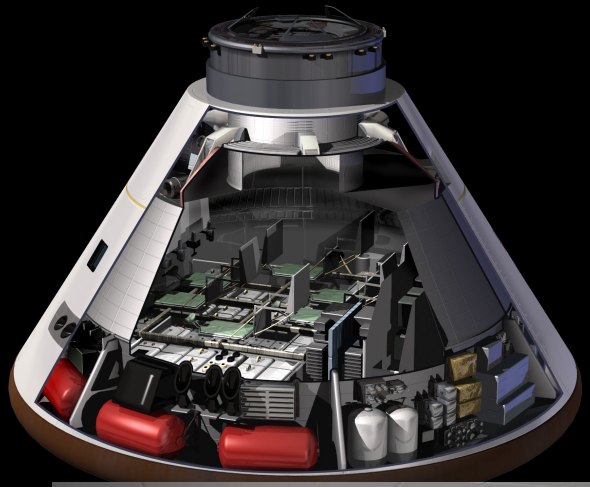
Software Modeling & Engineering for Resilience and Safety

백 옥기
白 鈺淇
O.K. Baek

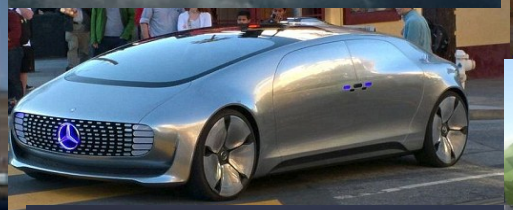
The Global Knowledge-based Economy and the Internet (WWW) are changing the ICT landscape

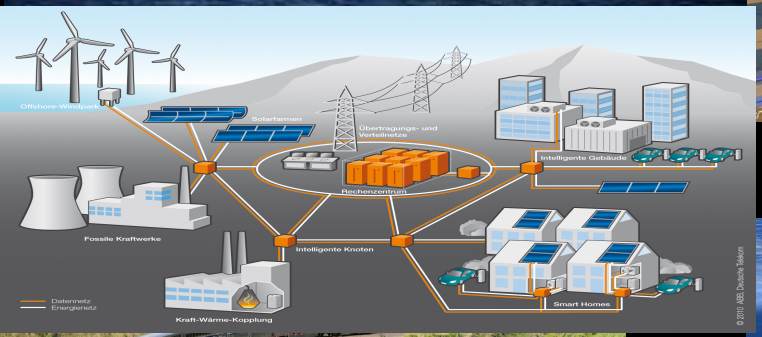
Next Big thing is The Internet of Things (IoT)

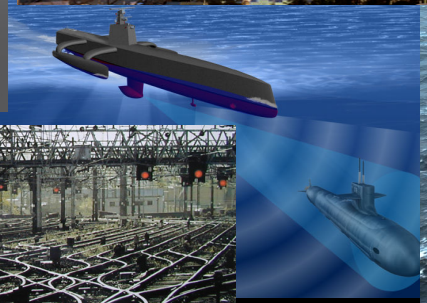
$11 Trillion/year industry by 2025 from airplanes to appliances, manufacturers in every industry. (source: IBM Watson Group Study 2015)



Cyber Physical System and the Internet of Things require new approach & method for software engineering



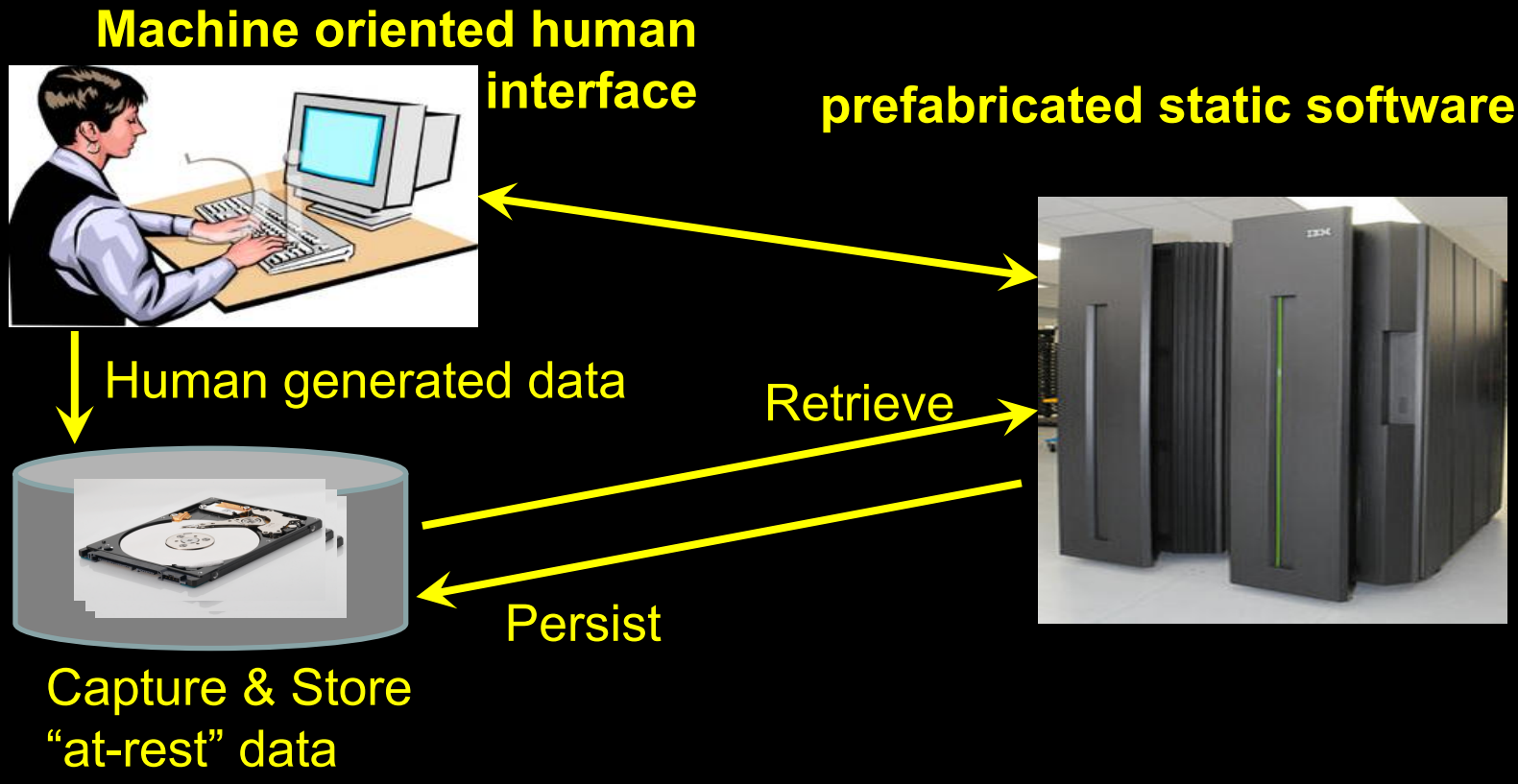




The software for CPS/IoT needs to

- Process events and analyze data (in-motion) in real time
- Consider the Context for data filtering
- Act on unanticipated errors and exceptions in real time (within milliseconds or microseconds)
  - E.g. Time window to react to an overloaded distribution substation in a power grid to avoid a power disruption
- Make ethical decisions on the spot
  - Runaway truck with hazardous materials
  - Military drone for a target in an urban location
  - Cargo train on a collision course
- Self-protect from unauthorized accesses for integrity

Traditional computing model is targeted for processing human-generated data in a machine-oriented interfaces; ICT belongs to the professionals and the business for information services.



