Data and Information Management in the Built Environment

Mark A. Austin

University of Maryland

austin@umd.edu

ENCE 688P, Spring Semester 2026

January 26, 2026

Overview

1. Definition and A Little History
2. Near-Term Challenges (2020-2060)
3. Features of Modern Computing
4. Urban and Global Applications
5. Appendix A: Cyber-Physical Systems
6. Appendix B: Engineering Sensor Systems

Part 03

Text

Appendix A

Cyber-Physical Systems

New Computing Infrastructure $\rightarrow$ New System Abstractions

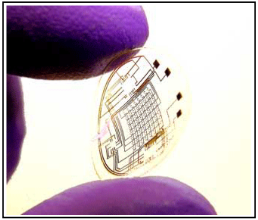
Appendix A: Cyber-Physical Systems

General Idea

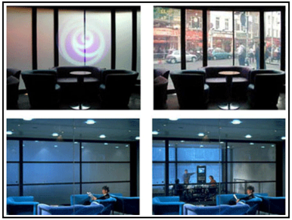
Embedded computers and networks monitor and control the physical processes, usually with feedback loops where computation affects physical processes, and vice versa.

Two Examples

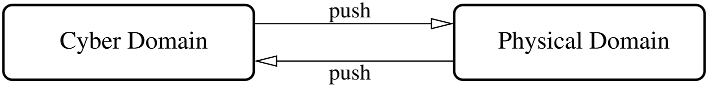
Programmable Contact Lens



Programmable Windows



Cyber-Physical Systems Overview



C–P Structure

Cyber capability in every physical component
Executable code
Networks of computation
Heterogeneous implementations

Spatial and network abstractions

- –– physical spaces
- –– networks of networks

Sensors and actuators.

C–P Behavior

Dominated by logic
Control, communications
Stringent requirements on timing
Needs to be fault tolerant

Physics from multiple domains.
Combined logic and differential equations.
Not entirely predictable.
Multiple spatial– and temporal– resolutions.

Cyber-Physical Systems

Physical System Concerns

- Design success corresponds to notions of resilience and reliability.
- Behavior is constrained by conservation laws (e.g., conservation of mass, conservation of momentum, conservation of energy, etc..).
- Behavior often described by families of differential equations.
- Behavior tends to be continuous – usually there will be warning of imminent failure.
- Behavior may not be deterministic – this aspect of physical systems leads to the need for reliability analysis.
- For design purposes, uncertainties in behavior are often handled through the use of safety factors.

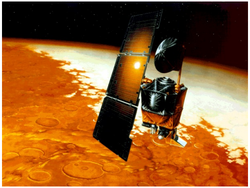
Cyber-Physical Systems

Software System Concerns

- Design success corresponds to notions of correctness of functionality and timeliness of computation.
- Computational systems are discrete and inherently logical. Notions of energy conservation ...etc... and differential equations do not apply.
- Does not make sense to apply a safety factor. If a computational strategy is logically incorrect, then "saying it louder" will not fix anything.
- The main benefit of software is that functionality can be programmed and then re-programmed at a later date.
- A small logical error can result in a system-wide failure.

Cyber-Physical Systems (Notable Failures)

Example 1. NASA's Mars Climate Orbiter, September 1999.



NASA's systems engineering process did not specify the system of measurement. One of the development teams used Imperial measurement; the other metric.

When parameters from one module were passed to another during orbit navigation correct, no conversion was performed, resulting in $125m loss.

Cyber-Physical Systems (Notable Failures)

Example 2. Denver Airport Baggage Handling System

1995. Baggage handling system is 26 miles of conveyors; 300 computers. Fixing the incredibly buggy system requires additional 50 percent of the original budget - nearly $200m.

2005. System still does not work. Airport managers revert to baggage carts with human drivers.

Source: Jackson, Scientific American, June 2006.

Cyber-Physical Systems (Error-Free Software)

Embedded computer systems and software need to deliver functionality that is correct and works with no errors.

CPS Design Requirements:

- Reactivity: System response need to occur within a known bounded range and delay.
- Autonomy: Systems need to provide continuous service without human intervention.
- Dependability: Systems need to be resilient to attack and hardware/software failures.
- Scaleability: System performance needs to scale with supplied resources.

Software for smart electronic devices is how Java got started !!!

Causes of Software-Related Accidents

Modern Software

Modern software is simply the design of a machine abstracted from its physical realization.

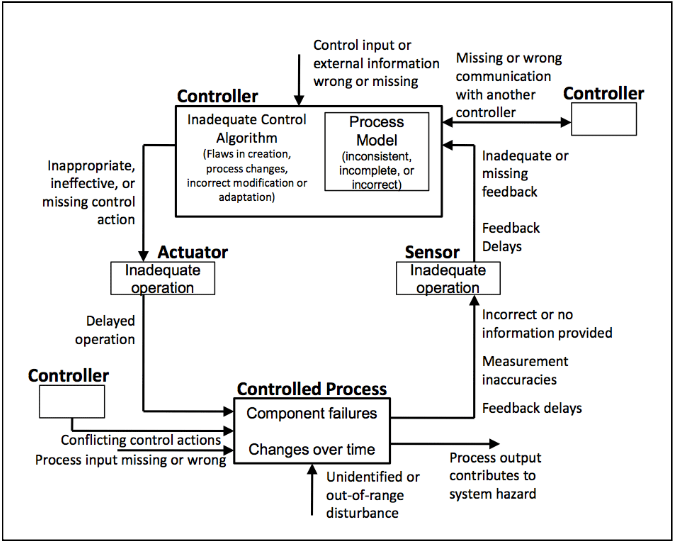
Software Accidents

Software accidents are usually caused by flawed requirements and not standard wear-out failures.

