/97)

SERIES X: DATA NETWORKS AND OPEN SYSTEM
COMMUNICATION

Open distributed processing

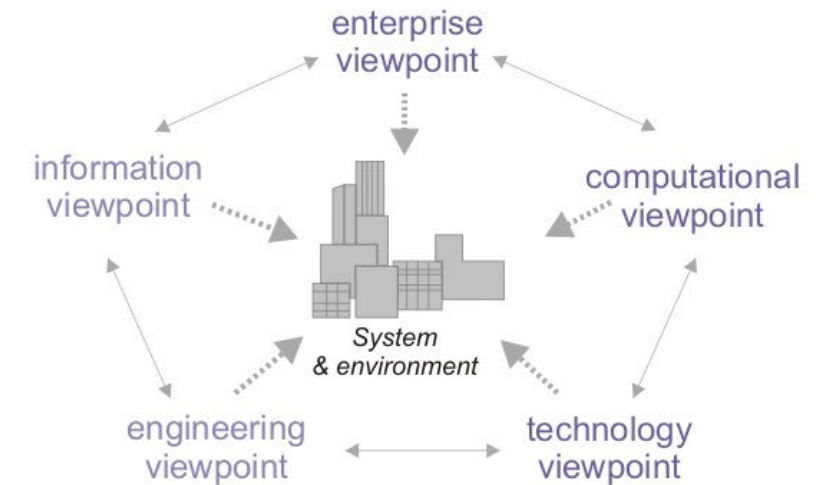
**Information technology – Open distributed
processing – Reference Model: Overview**



Reference Models matter

Systematic encoding of generalized knowledge.

- **Descriptive**: understand and survey the field
- **Prescriptive**: deal with complexity, build in a disciplined way



We are systems people:

we are the ones to trust with creating the right abstractions

Let's talk about Operating Systems

There is a crisis in Operating Systems

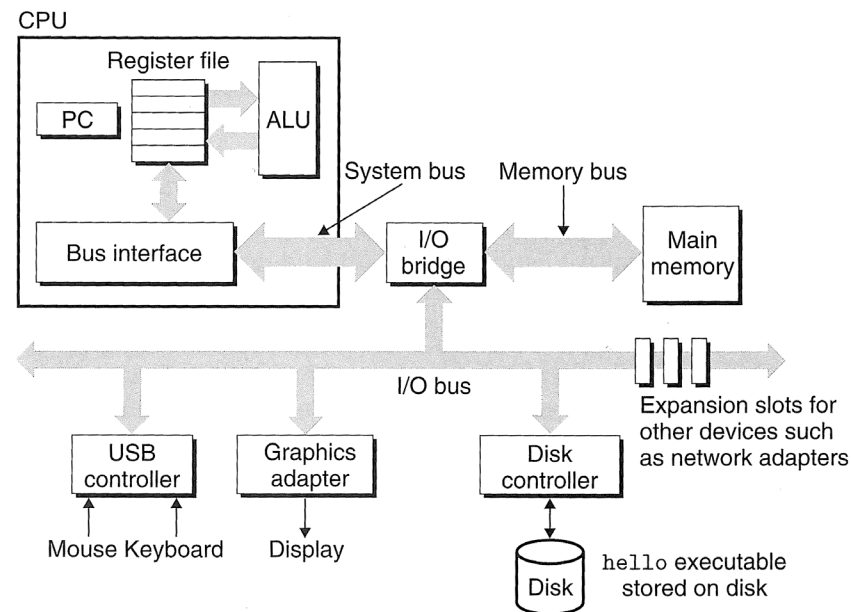
- The OS is **fundamental** to how all computers work
 - **Securely** multiplexes hardware resources
 - **Abstracts** them to application code
- What we call the OS *no longer does this*.
 - Based on a model of hardware that doesn't exist
 - Result: inefficiency, development cost, bugs, **security problems**.



How a modern OS views the computer hardware

Figure 1.4

Hardware organization of a typical system. CPU: Central Processing Unit, ALU: Arithmetic/Logic Unit, PC: Program counter, USB: Universal Serial Bus.