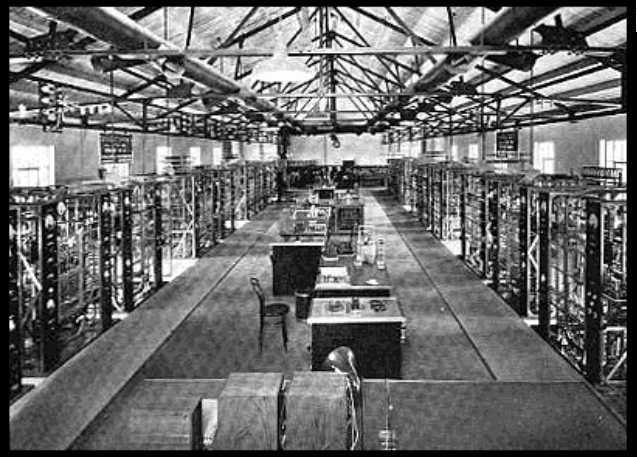
Current Computing Model & Software Engineering Methods are Inefficient, Inadequate and Unsafe!

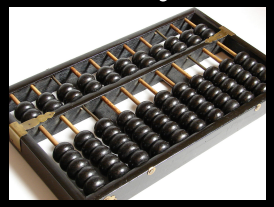
Themes of Today's Talk

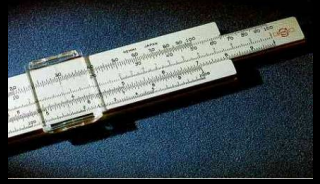
- Industry Trends and Challenges
- Assessment of current computing model and software engineering method
- Conventional approach and method and common mistakes negatively impacting safety
- New challenges and requirements for IoT/CPS
- Efficacy and efficiency of traditional methods and systems
- Novel approaches, methods and systems to address the requirements for 2020 and beyond

We have come a long way from hardware perspective

- 1 PFLOP/S Blue Gene
- Supercomputers (miniaturization, parallel clusters)
- VLSI, supercomputers
- LSI, general purpose computers
- Integrated Circuits, microprocessors
- Calculators
- Transistors
- Vacuum tubes + Core Memory
- Slide Rules
- Abacus







Evolution of Computing Models

- Batch processing: ISO FTAM/JTAM, CDC PTF/QTF, IBM JES/NJE
- Parallel processing
- OLTP: IBM CICS, IMS
- Client/Server & Distributed Computing (OSF DCE)
- OLAP, System R
- Message-Oriented Asynchronous Processing: MOM, MQSeries
- ORB, SOM/DSOM: OMG CORBA
- Big Data, Social Media
- Internet of Things: M2M interfaces
- RTAP, System S

Computing Model and Software Engineering Model remained, more or less, the same (SOA).

What is Software

"part of a computer system that consists of encoded information or computer instructions"

"includes computer programs, libraries and related non-executable data" - www.wikipedia.com

- Firmware
- Device drivers
- Operating System
- Middleware
- Application Framework
- Application + Configuration files + Application profile

Software Safety equates to

assurance of Quality, Integrity and Resilience of a System as a whole
and
mitigation of Anticipated Risks in advance

