Evolution of SE Technologies:

Looking Back and Looking Forward

강 교 철

명예교수

July 13, 2022

Pohang University of Science and Technology (POSTECH)

Agenda

- **Prologue**
- Technology Evolution: Looking Back
- Driving Forces of Technology Evolution
- Technology Evolution: Looking Forward
- Epilogue

Prologue

“There are no shortcuts in evolution”

Louis D. Brandeis

“Creativity involves breaking out of established patterns in order to look at things in a different way.”

Edward de Bono

About me

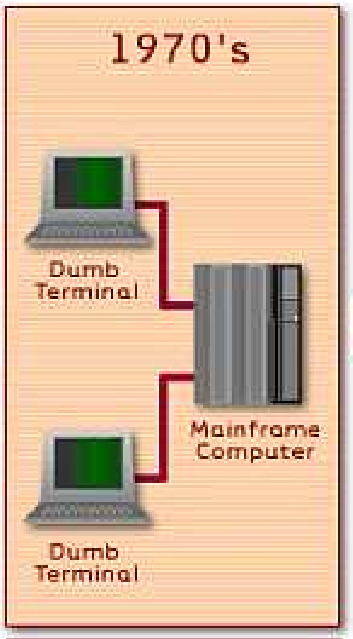
- At the University of Michigan (ISDOS Project)
  - Developed requirements engineering and meta modeling techniques and tools (PSL/PSA, Meta System)
- At Bellcore and Bell Labs
  - Experienced software reuse issues in industry
- At SEI
  - Explored reuse issues
  - Developed FODA, a commonality/variability analysis method
- At POSTECH
  - Experienced SE problems in industry
  - Extended FODA and created a successor (FORM)
  - Developing a CASE environment (ASADAL, VULCAN)
- At Samsung Electronics
  - Application of SE techniques to HW engineering
  - Reengineering of SW products

Agenda

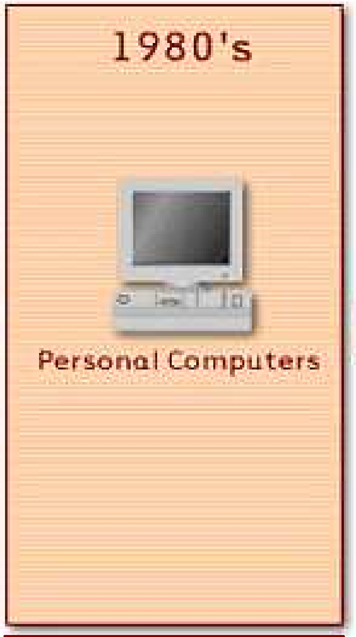
- Prologue
- **Technology Evolution: Looking Back**
- Driving Forces of Technology Evolution
- Technology Evolution: Looking Forward
- Epilogue