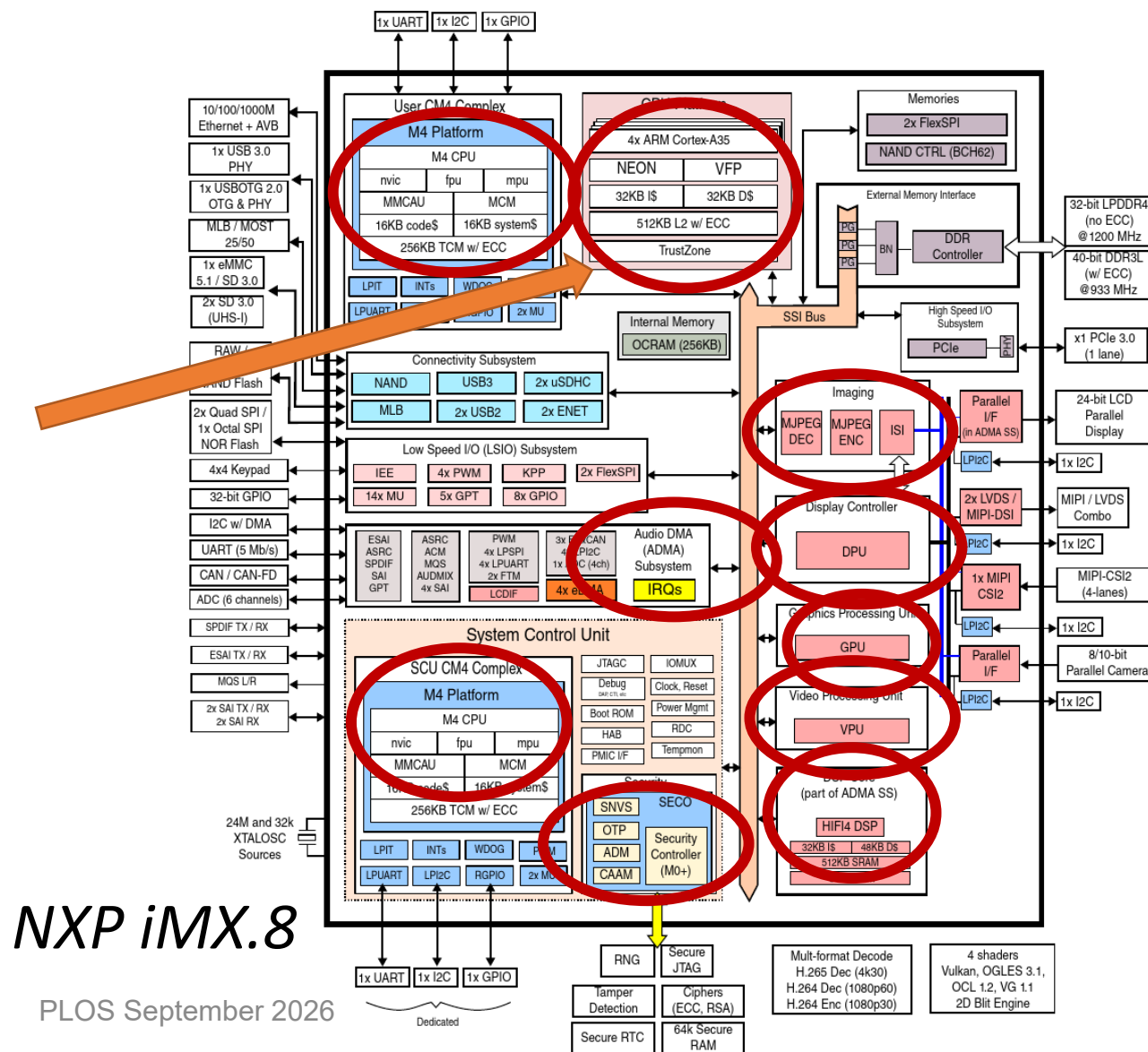


from: *Computer Systems, A Programmer's Perspective*, Bryant & O'Hallaron, 2011



Representative mobile System-on-a-Chip

Linux just runs here.



De facto OS:
sum total
of system
software on all
cores,
*trustworthy or
not!*

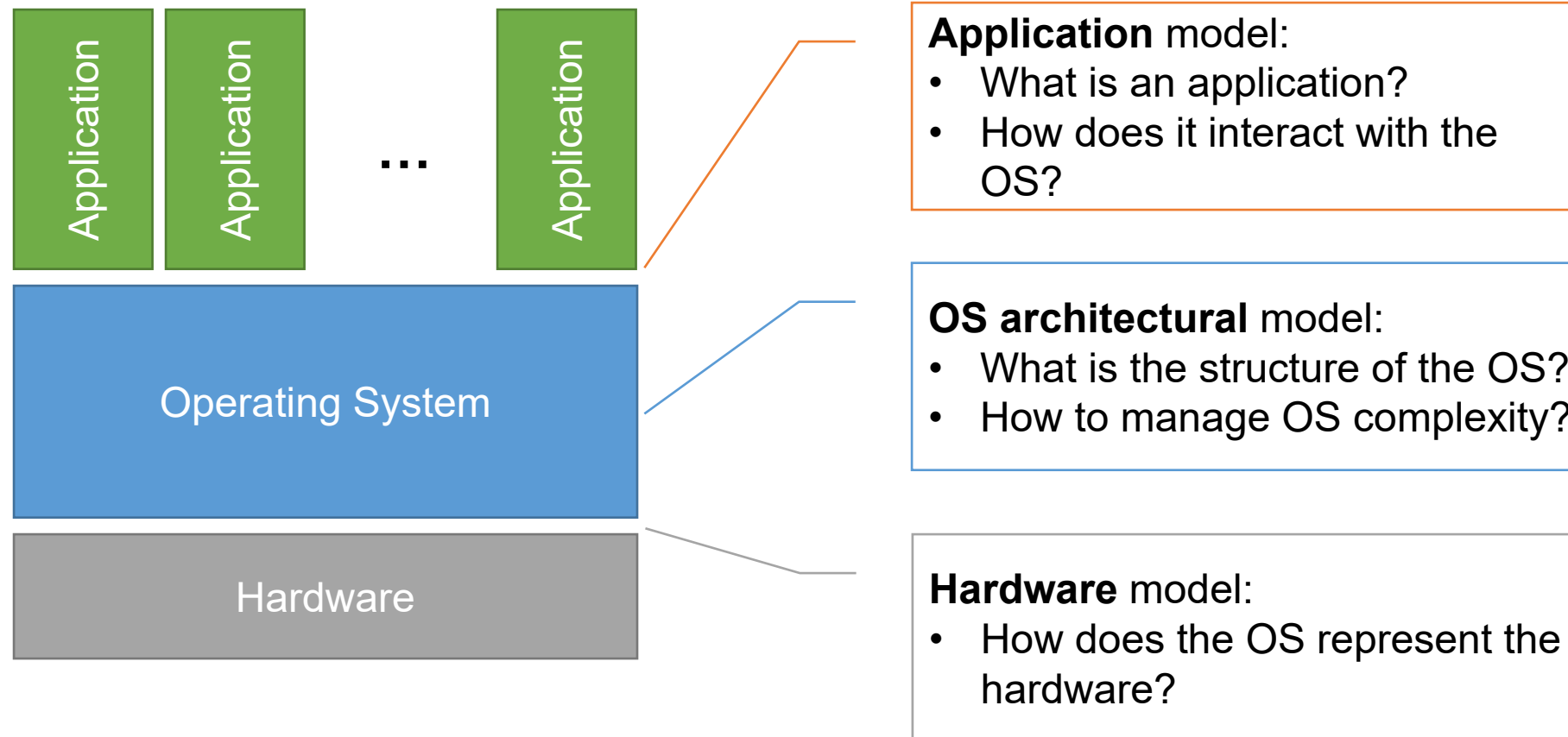
The OS problem is not going away

- Hardware is still getting more complex
- OS designs are not evolving
 - New OS designs make the same error
- Industry will not fix this.
 - Ad-hoc fixes cheaper in the short term
- Academia to date is reluctant
 - Seems a radical new approach needed
 - Much research focuses on modifications to Linux



OS models

Models in Operating Systems



Models in Operating Systems

Descriptive model:
Should capture all reasonable hardware...

