Data and Information Management in the Built Environment

Mark A. Austin

University of Maryland

austin@umd.edu

ENCE 688P, Fall Semester 2020

September 5, 2020

Overview

Part 2

Getting Started

Definition of Built Environment

Various Sources (Google, ScienceDirect):

- Human-made surroundings that provide for human activity, ranging in scale from buildings to cities.
- Includes supporting infrastructure: water supply networks; energy networks; transportation systems, communication systems.

Human Needs:

- Basic: Access to clean air and clean water.
- Health: Access to good medical services.
- Economic: Affordable low maintenance housing.
- Security: Protections against crime, environmental attack.

Definition of Built Environment

- Transportation: Good roads; parking; fast access to work.
- Educational: Access to good schools.
- Green Spaces: Access to parks, bike paths, etc.
- Retail: Access to shopping; reliable supply chains.
- Lifestyle: Access to social and recreational spaces.

Urban Planning and Engineering Concerns:

- Understand short- and long-term planning needs.
- Efficiency in design – aesthetically pleasing design.
- Efficiency in operations – better use of limited resources.
- Improved response to unexpected events.

Framing the Opportunity

We seek:

- Data-driven approaches to measurement of performance in the building environment and identification of trends and patterns in behavior.
- Solutions that account for unique physical, economic, social and cultural characteristics of individual cities.

Sources of Complication:

- Multiple domains; multiple types of data and information.
- Network structures that are spatial and interwoven.
- Behaviors that are distributed and concurrent.
- Many interdependencies among coupled urban subsystems.

Framing the Opportunity

Systems Perspective:

- Entities in the built environment have both system structure and system behavior ....

Decision makers use behavior modeling to understand:

- Sensitivity of systems to model parameter choices.
- Influence of resource constraints.
- Potential emergent interactions and propogation of cause-and-effect relationships.
- Identification of parts of the systems that are vulnerable.

Cannot play with a real building/city – so a reasonable first step is data-driven building science in gaming environments ...

Framing the Opportunity

Premise of this Class:

- Data mining and machine learning technologies can enhance (not destroy) the built environment.

Basic Questions:

- What are the challenges facing the built environment in the time frame 2020-2060?
- Is present-day technology where it needs to be to make a worthwhile contribution?
- What will the data mining do? What will the machine learning do?
- Are there opportunities for AI, data mining and machine learning to work as a team?

Features of Modern Computing

Key Question: **How can we use modern computing technologies to improve Civil Engineering Systems?**

Man and Machine (Traditional View)

Man	Machine
• Good at formulating solutions to problems. • Can work with incomplete data and information. • Creative. • Reasons logically, but very slow. • Performance is static. • Humans break the rules.	• Manipulates Os and 1s. • Very specific abilities. • Requires precise decriptions of problem solving procedures. • Dumb, but very fast. • Performance doubles every 18-24 months. • Machines will follow the rules.

Sensible Problem Solving Strategy

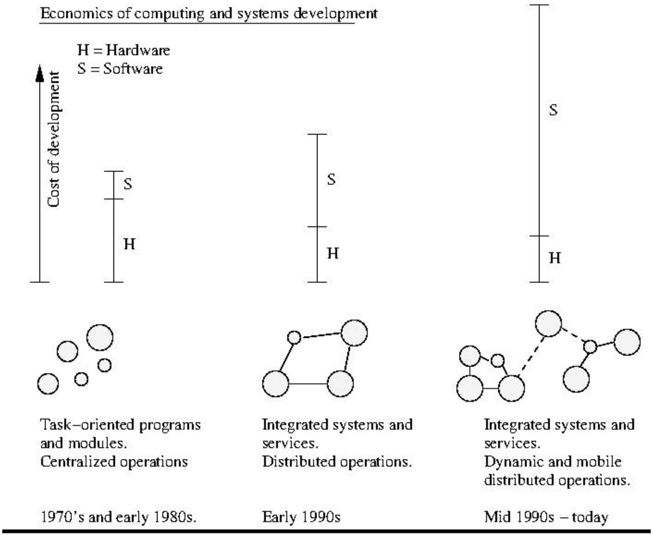
Let engineers and computers play to their strengths:

- Accelerates the solution procedure.
- Enables the analysis of problems having size and complexity beyond manual examination.
- Adds value in areas that will lead to long-term economic growth.