Evolution of Computing Technologies

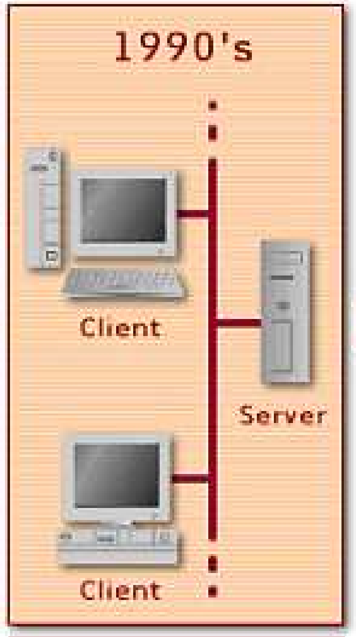
Stand-alone → **Interoperate**



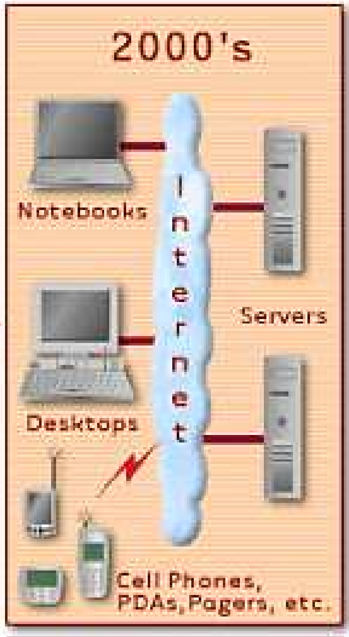
Mainframe Computing



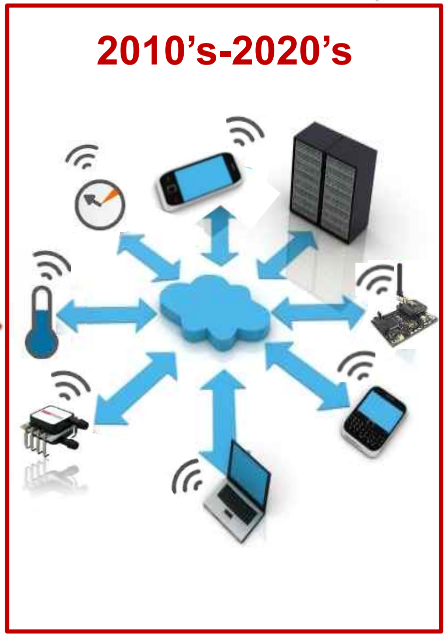
Desktop Computing



Wired Network



Wired + Wireless Network



Open Environment (Fuzzy Boundary)

Performance **Maintainability** **Complexity** **Security** **Self-Adaptiveness** **Feature Interactions**

Top-down decomposition → Bottom-up integration

Images from: 1) "Information Model: The Key to Integration," AutomatedBuildings.com Article, Jan. 2002.; 2) Judy Quinn, "Why switch from Desktop to Cloud Project Management Software?," Project Management Articles, Apr. 8, 2012.

Looking Back: 60's and 70's

- Programming Language
  - Problems: Coding
  - Technical developments:
    - Abstraction of hardware machine