Application model:

- What is an application?
- How does it interact with the OS?

OS architectural model:

- What is the structure of the OS?
- How to manage OS complexity?

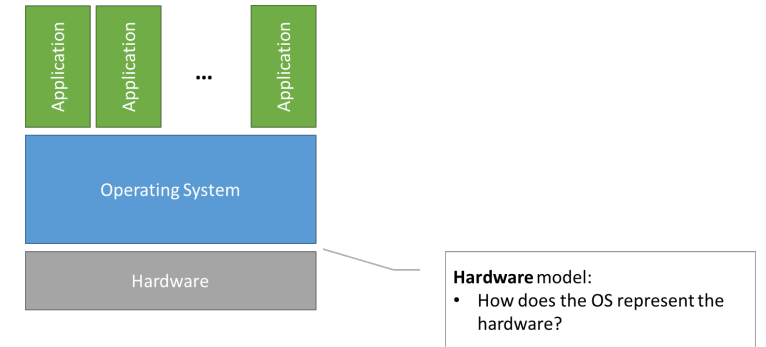
Hardware model:

- How does the OS represent the hardware?

The current hardware model

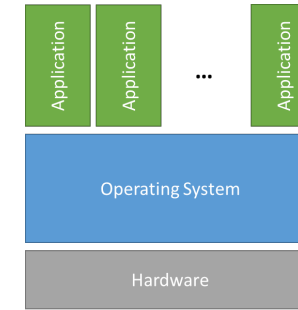
Classical Unix: Hardware Model

- Homogeneous cores
- User and kernel mode
- Single shared physical address space
- Paged MMUs, virtual address spaces
- Memory-mapped I/O
- DMA
- Array of interrupt vectors
- Devices are trustworthy



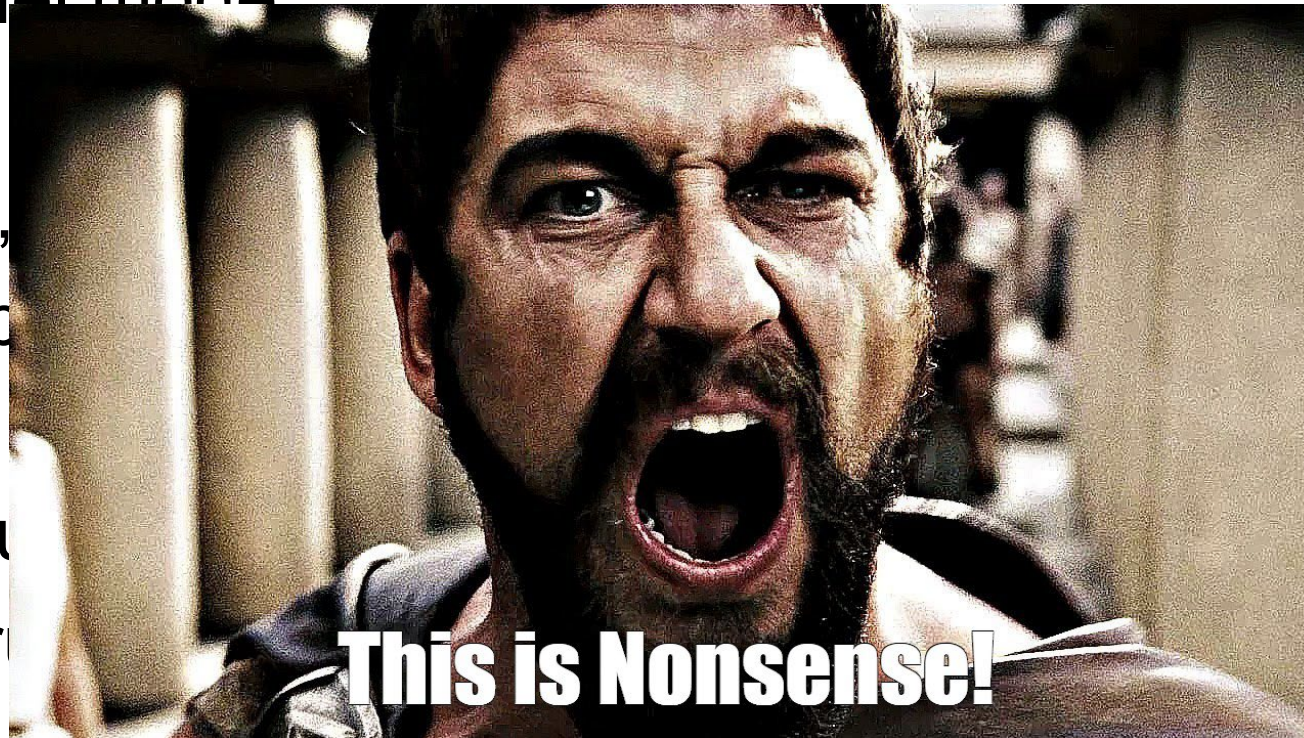
Classical Unix: Hardware Model

- Homogeneous cores
- User and kernel mode
- Single shared
- Paged MMUs,
- Memory-mapp
- DMA
- Array of interr
- Devices are tr



Hardware model:

- How does the OS represent the hardware?