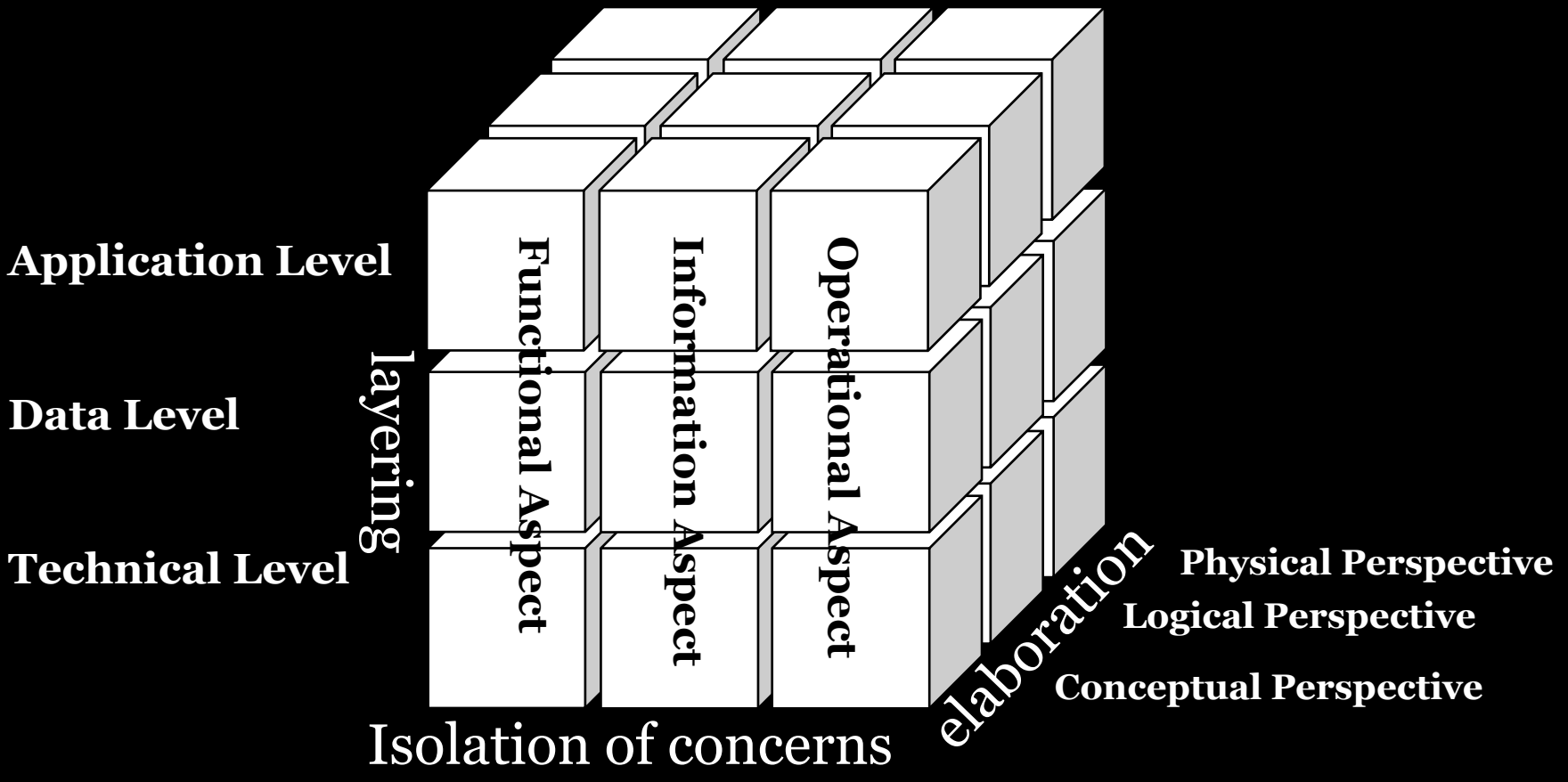
Safety should be addressed holistically

- All vertical system components: hardware, operating system, middleware, application
- End-to-end horizontal systems: from end-user devices through upstream and downstream systems
- End-to-end operating model and procedures
- Entire *lifecycle* from conception through decommissioning
- Safety is closely related to security, performance, availability, scalability, and reliability

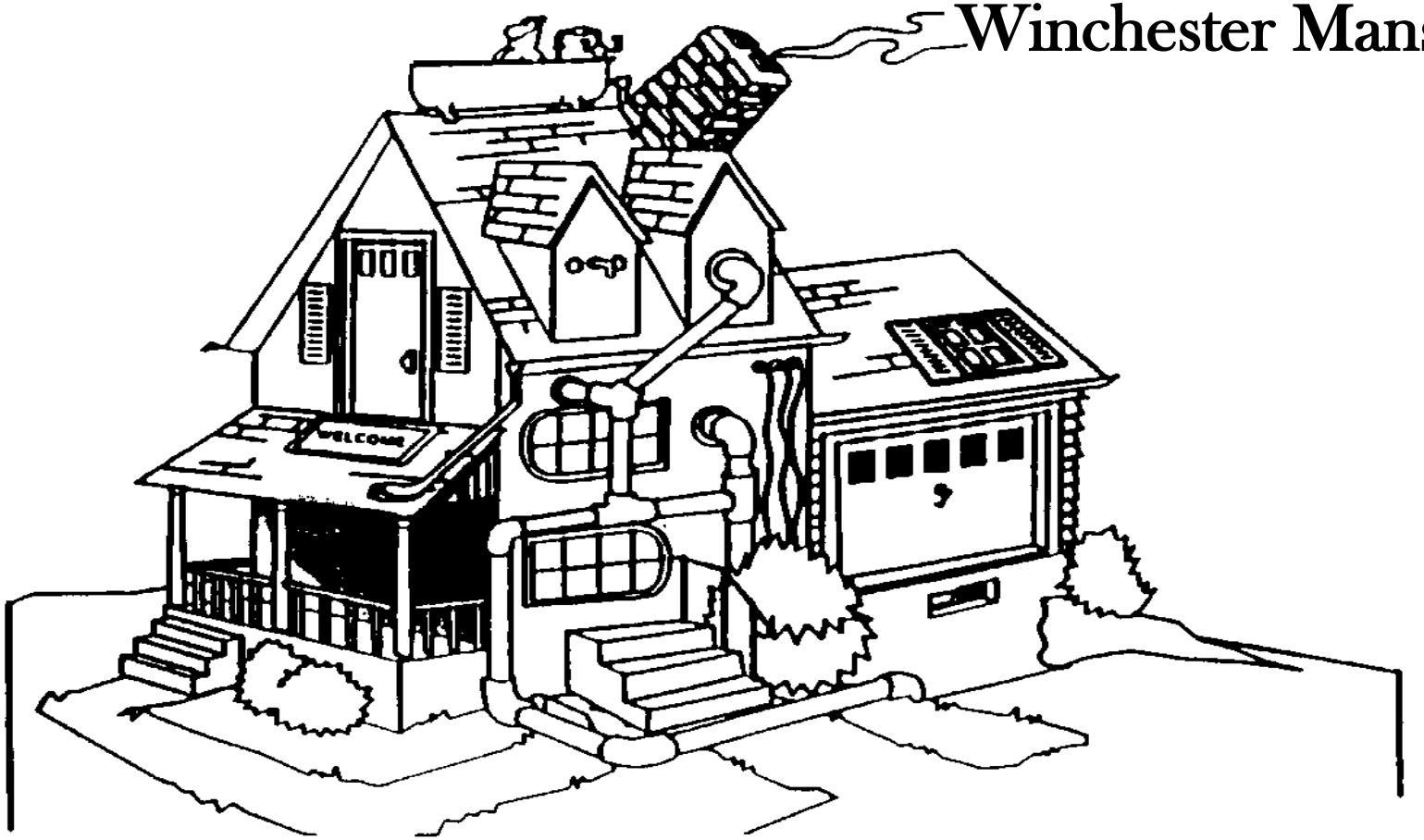
Key Areas influencing Safety and Resilience

- Modeling and Architecture
- Constraints
  - Budget
  - Time
  - Resources & Skills
  - Current system & process (paradigm?)
- Statutory requirements and Regulations
- Governance
  - Organization and Culture
  - Model and Process
  - Decision making process
- Methodology
  - Development method
  - Interdisciplinary communication & collaboration
  - System validation method
- Business Model & Process
  - Process, Function, Information, Security, Integration, …
  - Uncertainty (assumptions)
  - Ambiguity (source of defects)
- Target Operating Model

Quite often, discrepancies in modeling & architecture for Big Picture are the root cause of major problems



Point in time tactical decisions, deviating from the architecture and out of context, lead to:



Typical SDLC Lifecycle

- Conception with goals/objectives and Business case assessment
- Project definition (funding, staffing, planning)
- Requirement analysis
  - Functional capabilities
  - Operational requirements
- Modeling and Architecture
- Design
- Build
- Validation
- Deploy
- Maintenance and enhancement