    - Creation of logical machines and languages
    - Development of compilers for "specialization"
    - Attempted to raise the level of abstraction
      - Creation of 1st, 2nd, and 3rd generation languages

Looking Back: 70's and 80's

- Methodology and Code Generator
  - Problems: Problem complexity; analysis and design
  - Technical developments: Methodologies and support tools
    - Requirements models and languages
    - Attempts were made to generate code from requirement specifications, but failed; Why?

Toward Automating the Software Development Cycle:

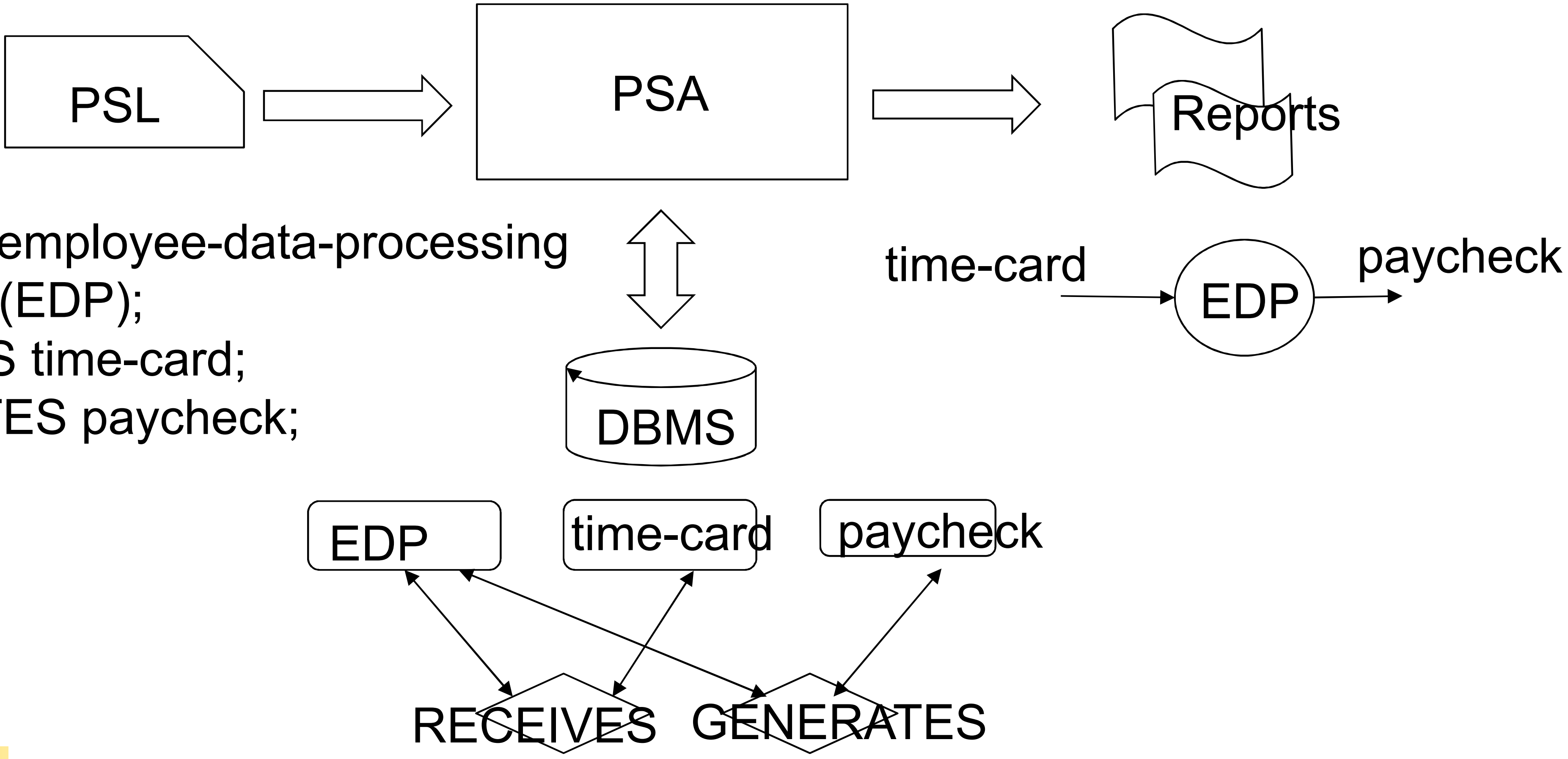
“A team of programmers or users would focus their efforts on developing specification of a user’s requirement. Then an intelligent system would write the program in much the same way that today’s compilers generate machine code from high-level languages.”