A better hardware model

Sockeye: the semantics of hardware *as seen by software:*

1. What can hardware actually **do**?
2. How can it be **configured**?
3. What **properties** should it enforce?

- Imperative, timed model of hardware execution

Sockeye: Bug-finding *and* proofs for platform configurations and hardware based on reference manuals

Ben Fiedler ETH Zürich Zurich, Switzerland	Sedan Abdelgawad ETH Zürich Zurich, Switzerland	Teymour Aldridge ETH Zürich Zurich, Switzerland	Viktor Fukala ETH Zürich Zurich, Switzerland
Jan Häussermann ETH Zürich Zurich, Switzerland	Gamal Hassan ETH Zürich Zurich, Switzerland	Lars Leuthold ETH Zürich Zurich, Switzerland	Konstantin Lucny ETH Zürich Zurich, Switzerland
Max Wierse ETH Zürich Zurich, Switzerland	Samuel Gruetter ETH Zürich Zurich, Switzerland	Timothy Roscoe ETH Zürich Zurich, Switzerland	

Abstract

The ever increasing complexity of hardware platforms poses a challenge to systems programmers. Correctly programming a multitude of components, providing functionality *and* security, is difficult: semantics of individual units are described in prose, underspecified, and prone to inaccuracies. Rigorous statements about platform security are often impossible.

We are the first to address this problem for closed-source hardware platforms, by introducing a domain-specific lan-

ACM Reference Format:

Ben Fiedler, Sedan Abdelgawad, Teymour Aldridge, Viktor Fukala, Jan Häussermann, Gamal Hassan, Lars Leuthold, Konstantin Lucny, Max Wierse, Samuel Gruetter, and Timothy Roscoe. 2026. Sockeye: Bug-finding *and* proofs for platform configurations and hardware based on reference manuals. In *ACM SIGOPS 32nd Symposium on Operating Systems Principles (SOSP '26)*, September 29–October 02, 2026, Prague, Czech Republic. ACM, New York, NY, USA, 20 pages. <https://doi.org/10.1145/3830418.3843883>

Session 2C: Thursday afternoon!

What are today's architectural OS models?

Models in Operating Systems

Prescriptive models:

- Constrain the design
- Rule out some structures

Application model:

- What is an application?
- How does it interact with the OS?

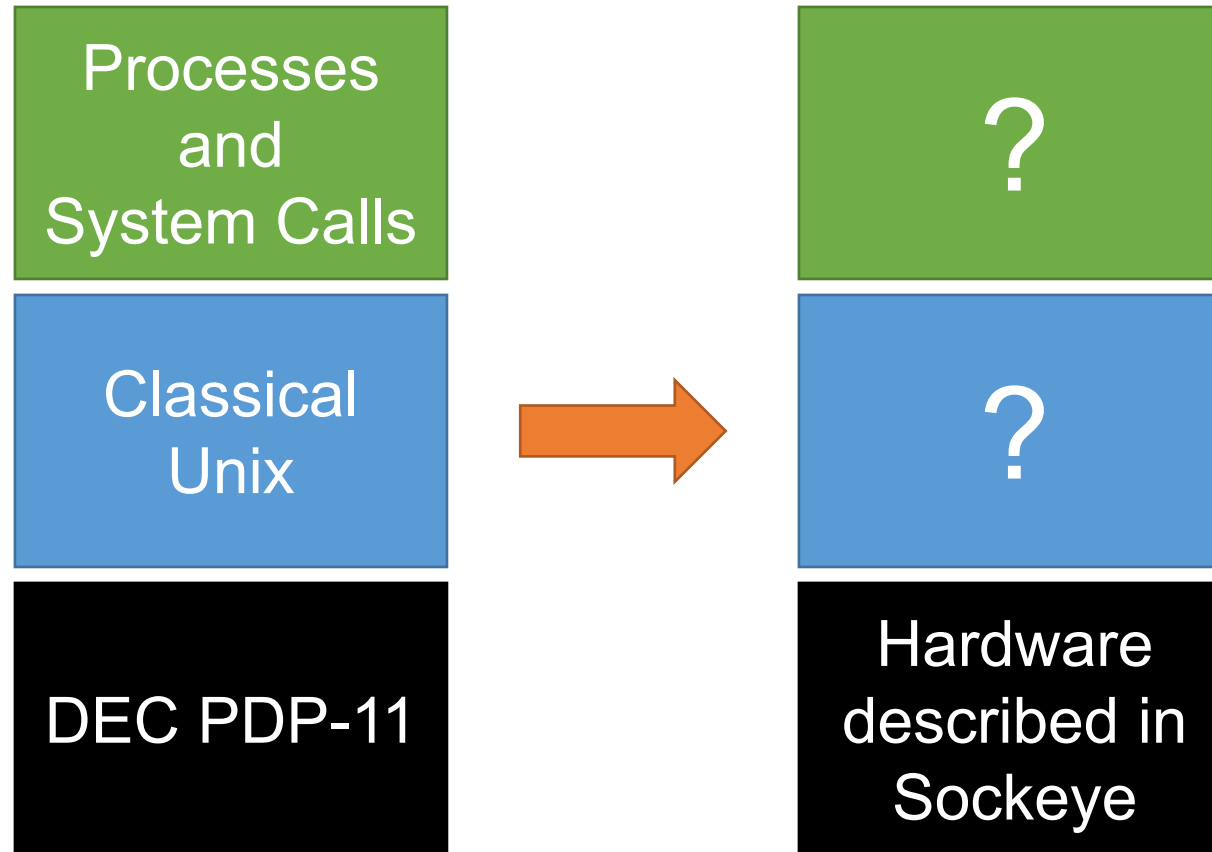
OS architectural model:

- What is the structure of the OS?
- How to manage OS complexity?