SMART specification

Specific - Unambiguous, consistent, and at the appropriate level of detail

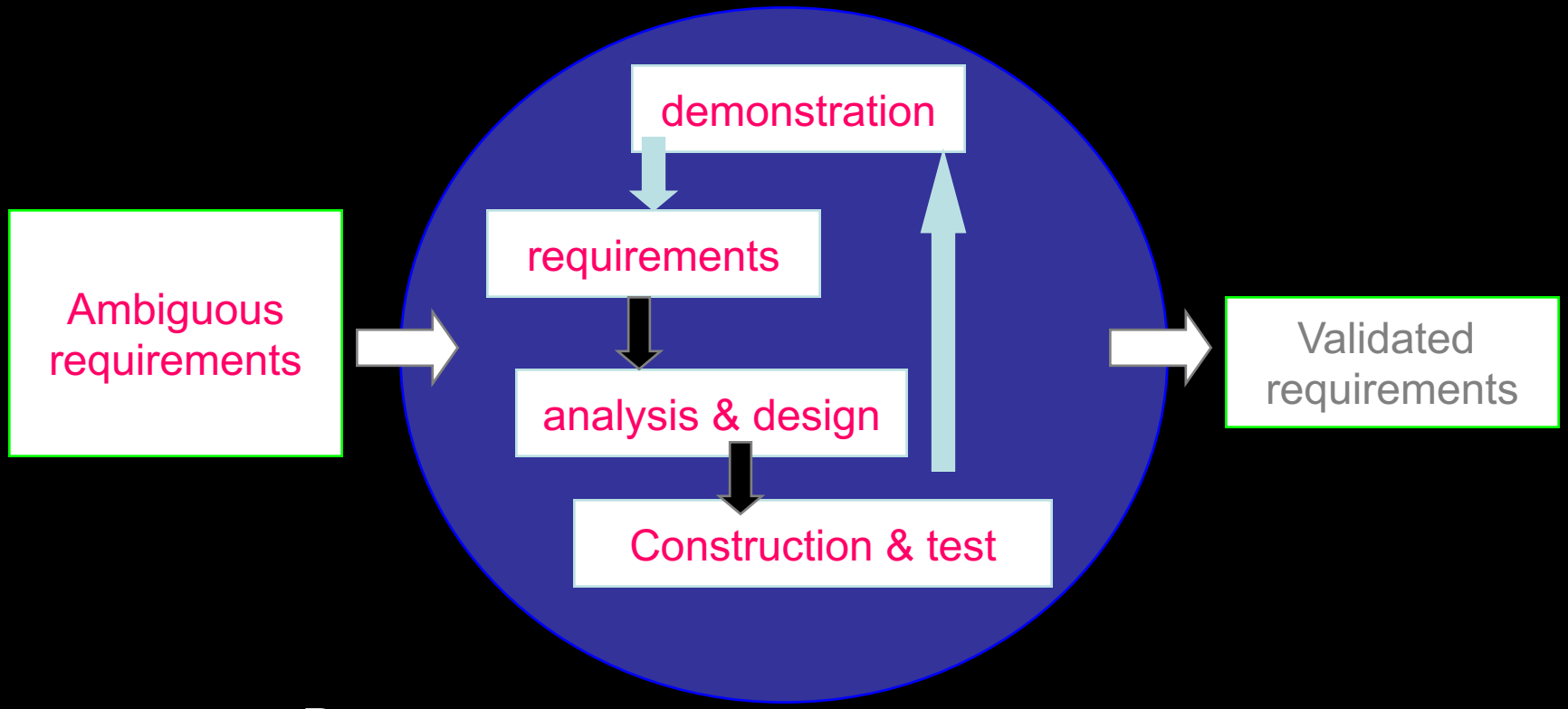
Measurable - Possible to verify that the job is done

Attainable - Technically feasible and viable from business perspective

Realizable - Realistic given all the constraints (e.g., budget, resources, time, infrastructure)

Traceable - Linked from conception through specification, design, implementation, and test

Advanced Prototyping significantly improves software quality, integrity, safety and resilience



Purposes:

- Validation of ambiguous business requirements
- Proof of concept for technical feasibility assessment
- Mitigation of risks early in the development cycle

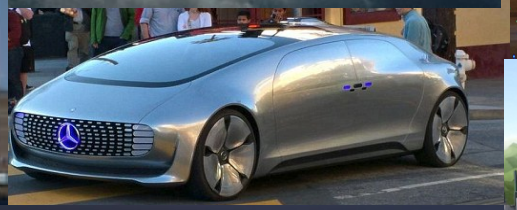
Typical Test Methodology and Cycle

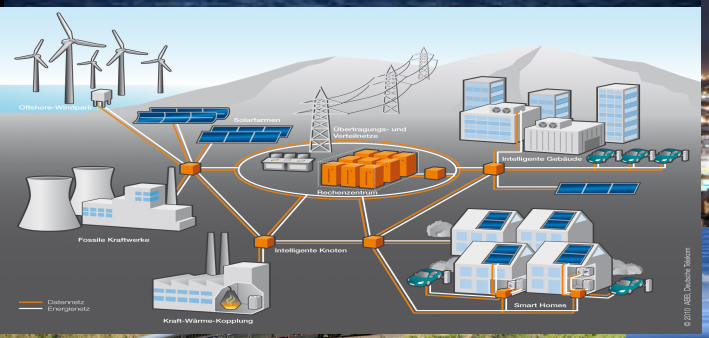
- Component Development and Unit Tests
- Functional Verification Test
- System Integration Test
- Negative Test (business exceptions, environmental errors)
- E2E System Test
  - Various subsystems are combined
  - Tests the entire system as a complete entity
- Nonfunctional Test (availability, performance, scalability, recovery)
- E2E Performance Test and Tuning
- User acceptance testing
  - Independent testing performed by trained end users
  - Ensures that the system operates as they expect
- Regression Test
- Preproduction Test

Integration Testing Approach (common mistake)

- Black-Box testing
  - Testers have very limited or no knowledge of design or code
  - Testers blindly run the test cases and observe behavior, often out of context
- White-Box testing
  - Testers have intimate knowledge of the internal systems (expected behaviors, control flows, data flows)
  - Scope of testing include all possible scenarios and use cases
  - Testers focus on handling exceptions & environmental errors (negative testing), as well as happy paths