Karen A. Frenkel, 1985

-Psi Project at Stanford University
-Chi Project (Knowledge-Based Software Assistant) at Kestrel Institute
-Programmer’s Apprentice Project at MIT

- My experience at the ISDOS Project (Mid 70's- Early 80's)
  - Led by Late Prof. Daniel Teichroew
  - Separation of representation (views) from the model with increasing number of analysis needs
  - Separation of system "aspects" with widening application areas
  - Meta modeling (to handle evolution the language and tools)
  - CASE tools (PSL-Problem Statement Language/PSA-Problem Statement Analyzer, Meta Generators, Report Specification Language/Report Generator, etc.)

- Separation of representations/views from the model



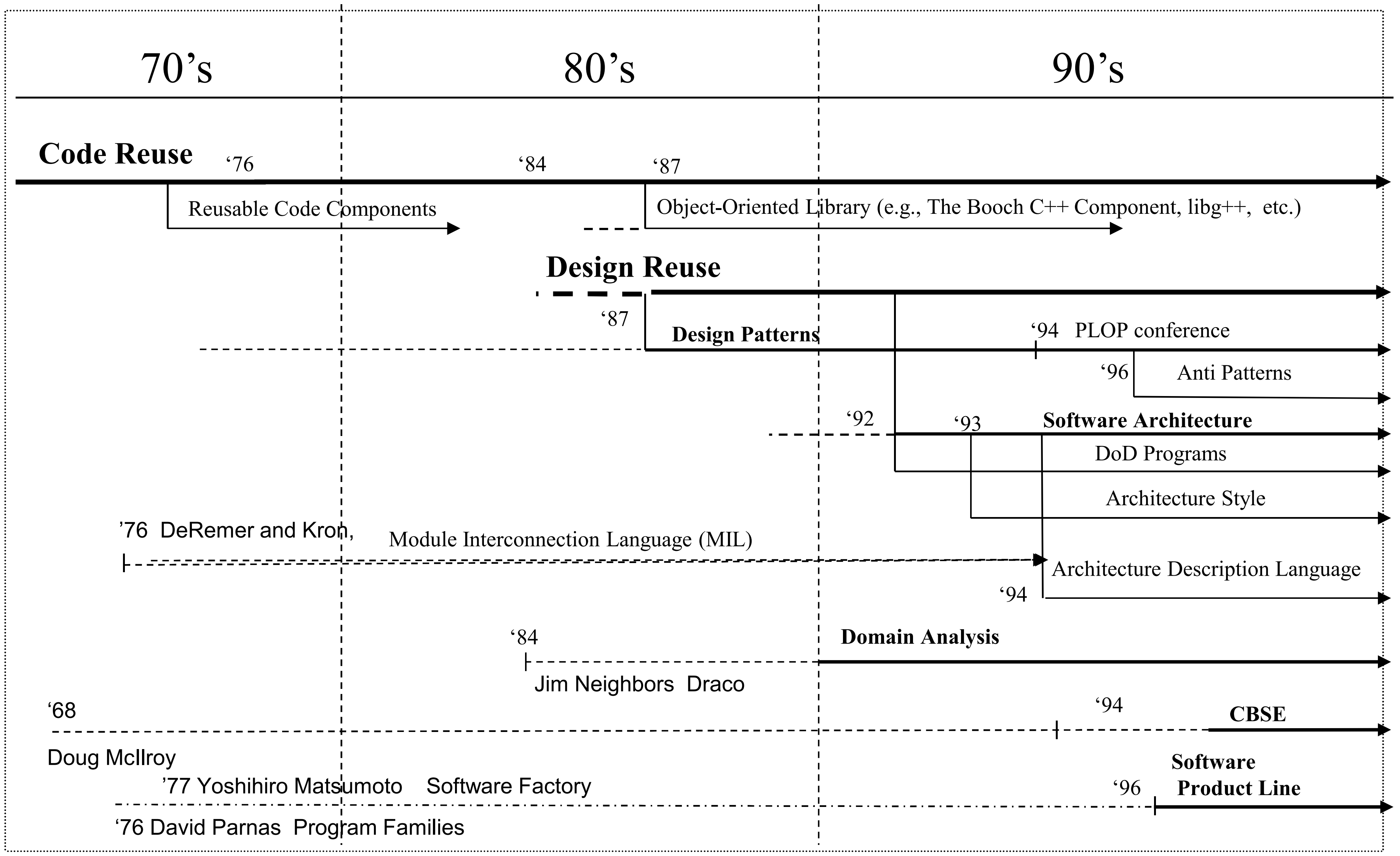
DEF PROC employee-data-processing (EDP);
RECEIVES time-card;
GENERATES paycheck;

- Separation of “system aspects” with diverse application areas
  - From batch processing systems to real-time reactive systems
  - Creation of a meta modeling technique and support tools such as a report specification language (much like XML)
  - “Generalization” in modelling and support tool development (much like Eclipse)

Looking Back: 80's thru 90's

- Reuse and High-Productivity Techniques
  - Problems: productivity and quality; Resource sharing among projects
  - Technical developments:
    - Software libraries
    - Design patterns
    - Architecture styles
    - Stepwise refinement/transformation
    - Model-driven software development

Evolution of Reuse Concepts



Code Reuse

- Practices:
  - Generic library reuse
  - Copy and paste of application level code

- Issues:
  - Structural mismatch between applications was a major impediment to reuse.

<u>Reuse of design must occur prior to code reuse.</u>

Design Reuse

- Technologies:
  - Design Patterns.[1]
    - Design should be specific to the problem at hand but also general enough to address future problems and requirements.
  - Anti Patterns.[2]
    - It is useful to show the presence of certain patterns (AntiPatterns) in unsuccessful systems.
  - PLoP (Pattern Languages of Programs) conferences.[3]
    - Authors have the opportunity to refine and extend their patterns with help from knowledgeable and sympathetic fellow pattern enthusiasts.

[1] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley, 25th Printing October 2002, pages:1-31.

[2] William J. Brown, Raphael C. Malveau, Hays W. McCormick III and Thomas J. Mowbray, "AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis," John Wiley & Sons, Inc., 1998, pages:3-14.

[3] "PLoP 2004 Conference," http://hillside.net/plop/2004/.

Design Reuse

- Practices:
  - Low level design patterns
  - Copy other design

- Issues:
  - A design made for a specific application context had low reusability.

⇨ ***<u>A common design framework (called architecture) must be established to promote reuse of design.</u>***

Software Architecture

- Technologies:
  - Architectural Styles.[4]
    - Some architectural styles are broadly-used.
    - They provide a reuser with the concentrated wisdom of many preceding designers faced with similar problems.
  - Architecture Description Languages (ADLs).[5]
    - To support architecture-based development, formal modeling notations and analysis and development tools that operate on architectural specifications are needed.