Getting things to work We need to:

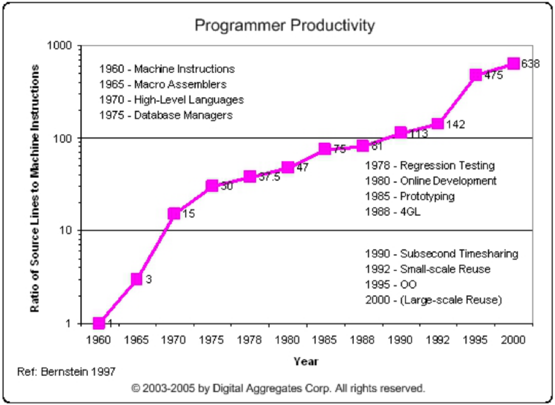
- Describe to the computer solution procedures that are completely unambiguous.
- Look at data, organization and manipulation of data, and formal languages.

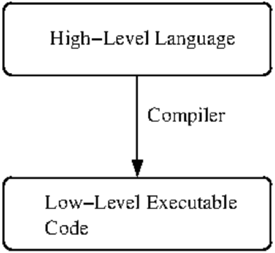
Expanding Expectations of Computing



Pathway to Improved Programmer Productivity

Increasing System Complexity: Software programmers need to find ways to solve problems at high levels of abstraction.





Evolution of Computer Languages

Computer Languages. Formal description – precise grammar – for how a problem can be solved.

Evolution. It takes about a decade for significant advances in computing to occur:

Capability	1970s	1980s	1990s
Users	Specialists	Individuals	Groups
Usage	Numerical computations	Desktop computing	E-mail, web, file transfer.
Interaction	Type at keyboard	Screen and mouse	audio/voice.
Languages	Fortran, C	MATLAB	HTML, Java

Popular Computer Languages

Tend to be designed for a specific set of purposes:

- FORTRAN (1950s – today). Stands for formula translation.
- C (early 1970s – today). New operating systems.
- C++ (early 1970s – today). Object-oriented version of C.
- MATLAB (mid 1980s – today). Stands for matrix laboratory.
- Python (1990s – today). A great scripting language.
- HTML (1990s – today). Layout of web-page content.
- Java (1994 – today). Object-Oriented language for network-based computing.
- XML (late 1990s – today). Description of data on the Web.

Post- 2000 Era

Imagine: What if COVID-19 had arrived in 2000?

- No iPhone, No iPad, No iTunes.
- No Facebook, No Instagram, No WhatsApp.
- No Google Maps, No Google Streetview.
- No Dropbox, No Zoom.

Recent Advances in Technology:

- Average internet speeds: In 2000, 0.07 Mbs; In 2009, 5-7 Mbs; In 2020, 100-200 Mbs; 5G, 1000-2000 Mbs.
- Cloud-based data storage and computational services (AWS).
- New languages: Swift $\rightarrow$ App development on iPhone/iPad.
- Many new types of sensors and methods of data collection.

Post- 2000 Era

New Computing Infrastructure → New Architectures, Languages, ...

Capability	2000-present	2020-2030
Users	Groups of people, sensors and computers.	Integration of the cyber and physical worlds.
Usage	Mobile computing. Control of physical systems. Social networking.	Embedded real-time control of physical systems.
Interaction	Touch, multi-touch, proximity.	....
Languages	XML, RDF, OWL.	New languages to support time-precise computations.

Post- 2000 Era

Just in case you were wondering:

Google: cloud computing origin of term

Cyber-Physical Systems

New Computing Infrastructure $\rightarrow$ New System Abstractions

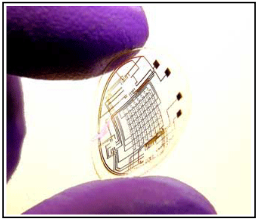
Cyber-Physical Systems

General Idea

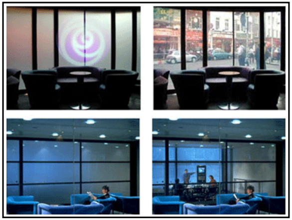
Embedded computers and networks monitor and control the physical processes, usually with feedback loops where computation affects physical processes, and vice versa.