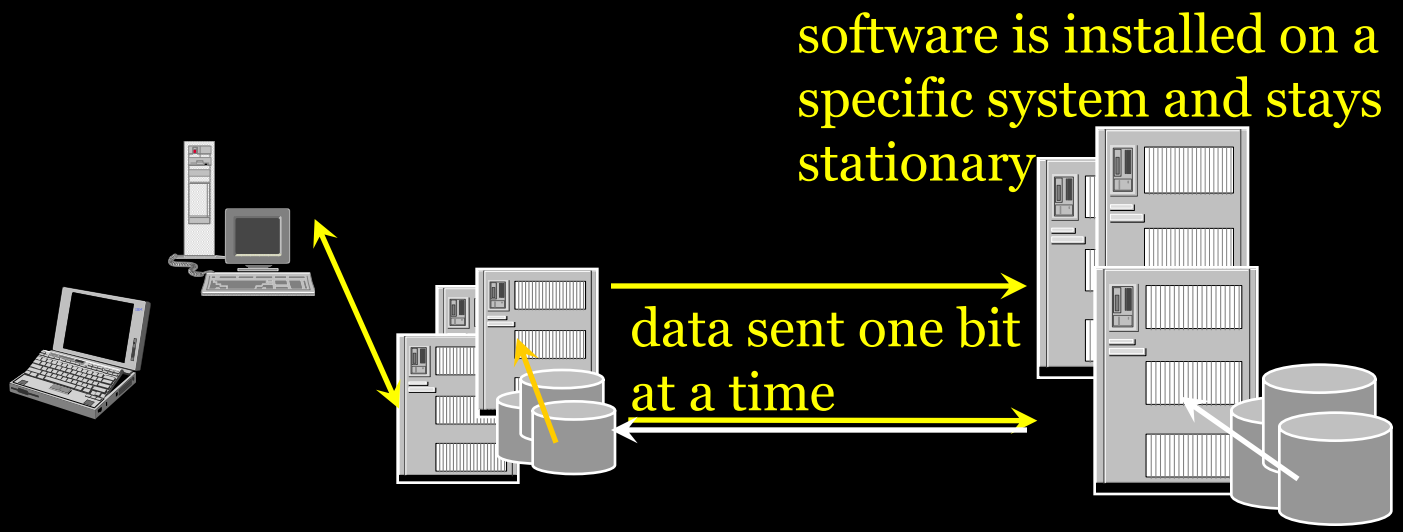
Cyber Physical System and the Internet of Things require new approach & method for software engineering





Traditional computing model is inefficient and inadequate

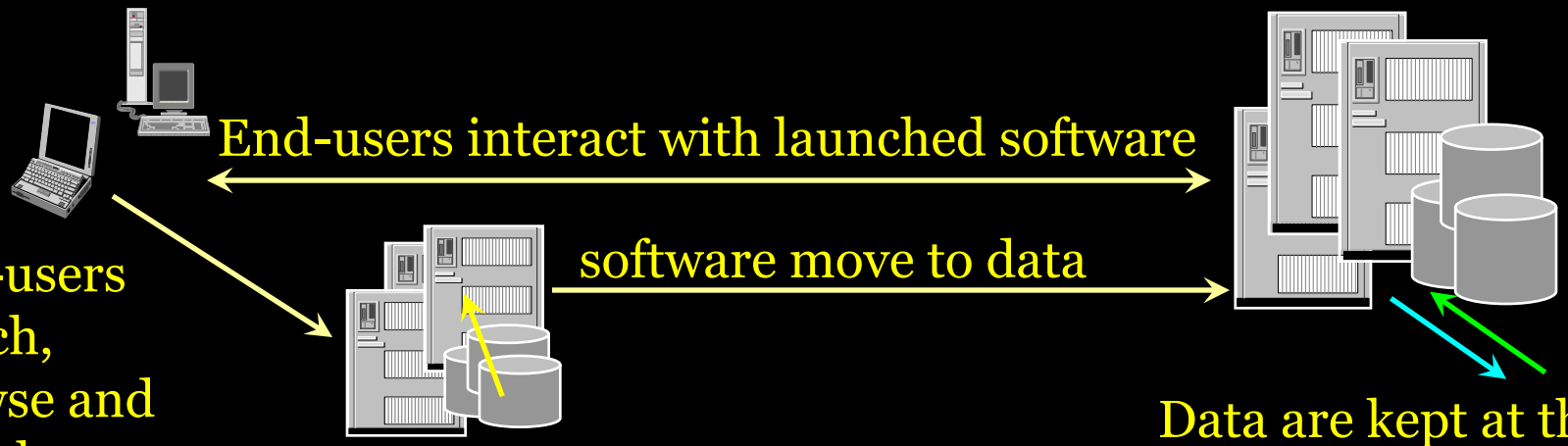
Software is static and stationary. Data moves to software.



Issues:
performance, security, integrity, privacy, IP, TCO, operation, management

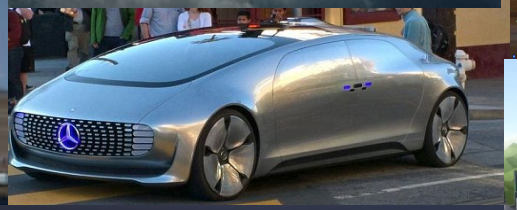
Leverage the patent-protected Data-Centric Computing Model based on Itinerant Mobile Agent Software to address the challenges

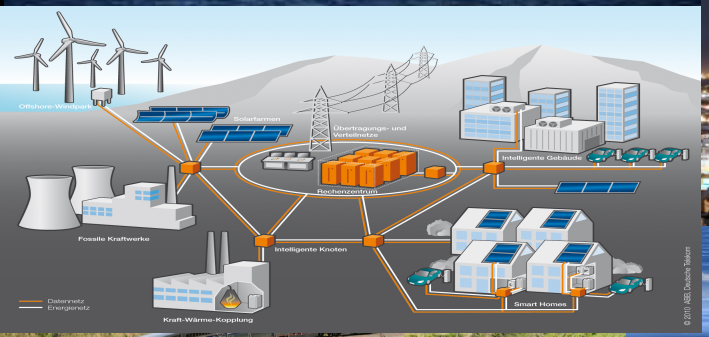
Software migrates to Data.



How can we assure the authenticity and integrity of the software?

Machine-to-machine interactions based on imbedded software are pervasive. How safe are they?





Application Software examples for Industrial Automation

- Air traffic control system
- Airplanes
- Fighter jets
- Train operation and railway signal control system
- Nuclear power plant operation
- Military drones
- Self-driving automobiles
- Operating rooms for brain surgery
- ICU monitoring system
- Defense system
- IEDs
- Space exploration

Software Safety for Industrial Automation

- Malfunction or disruption of industrial automation software leads to much greater consequences, sometimes to a disaster.
- Embedded software for real-time measurements and controls in IoT/CPS systems require rigid standards and regulations for system safety and resilience.
- Software safety goes hand in hand with availability, performance, reliability, and security.

What if a software component needs to be restarted or the server needs to be rebooted, while an airplane is taking off, self-driving car is skidding on an icy road, a drone is executing a military command?

How can we prevent unauthorized access to the imbedded software and taking over the control (of airplanes, armed drones, trains)?