[4] David Garlan and Mary Shaw, “An Introduction to Software Architecture,” tech. report CMU-CS-94-166, ESC-TR-94-21, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, January 1994.

[5] Nenad Medvidovic and Richard N. Taylor, “A Classification and Comparison Framework for Software Architecture Description Languages,” Technical report UCI-ICS-97-02, University of California at Irvine, 18 June 1997.

Software Architecture

- Practices:
  - Discovery of architecture style
  - Architecture description language
- Issues:
  - It is difficult to define generic architectures that will promote design reuse.

➡ ***<u>Domain specific architectures must be established.</u>***

Domain Analysis

- Technologies:
  - Feature-Oriented Domain Analysis (FODA).[6]
    - FODA is a method for discovering and representing commonalities among related software systems.
  - Domain Specific Software Architecture (DSSA).[7]
    - A DSSA can be partly seen as a reference architecture, a meta architecture describing a known solution for a particular domain.

[6] K. Kang et al., “Feature Oriented Domain Analysis (FODA) Feasibility Study,” tech. report CMU/SEI-90-TR-21, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1990.

[7] Hans de Bruin, “Domain-Specific Software Architecture,” http://www.cs.vu.nl/~hansdb/state/node14.html.

Domain Analysis

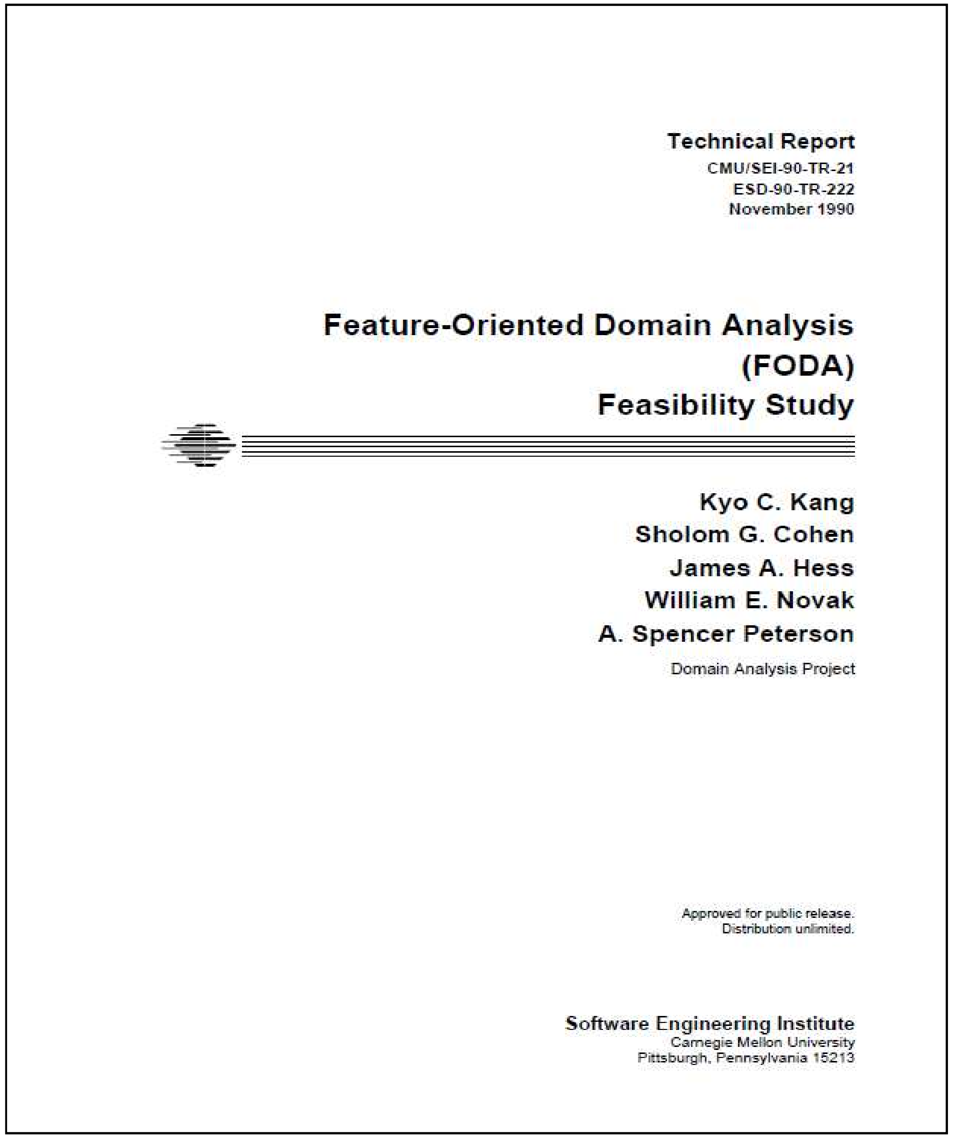
- Practices:
  - Methods for analyzing a domain and developing architectures
- Issues:
  - Establishing / scoping the boundary of a domain was difficult.
  - Up-front investment is high.