Hardware model:

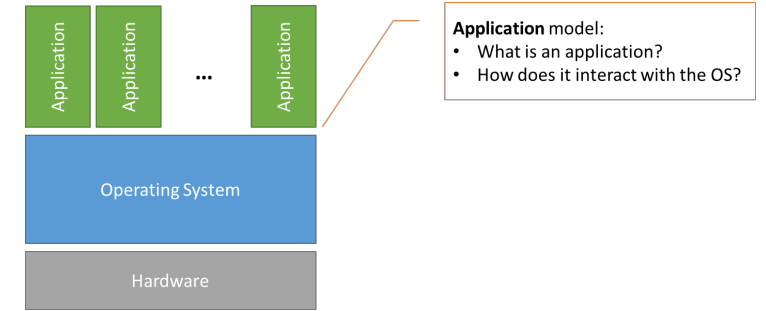
- How does the OS represent the hardware?

What does this mean for operating systems?



Classical Unix: Application Model

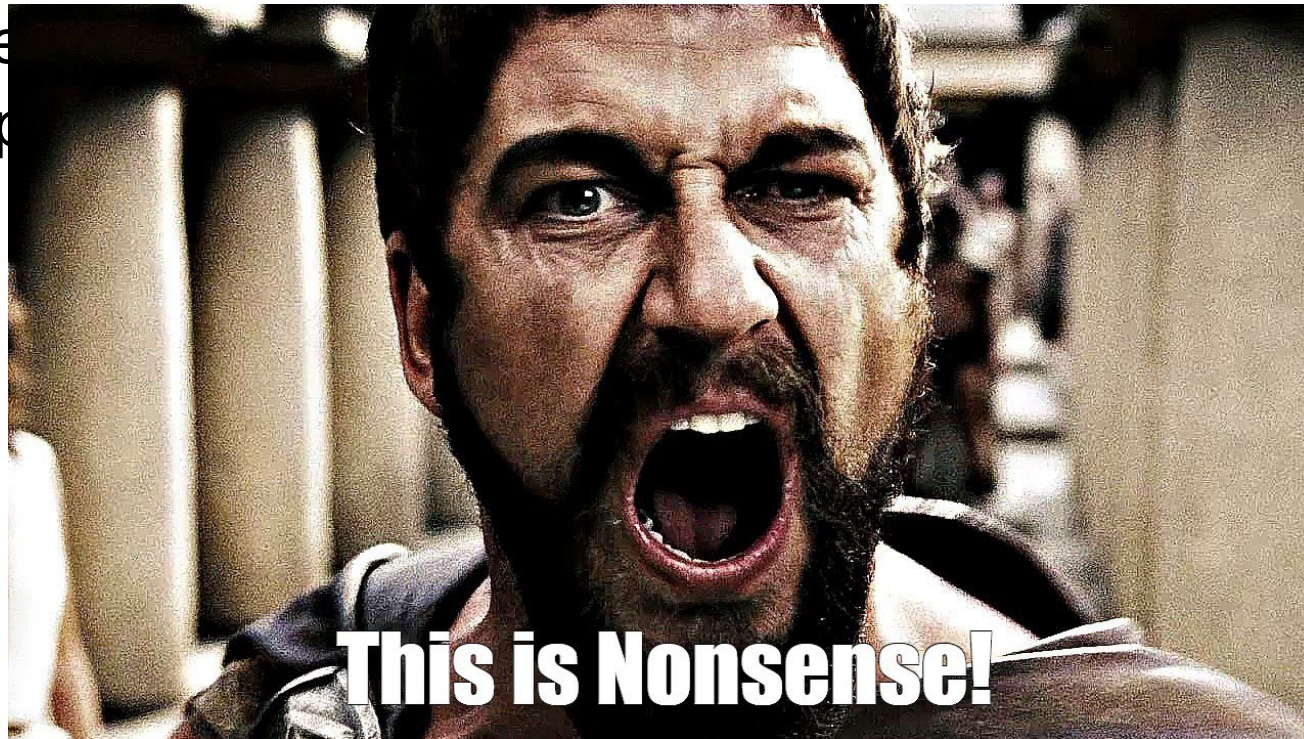
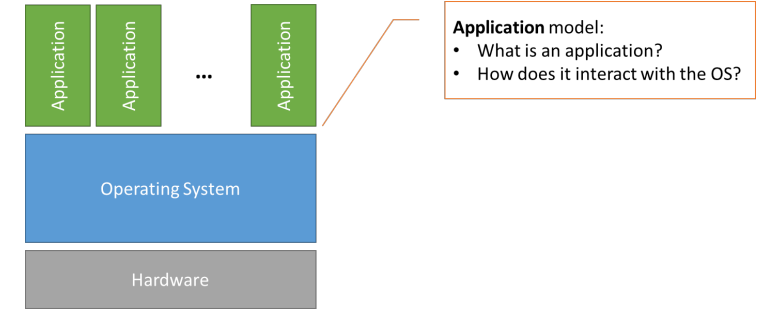
- Application = Process
 - Single private **virtual address space**
 - One, later many, homogeneous **virtual processors** (threads)
 - I/O and privileged operations via **system calls**



Classical Unix: Application Model

- Application = Process

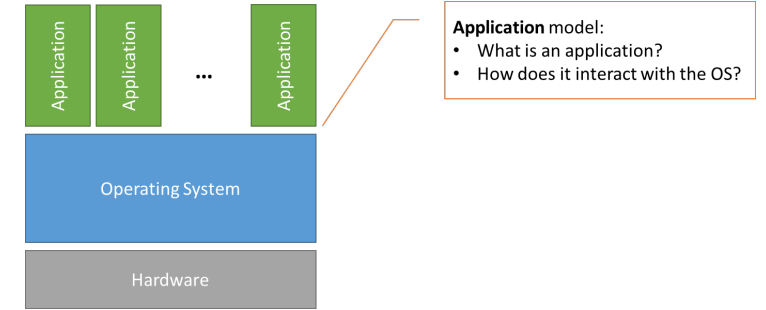
- Single private **virtual address space**
- One, late
- I/O and p



reads)

Classical Unix: Application Model

- Application = Process
 - Single private **virtual address space**
 - One, later many, homogeneous **virtual processors** (threads)
 - I/O and privileged operations via **system calls**
- Today: obviously not the case!
 - **Multi-process** applications are the norm
 - GPU **contexts**, eBPF programs, etc.
 - Sharing of application **services**
 - **Multi-tenant** applications
- Industry response:
 - Virtual machines, docker containers, etc.