Software Security

Entity Identification and Validation for Authentication

- For human users
  - carbon-based chemical entity
- For servers and devices
  - Silicon-based electronic entity
  - Only for hardware devices and network connections

Entity = CBCE & SBEE

- Cybersecurity threats
  - Insiders
  - Malicious software
    - Antivirus is an after-the-fact reactive, ineffective measures

Cyber Security

- Outsider attacks
  - Perimeter defense (physical and logical)
  - Strong authentication
  - Limited access
  - Untrusted
- Insider threats
  - No perimeter defense
  - Privileged access
  - Trusted

- Software identity and credentials for authenticity

Cyber Security – Common Threats

- Viruses - Infect computers through email attachments or file sharing
- Spyware - Piggybacks on programs you download, gathers information about your behavior, and collects personal information
- Hackers - Hack into your computer and temporarily take control of your computer
- Identity Thieves - Obtain unauthorized access to your personal information and then use the information to claim to be you

Throttling for Unusual Behavior or Abnormal Volume of Data for IoT/CPS

- Throttle and queue for deferred processing of events and associated data, upon detection of:

- Unusual pattern of events with high velocity
- Burst volume of data in an abnormal rate

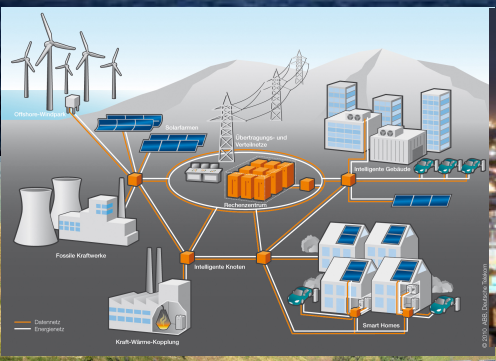
Shifting the focus of Cyber Security

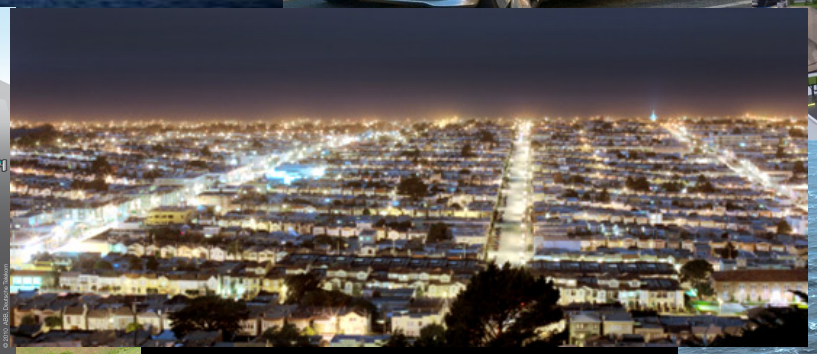
- ICT Security, Cyber Security, has been focused so far on Data Protection.
- Traditional approach is to protect hardware (servers and storage) and to secure the network transport.
- Software-based (malicious software) security breaches lead to much more significant damage and yet are very hard to detect.
- Antivirus software is a reactive after-the-fact remedy to feel good.

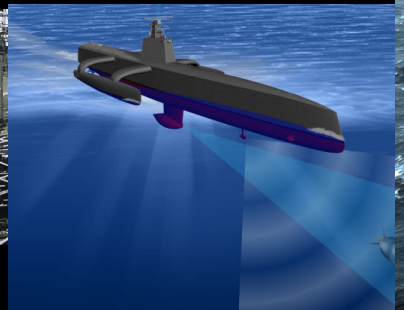
We need a novel method and system to ensure authenticity and integrity of software.

Systematic authentication of software at the component/module level before running.

Software-based Security Threats can lead to inconceivable consequences to human civilization







It is time to Authenticate and Authorize Software at the Component Level

- Software Component identifier
- OSF/DCE UUID for registration and validation of software components via Kerberos 3rd-Party authentication
- UUID - Universally Unique Identifier, IETF RFC 4122
  - 128-bit opaque structure, 2 ** 128 = 3.4E38
  - MAC address + time in 100 nanoseconds + network domain name + distinguished name
  - Spatiotemporally unique
  - Assured uniqueness, but not guaranteed uniqueness

Global Software Component Identifier

- Identify the source/supplier of a software component and validate its authenticity
- Software vendor registration
- Safeguard and protect software vendor registry (vault)
- International statues for safeguard and protect software vendor identifiers
- Extension of OS kernels
  - Authenticate and authorize software components
    - 1024-bit opaque structure
      - ~2.7E154 components for von Neumann
      - E.g. Registered software vendor identifier + software component identifier + validity time period + digital signature
  - Validate authenticity and integrity via federated PKI security realms for the Internet
  - Pre-run to assess the behavior in a protected sandbox

Performance, Scalability and Resilience of Software are directly relevant to Safety

- Systems are most vulnerable during recovery (reboot).
- That is the time when the protection shield is down and hackers exploit the systems..

Software Performance, Availability, Scalability and Resilience

Novel Programming Model for Higher Performance and Scalability

Traditional Model

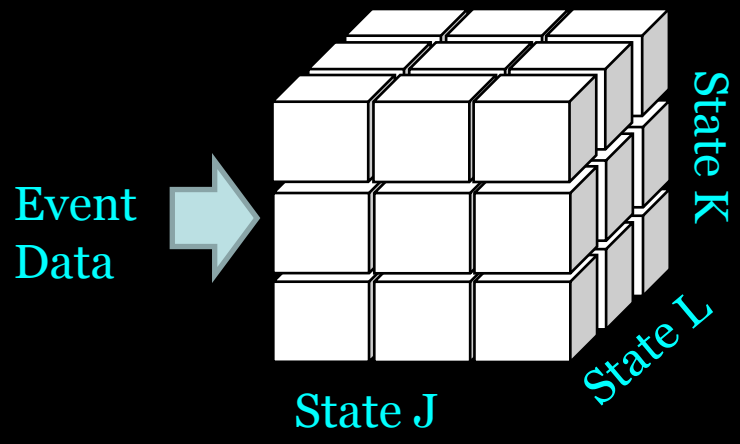
```
IF (Y or Z) & (A&B&C or D&E)
    IF ...
        IF ...
            IF ...
                IF ...
                    IF
                        do 1, do 2,  ...
                    ELSEIF ....
                            case

ELSEIF K & ...
```

Took 8 minutes

Multidimensional FSM

F(x) = mdFSM (e1,e2,... s1,s2,... t+n)
Perform F(x);