- ***Business context specific scoping is necessary.***
- ***Various process models are necessary.***

Looking Back: 1990's thru 2000's

- Strategic Reuse Complying with Market Needs; New Computing Environments (Ubiquitous Mobile Computing, Cloud Computing, Service-Orientation)
  - Problems: Competition to reduce "time-to-market"; Mobile and pervasive computing
  - Technical developments:
    - Product Line Engineering: Strategic software reuse with the focus on targeting market segments.
    - Ubiquitous Computing: Pervasive, mobile, context-aware, and self-healing systems; reflective middleware.
    - Context-free Computing: Open API
    - SOA

Feature-Oriented Domain Analysis (FODA)



Technical Report
CMU/SEI-90-TR-21
ESD-90-TR-222
November 1990

Feature-Oriented Domain Analysis (FODA) Feasibility Study

Kyo C. Kang
Sholom G. Cohen
James A. Hess
William E. Novak
A. Spencer Peterson

Domain Analysis Project

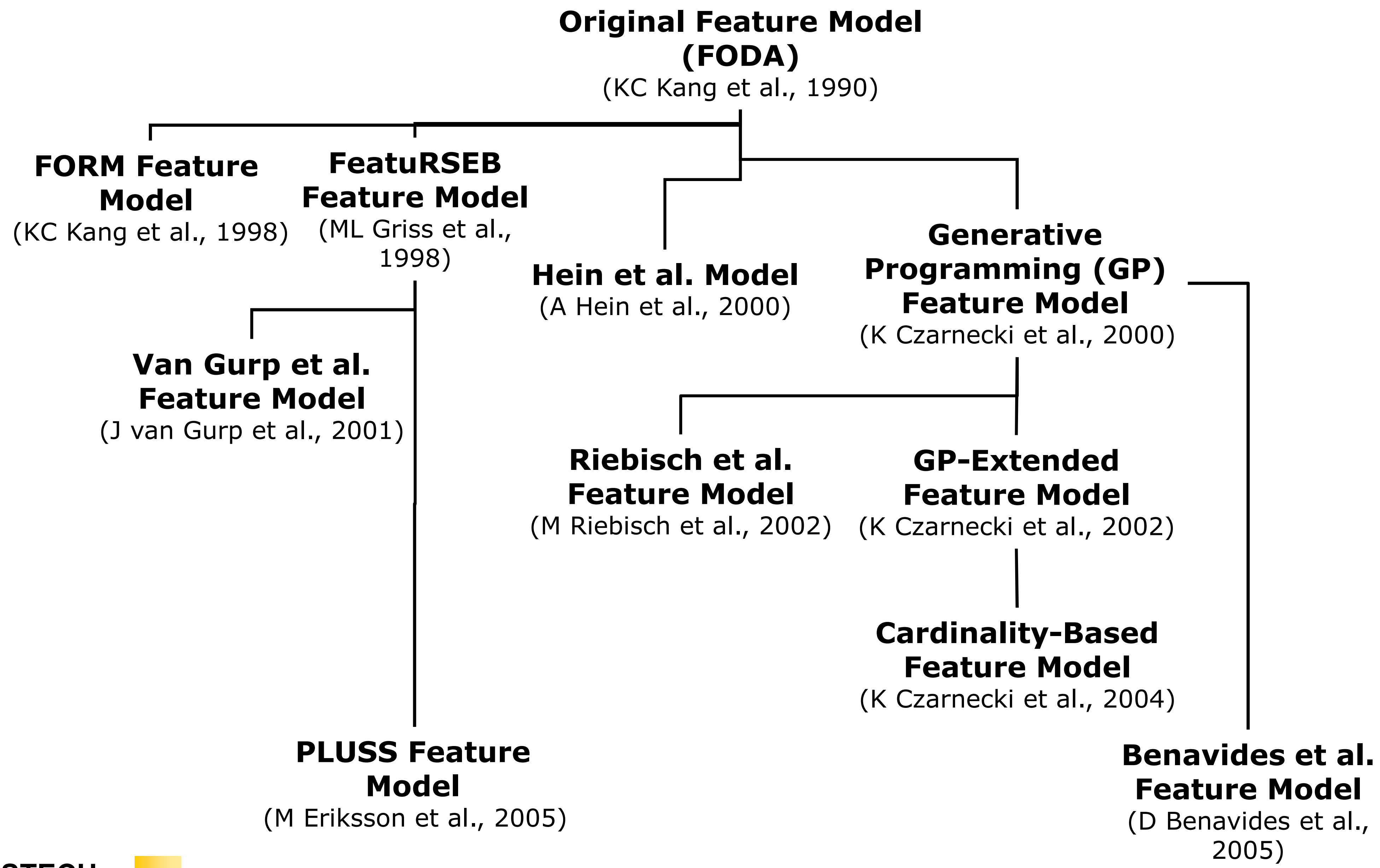
Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