Better analogy today

Even on a single machine, an application is:

- Distributed **collection** of processes
- Each with constraints on **where they execute**
- Managed as a **single unit**
- Communicating via **channels**

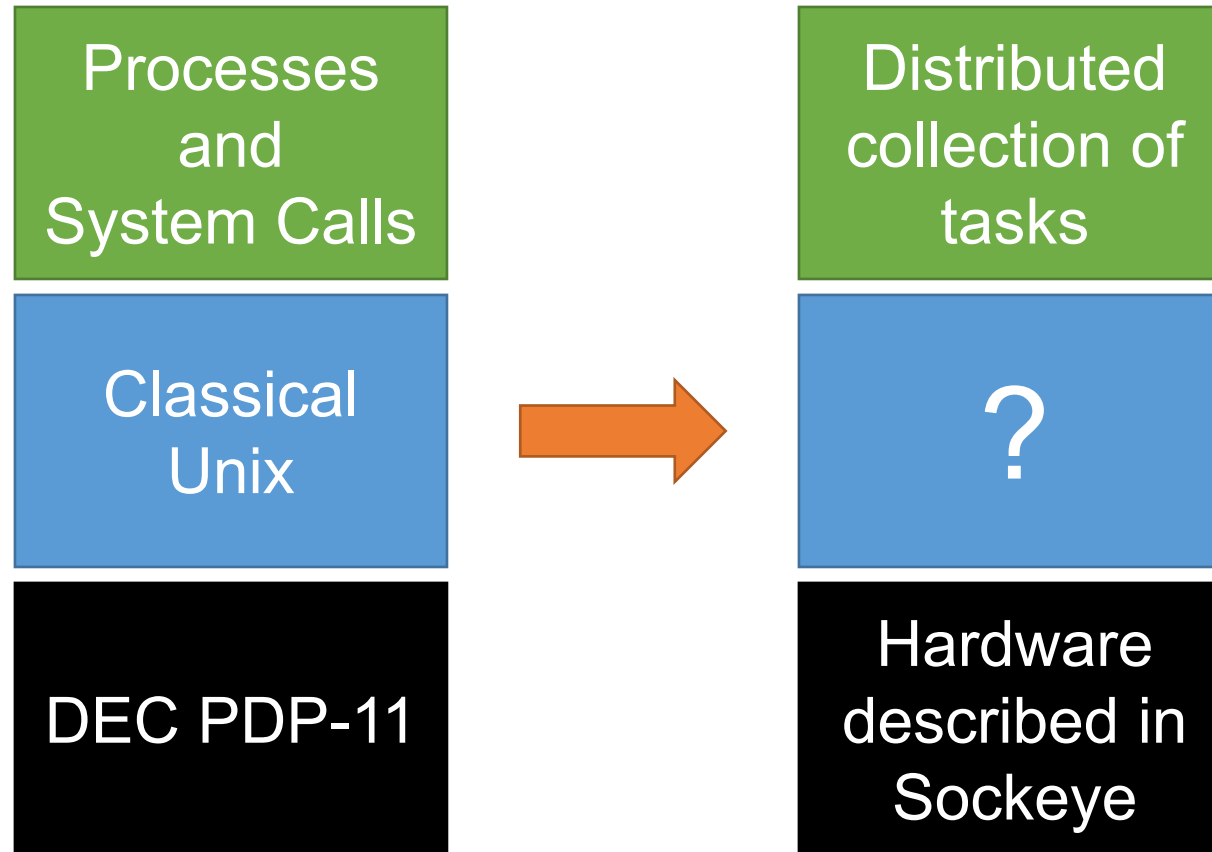
OS manages and allocates execution environments.



kubernetes

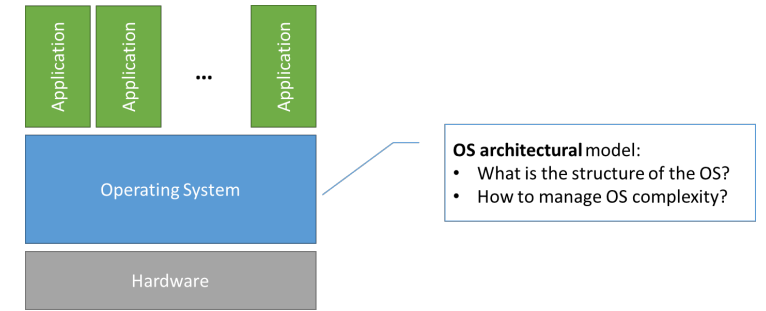


What does this mean for operating systems?



Classical Unix: OS architecture model

- Kernel
 - Single shared memory, multithreaded program
 - Loadable binary modules
 - Recently, eBPF extensions
 - Entire kernel is the Trusted Computing Base.
- Daemon processes
 - Varying rights for privileged kernel operations
- Peripherals
 - Abstracted behind ad-hoc software drivers



Classical Unix: OS architecture model

- Kernel

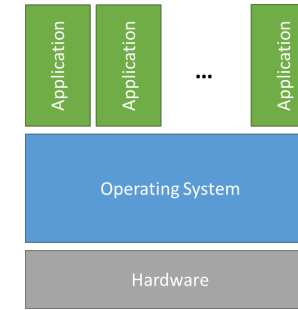
- Single shared memory, multithreaded program
- Loadable binaries
- Recently, eBPF
- Entire kernel