Two Examples

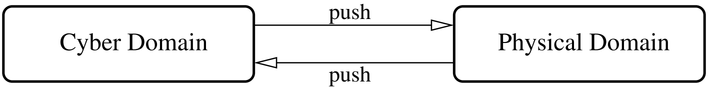
Programmable Contact Lens



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.

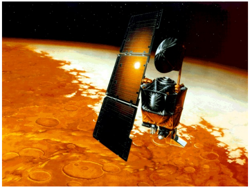
Cyber-Physical Systems

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.

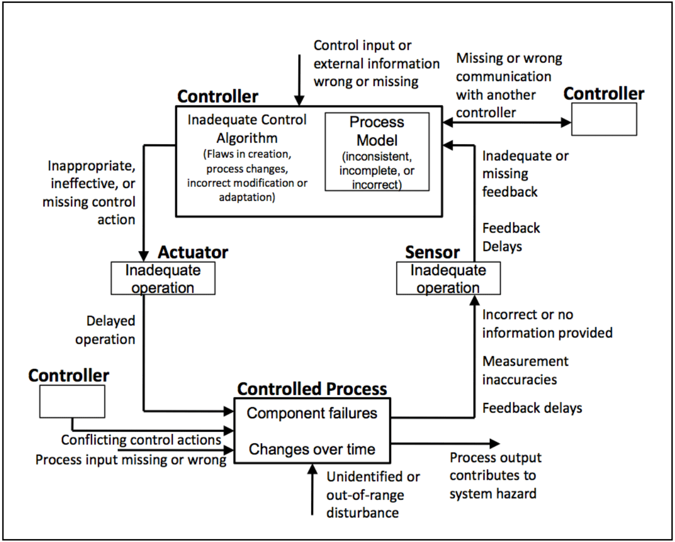
Software Accidents

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)

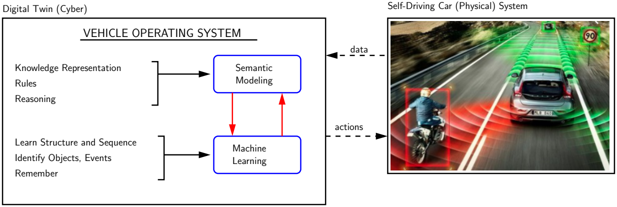


Digital Twin Systems

New Computing Infrastructure $\rightarrow$ New System Abstractions

Digital Twins (2000-today)

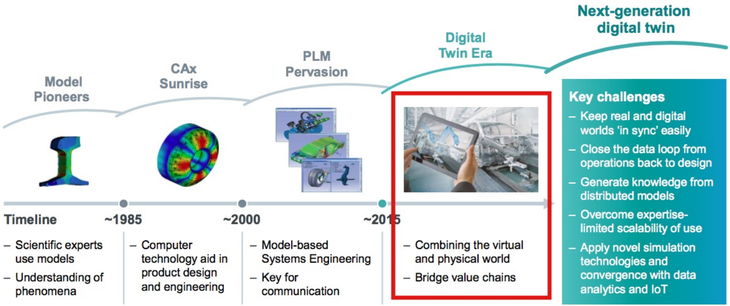
Definition. Virtual representation of a physical object or system that operates across the system lifecycle (not just the front end).



Required Functionality

- Mirror implementation of physical world through real-time monitoriing and synchronization of data with events.
- Provide algorithms and software for observation, reasoning, and physical systems control.

Digital Twins (Business Case + Applications)

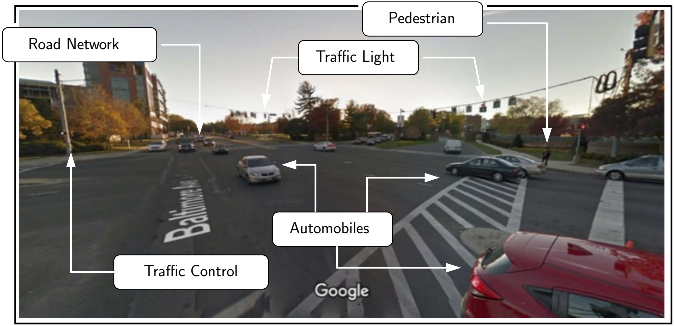


Many Applications

- NASA Spacecraft
- Manufacturing processes
- Building operations
- Personalized medicine
- Smart Cities
- ... etc.