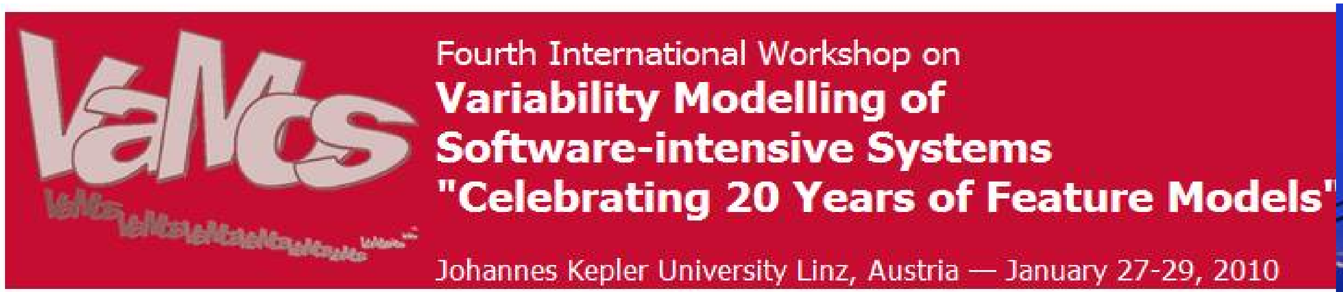
As of June, 2022

Feature-oriented domain analysis (FODA) feasibility studyKC Kang, SG Cohen, J A Hess, WE Novak, AS Peterson Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst	5538	1990
FORM: A feature-; oriented reuse method with domain-; specific reference architecturesKC Kang, S Kim, J Lee, K Kim, E Shin, M Huh Annals of Software Engineering 5 (1), 143	1610	1998
Software reuse research: Status and futureWB Frakes, K Kang IEEE transactions on Software Engineering 31 (7), 529-536	1017	2005
Feature-oriented product line engineeringKC Kang, J Lee, P Donohoe IEEE software 19 (4), 58-65	837	2002
Concepts and guidelines of feature modeling for product line software engineering K Lee, KC Kang, J Lee International Conference on Software Reuse, 62-77	549	2002
Issues in requirements elicitationMG Christel, KC Kang Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst	505	1992
Feature-oriented domain analysis feasibility studyKC Kang, SG Cohen, JA Hess, WE Novak, AS Peterson SEI Technical Report CMU/SEI-90-TR-21	265	1990
Feature-oriented domain analysis (FODA) feasibility study. Software Engineering InstituteK Kang, S Cohen, J Hess, W Novak, AS Peterson Universitas Carnegie Mellon, Pittsburgh, Pennsylvania	252	1990