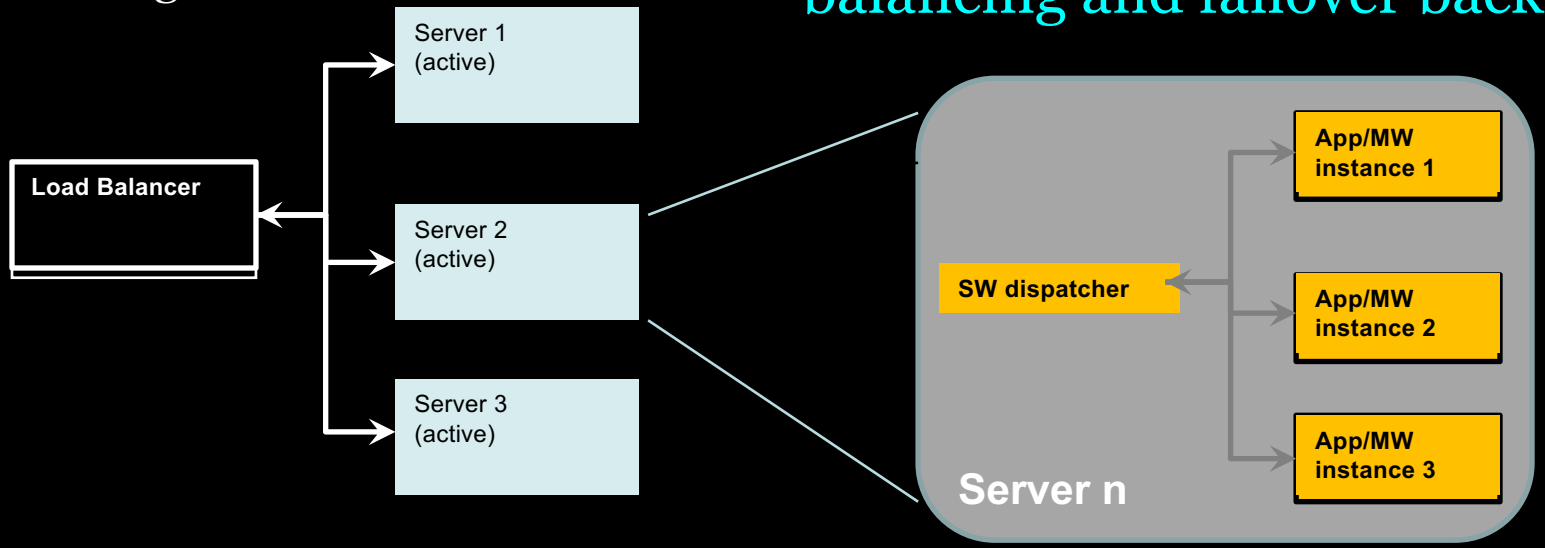
Took 11 milliseconds

Novel Approach for Availability and Performance

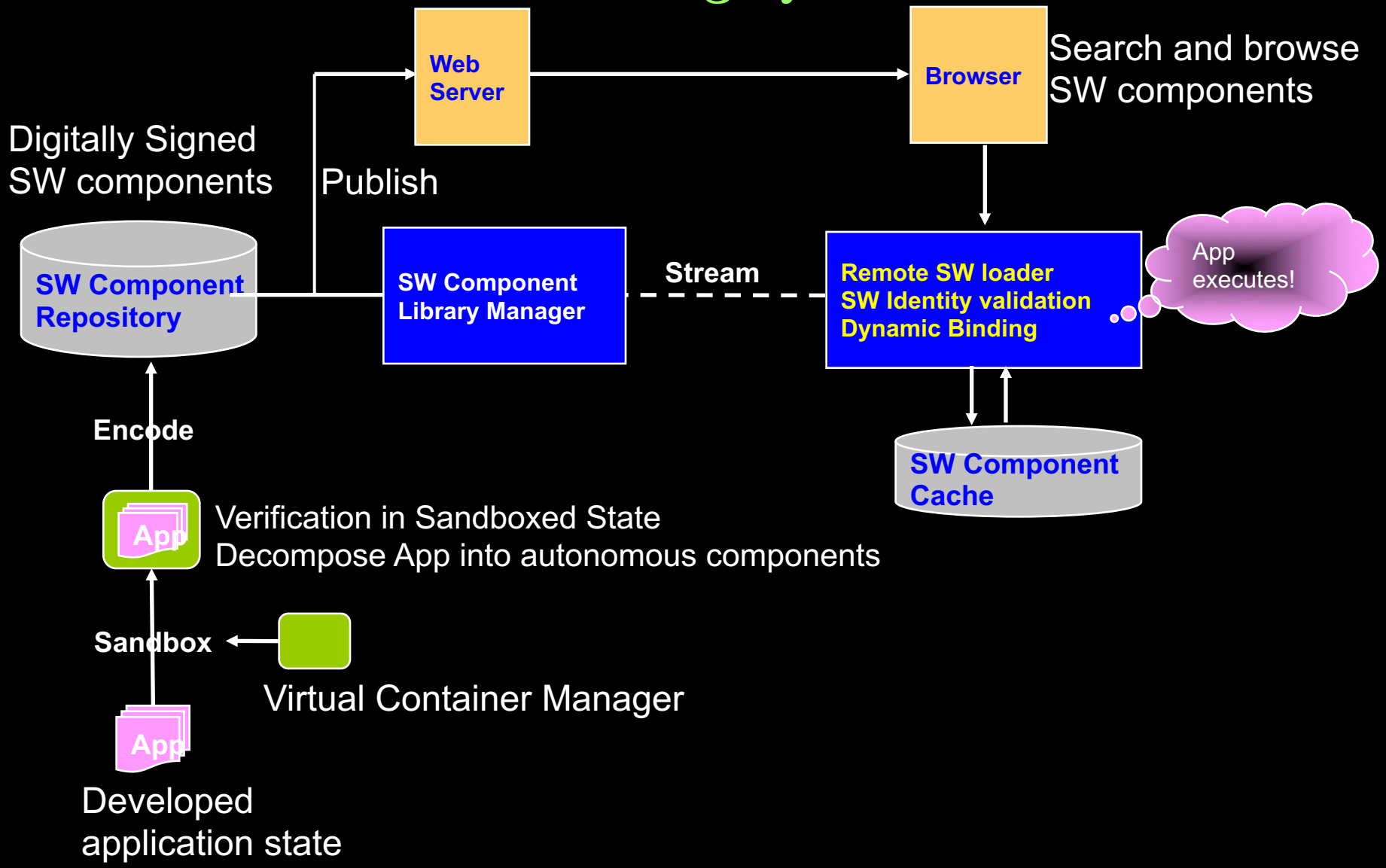
2/3 of system disruptions are caused by software problems, especially due to software deficiency in handling exceptions and environmental errors

Industry practice of hardware clustering

Augmentation via software cloning and clustering for load balancing and failover backup