This includes:

- Incomplete (or wrong) assumptions about the operation of the controlled system or required operation of the software.
- Unhandled control system states and environmental conditions.

Engineering Sensor Systems (Error-Free Software)



Appendix B

Engineering Sensor Systems

Appendix B: Engineering Sensor Systems

General Opportunities for Sensing

- Enhanced levels of attainable performance ...
- Create new forms of functionality ...
- Improved economics and operational efficency (energy consumption).
- Improved resilience and agility ...

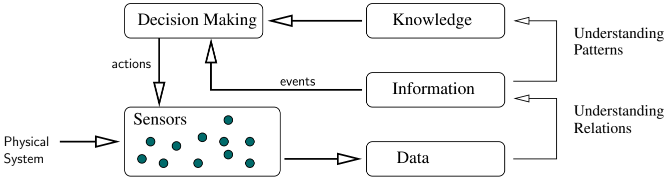
Sensing in the Built Environment

- We need sensors to serve as the eyes and ears of control and information systems designed to make buildings and cities more efficient and environmentally sensitive.

But how will such a system work?

Engineering Sensor Systems

Abstract Model for Sensor System Operations (Simplified!)



Implementation Options

- Human responsible for sensing and control.
- Automation (hardware and software) responsible for sensing and control.
- Human and automation systems cooperate in sensing and control.

Engineering Sensor Systems

Human-in-the-Loop Systems

Pros and Cons of Human Control:

- Human machine comes with five sensor types and reasoning capability builtin!
- Humans have slow response; sub-optimal performance; capabilities degrade with age. Approach isn't scalable.

Engineering Sensor Systems

Instrumented Systems:

Basic premise:

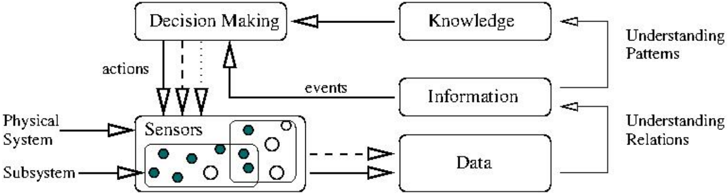
- Advances in computing, sensing, and communications technologies will allow for new types of systems where human involvement is replaced (or partially replaced) by automation.

Examples:

- Autofocus camera,
- Electronic systems in automobiles and planes → self-driving cars.
- Structural health monitoring / building automation systems.

Engineering Sensor Systems

Sensor networks and frameworks for decision making:

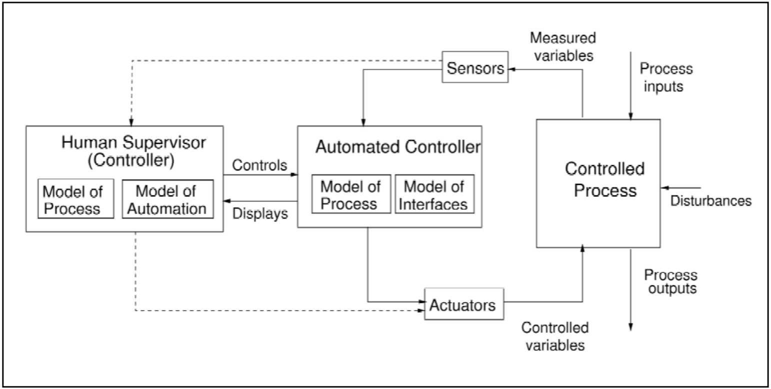


Chain of dependency relationships:

```
1.  improved performance <-- actions
2.  actions              <-- ability to identify events.
3.  identify events      <-- data processing
4.  data processing      <-- types and quality of data
5.  types and quality of data <-- sensor design and placement.
```

Engineering Sensor Systems

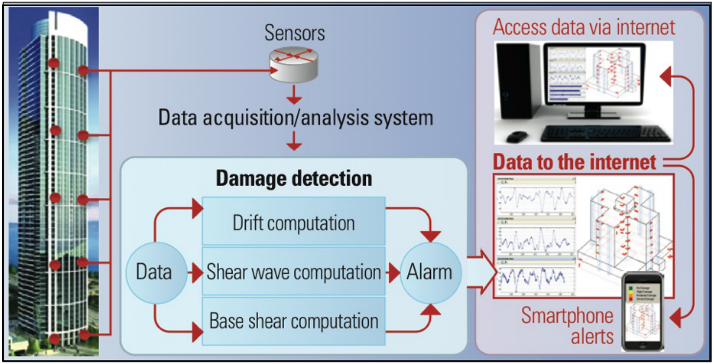
Human-in-the-Loop and Automated Control:



Source: Leveson, 2006.

Real-World Application (Structural Health Monitoring)

Flowchart of activities for real-time monitoring of instrumented buildings.



Source: http://earthquake.usgs.gov/monitoring/buildings/

Real-World Application (Modern Aircraft)

During the past three decades aerospace systems have seen increased use of electrical systems to achieve functionality.

Example 1. Boeing 777 → Boeing 787 (more electric aircraft).

Example 2. F-16 and F-35 Military Jets

F-16

F-35

Real-World Application (Modern Aircraft)

F-16 (production began 1974):

- 15 subsystems; $O(10^3)$ interfaces.
- Less than 40% of the functions managed by software.

F-35 (production began in 2006):

- 3-8 times the operational capability of previous aircraft.
- New sensor systems to support: situational awareness and targeting; sensor integration and data fusion.
- 130 subsystems; $O(10^5)$ interfaces.
- 90% of its functions are managed by software.