- Daemon processes

- Varying rights

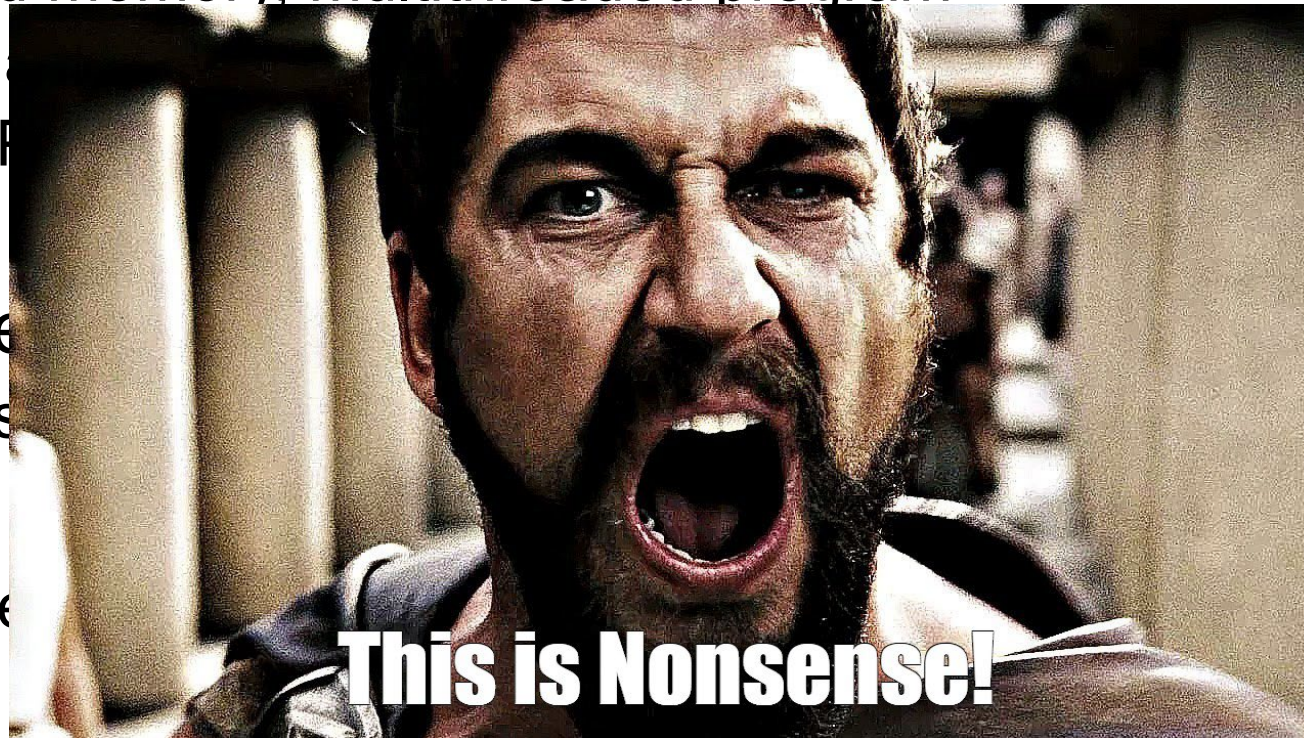
- Peripherals

- Abstracted by



OS architectural model:

- What is the structure of the OS?
- How to manage OS complexity?



Let's try to build a better OS model

A better architectural model

- Return to the **definition** of the OS:
 - that body of software that manages **the whole machine**
 - A single model for the whole system we reason about
- All the **cores**
 - Accelerators, board controllers, and all the rest
- All the **processor modes**
 - Firmware, reference monitors, management modes, bootstrap
- All the **trust domains**
 - Hidden cores, proprietary firmware
 - Sandboxed legacy code

What properties should we show?

- Most OS problems can be expressed as **integrity failures**
 - C.f. Biba model, "read-up, write-down"
 - Relative to specific context
- High-level correctness:
 - Application **non-interference**
 - Resilience to remote compromise
- **Confidentiality** violations are lower-level integrity failures
 - C.f. Bell-LaPadua – dual to Biba model.
 - E.g. process confidentiality $\Leftarrow$ integrity of kernel process structure

Contexts

Derived from Sockeye specifications!

- A **single-threaded sequential** execution environment
- Has a **class** which determines what kind of binary can execute in it
- Is a container for a **process**
- *Open question: can we encapsulate multithreaded environments like Linux VMs?*

Examples:

- *Aarch64 Exception Level 3*
- *x86 ring 3 with a specific page table and task descriptor*
- *An Intel TDX VM*
- *A descriptor-based DMA Network Card.*

Channels

- RPC Channel:
 - Asymmetric, authenticated, point-to-point, synchronous, unbuffered, reliable request-response channel.
 - Provided directly by hardware or created dynamically by software.
- Message Channel
 - One-way, one-to-one or one-to-many, asynchronous, possibly buffered, reliable, ordered message channel.
- *Experience suggests you need both, and both exist in hardware.*
- *Programming interface is not specified in the reference model.*

Examples:

- *x86 system call mechanism between user-space process and kernel on a single core.*
- *Shared-memory buffer writeable by one context and readable by others.*
- *A Unix-style “pipe”.*
- *Interrupts from one hardware context to another.*

This seems quite restrictive (prescriptive)

- Forces the OS to be a CSP-like system
 - Single-threaded contexts
 - Ordered messages
- Some evidence this can work
 - It's basically Barrelfish
 - Performance competitive with Linux
- Missing so far: *intra-OS trust and access control*.



Let's grab another good idea from distributed systems

In TAOS (a true distributed system)

- Contexts and channels are *principals*
- Messages convey statements
- Principals *speak-for* other principals

Sound basis for authorization in an OS

- Might be implemented using capabilities, PKC, hardware guarantees, etc.