Novel Approach for Dynamic Software Provisioning System



Flexibility and Adaptability via Dynamic Component Binding

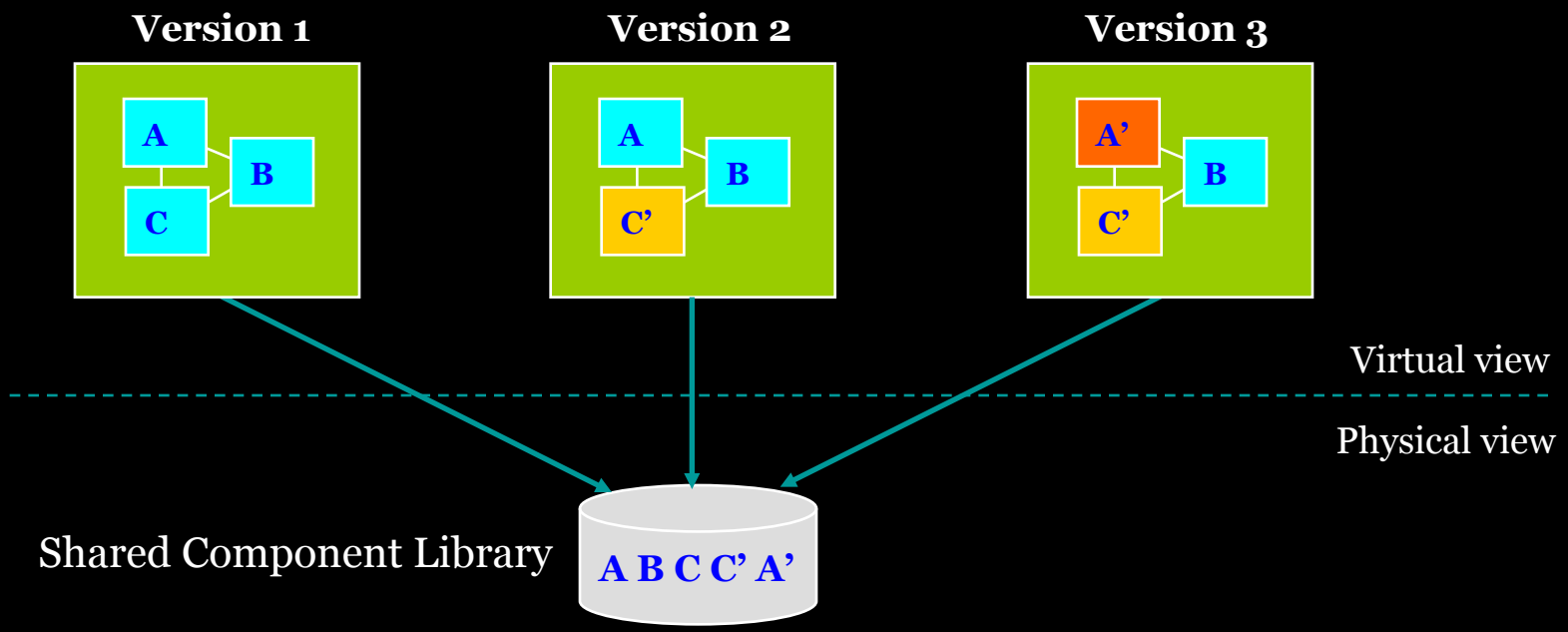
- Components, subcomponents, data adapters are registered in a shared library
- Components, subcomponents, data adapters are digitally signed for identity, authenticity and integrity

- Components, subcomponents, data adapters are dynamically loaded as required
- Inter-component bindings dynamically on demand (c.f., Ada rendezvous)

Novel Method for Dynamic Software Version Control & Migration

Software Versions" are only meaningful in virtual space

- Software component library is just a collection of autonomous software components (guaranteed to be duplicate free).
- Enables rapid roll-forward & roll-back between versions.



IDC Projections for 2020

- 212 billion installed Things
- 30 billion autonomously connected Things
- 3 zettabytes of embedded system data
- $8.9 trillion of business values

대단히 감사합니다
Thank you

One who successfully harnesses the Big Data and draws Insights and Foresights thru Cognitive Analytics wins the Future, in this Information Age.

OK Baek
1-416-729-5610
okbaek@ca.ibm.com

Bibliography (1/2)

- IBM Research, Global Technology Outlook
- IBM Research, Accelerated discovery of insights with Big Data
- www.google.com - Graphics, images, photos
- www.youtube.com - images, video clips
- www.MIT.edu
- www.harvard.edu
- www.the-scientist.com
- www.research.ibm.com
- www.ibm.com
- www.redbooks.ibm.com
- www.wikipedia.com
- Baek, OK and Haas, LM. Data-Centric eScience Solution Framework and Architecture, IBM TJ Watson Research, 2007
- Baek, OK. Privacy Solution Framework for eGovernment Solutions, 2001
- Baek, OK. Distributed security architecture for e-business solutions, 1999
- Baek, OK. Distributed Security and Privacy Framework for HIPAA Compliance, 2002
- Baek, OK. Distributed enterprise solution architecture (DTP, ORB, DCE, MQ, PKI), 1994

Bibliography (2/2)

- Baek, OK. Safety and Resilience in Advanced Distributed System for Smart Grid, 2003
- Baek, OK & Robson, Barry. The Engines of Hippocrates: From the Dawn of Medicine to Medical and Pharmaceutical Informatics, John Wiley & Sons, Inc. ISBN-10: 0-470-28953-8; ISBN-13: 978-0-470-28953-2, John Wiley & Sons, 2009
- Baek, O.K., S. Elkins, M.A.Z. Hupcey and A.J. Williams. 2011. "Collaborative Computational Technologies for Biomedical Research". John Wiley & Sons, Inc. ISBN 978-0-470-63803-3. 2011 (18:281-300)
- Baek, OK. Architectures for secure acquisition and use of patient data. The Fourth Bioinformatics Industrialization Workshop, University of California, 2003.
- Baek, OK. Bioinformatics and Information-Based Medicine: From Pharmacogenomics to Personalized Healthcare. IBM Academy topical conference, IBM TJ Watson Research, 2003
- Baek, OK. Medical Informatics and Integrated Medical Records for Personalized Medicine, European Union Electronic Health Record Conference (Toward Electronic Health Record Europe) Proceedings 2002
- Baek, OK. Leverage of IT for Optimal Healthcare, CIHR (Canadian Innovation for Health Research) Scientific Directors' Conference Proceedings 2002
- Baek, OK. Asynchronous Message-Driven Processing Model based on Distributed Objects, ISO/IEC JTC1/SC21 Open Distributed Processing 1992