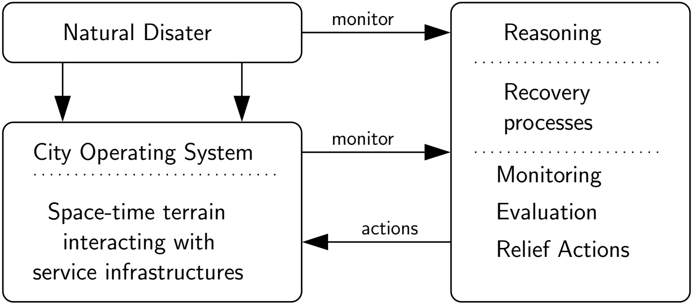
Digital Twin Application (Self-Drivng Car)

Goal. How to traverse traffic intersection safely and without causing an accident?

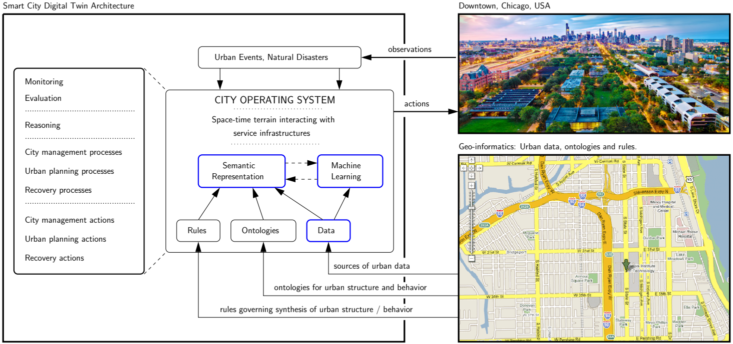


Required Capability. Observe, evaluate, reason, take actions.
Challenges. Multiple domains, multiple streams of heterogeneous data, event-driven behavior, dynamic, time critical.

Digital Twin: City Operating Systems

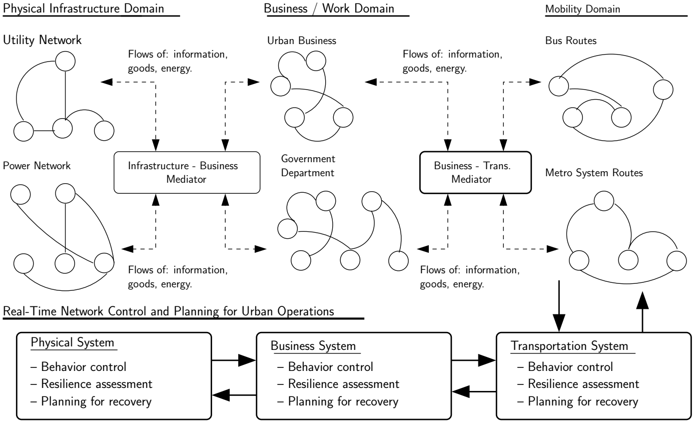


Smart City Digital Twins (2018-2019)



Required Capability. Monitoring and control of urban processes.
Complications. Potentially, a very large number of digital twins. Distributed decision making.

Smart City Digital Twins (2018-2019)



Requirements. Support for digital twin individuals and digital twin communities.

References

- Array of Things: See https://arrayofthings.github.io
- Austin M.A., Delgoshaei P., Coelho M. and Heidarinejad M. , Architecting Smart City Digital Twins: Combined Semantic Model and Machine Learning Approach, Journal of Management in Engineering, ASCE, Volume 36, Issue 4, July, 2020.
- Bello J.P. et al., SONYC: A System for Monitoring, Analyzing, and Mitigating Urban Noise Pollution, Communications of the ACM, 62, 2, 2019, pp. 68-77.
- Leveson N.G., A New Approach to Software Systems Safety Engineering, System Safety Engineering: Back to the Future, MIT, 2006.
- Tien J.M., Toward a Decision Informatics Paradigm: A Real-Time Information-Based Approach, to Decision Making, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, Vol. 33, No. 1, February, 2003.