Authentication in the Taos Operating System

EDWARD WOBBER, MARTÍN ABADI, MICHAEL BURROWS,
and BUTLER LAMPSON
Digital Equipment Corporation

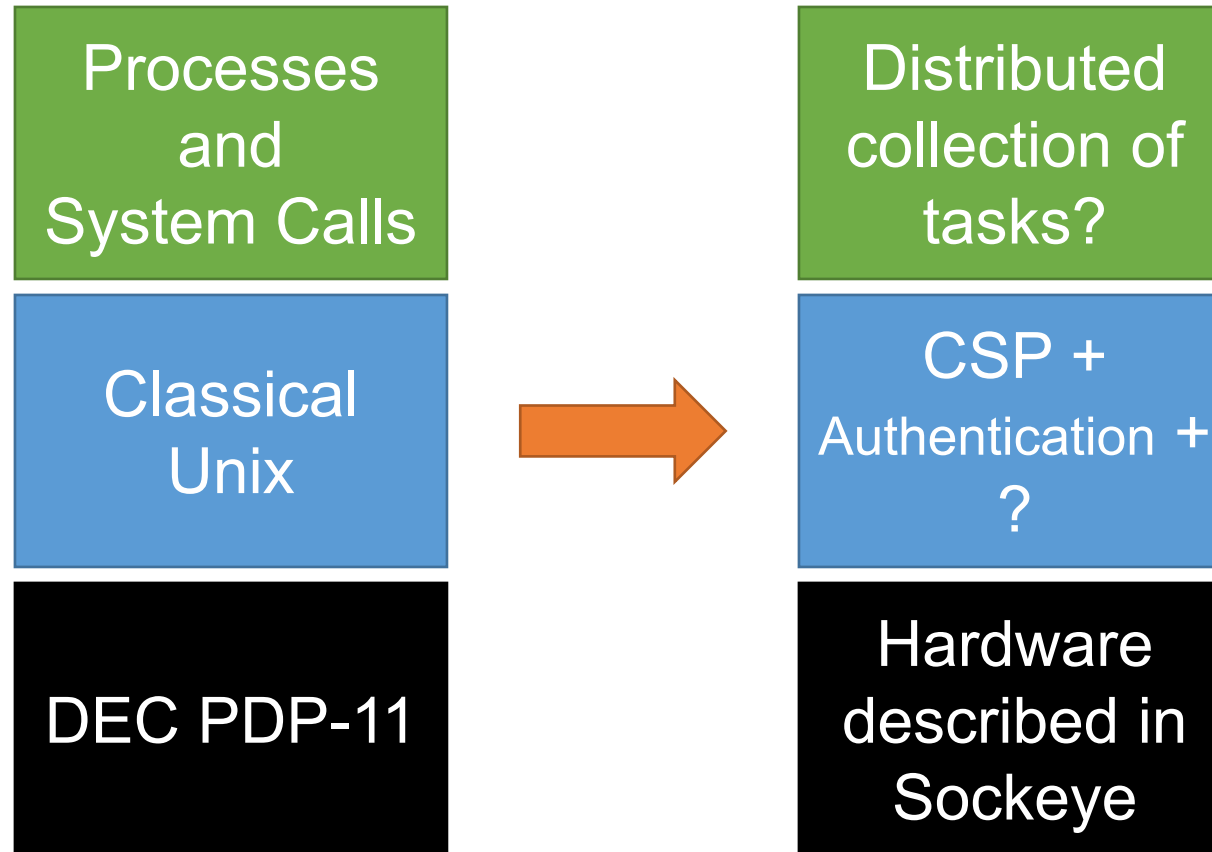
We describe a design for security in a distributed system and its implementation. In our design, applications gain access to security services through a narrow interface. This interface provides a notion of identity that includes simple principals, groups, roles, and delegations. A new operating system component manages principals, credentials, and secure channels. It checks credentials according to the formal rules of a logic of authentication. Our implementation is efficient enough to support a substantial user community.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems; D.4.6 [Operating Systems]: Security and Protection—*access control; authentication*

General Terms: Security, Theory

Additional Key Words and Phrases: Cryptography, mathematical logic

What does this mean for operating systems?

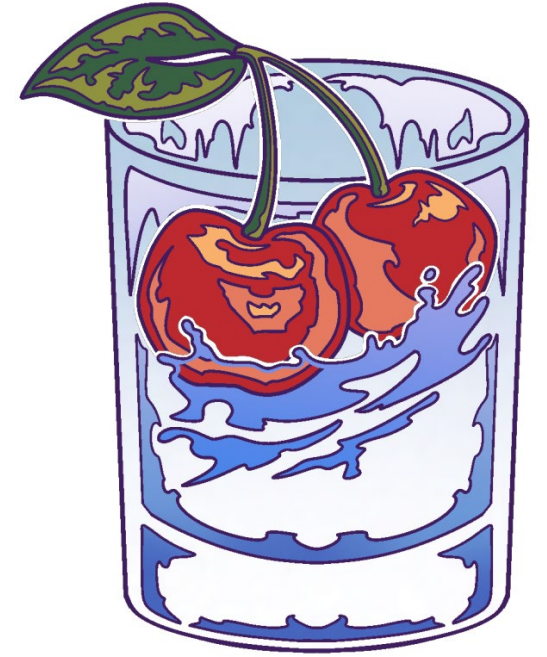


Lots of open OS questions!

- Some contexts can write each others' memory
 - Kernel K can write memory of a user process P
 - ⇒ K speaks-for P
- How can speaks-for relations be derived from Sockeye specs?
- Can you black-box untrusted devices as contexts?
- Is there really no need for fine-grained OS parallelism?
- How to express a Kubernetes-like application in the model?
- What breaks when you try to express existing OSes in this model?
- ...

Goal

- Assemble a **true OS** for a given platform...
 - *All the system software on the machine*
 - *Trusted and non-trusted*
 - *Existing/opaque and new*
- ... that also runs existing software ...
 - *Compatibility through virtualization and sandboxing*
- ... but which provides **formal guarantees of correctness**
 - *Down the **real hardware**, and across **all the cores***



Put **knowledge** about building computer systems
on an ever-more **structured, systematic** footing.

Many thanks!



Ben
Fiedler



Sam
Gruetter



Zikai
Liu



Roman
Meier



Jasmin
Schult



Pengcheng
Xu

Disclaimer:

- My field is **Operating Systems**.
- I've found myself solving problems that look like security and formal methods...