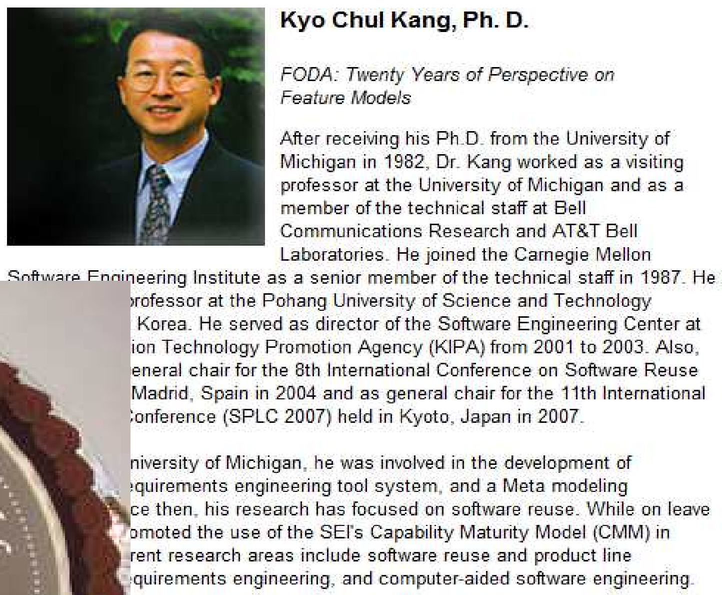
Feature Model Genealogy


Original Feature Model (FODA)
(KC Kang et al., 1990)
FORM Feature Model
(KC Kang et al., 1998)
FeatuRSEB Feature Model
(ML Griss et al., 1998)
Hein et al. Model
(A Hein et al., 2000)
Generative Programming (GP) Feature Model
(K Czarnecki et al., 2000)
Van Gurp et al. Feature Model
(J van Gurp et al., 2001)
Riebisch et al. Feature Model
(M Riebisch et al., 2002)
GP-Extended Feature Model
(K Czarnecki et al., 2002)
Cardinality-Based Feature Model
(K Czarnecki et al., 2004)
PLUSS Feature Model
(M Eriksson et al., 2005)
Benavides et al. Feature Model
(D Benavides et al., 2005)

Product Line Activities


Fourth International Workshop on
Variability Modelling of Software-intensive Systems "Celebrating 20 Years of Feature Models"
Johannes Kepler University Linz, Austria — January 27-29, 2010

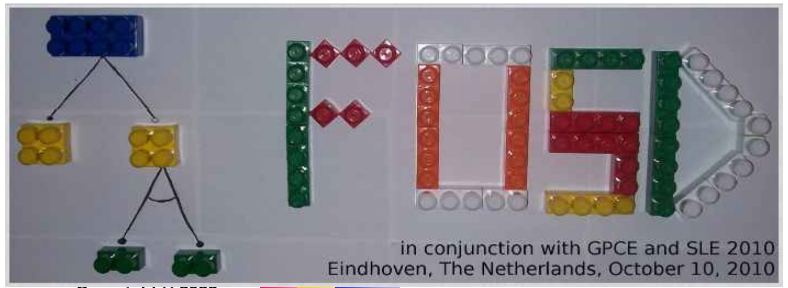
13th International Software Product Line Conference (SPLC)
August 24–28, 2009 | Airport Marriott, San Francisco, CA, USA


Kyo Chul Kang, Ph. D.

FODA: Twenty Years of Perspective on Feature Models

After receiving his Ph.D. from the University of Michigan in 1982, Dr. Kang worked as a visiting professor at the University of Michigan and as a member of the technical staff at Bell Communications Research and AT&T Bell Laboratories. He joined the Carnegie Mellon Software Engineering Institute as a senior member of the technical staff in 1987. He
rofessor at the Pohang University of Science and Technology
Korea. He served as director of the Software Engineering Center at
ion Technology Promotion Agency (KIPA) from 2001 to 2003. Also,
eneral chair for the 8th International Conference on Software Reuse
Madrid, Spain in 2004 and as general chair for the 11th International
onference (SPLC 2007) held in Kyoto, Japan in 2007.

niversity of Michigan, he was involved in the development of
equirements engineering tool system, and a Meta modeling
ce then, his research has focused on software reuse. While on leave
omoted the use of the SEI's Capability Maturity Model (CMM) in
rent research areas include software reuse and product line
quirements engineering, and computer-aided software engineering.



in conjunction with GPCE and SLE 2010
Eindhoven, The Netherlands, October 10, 2010

- Domain Oriented Reuse and Software Architecture
  - Problems: Diverse market needs, Short time-to-market
  - Technical developments:
    - Systematic software reuse
    - Domain Analysis Technology: FODA, Faceted Classification, KAPTUR, FAST
    - Software Architecture: Architecture Style, Architecture Description Language
    - DSSA (Domain Specific Software Architecture)
    - Generative programming/Domain-specific application generator

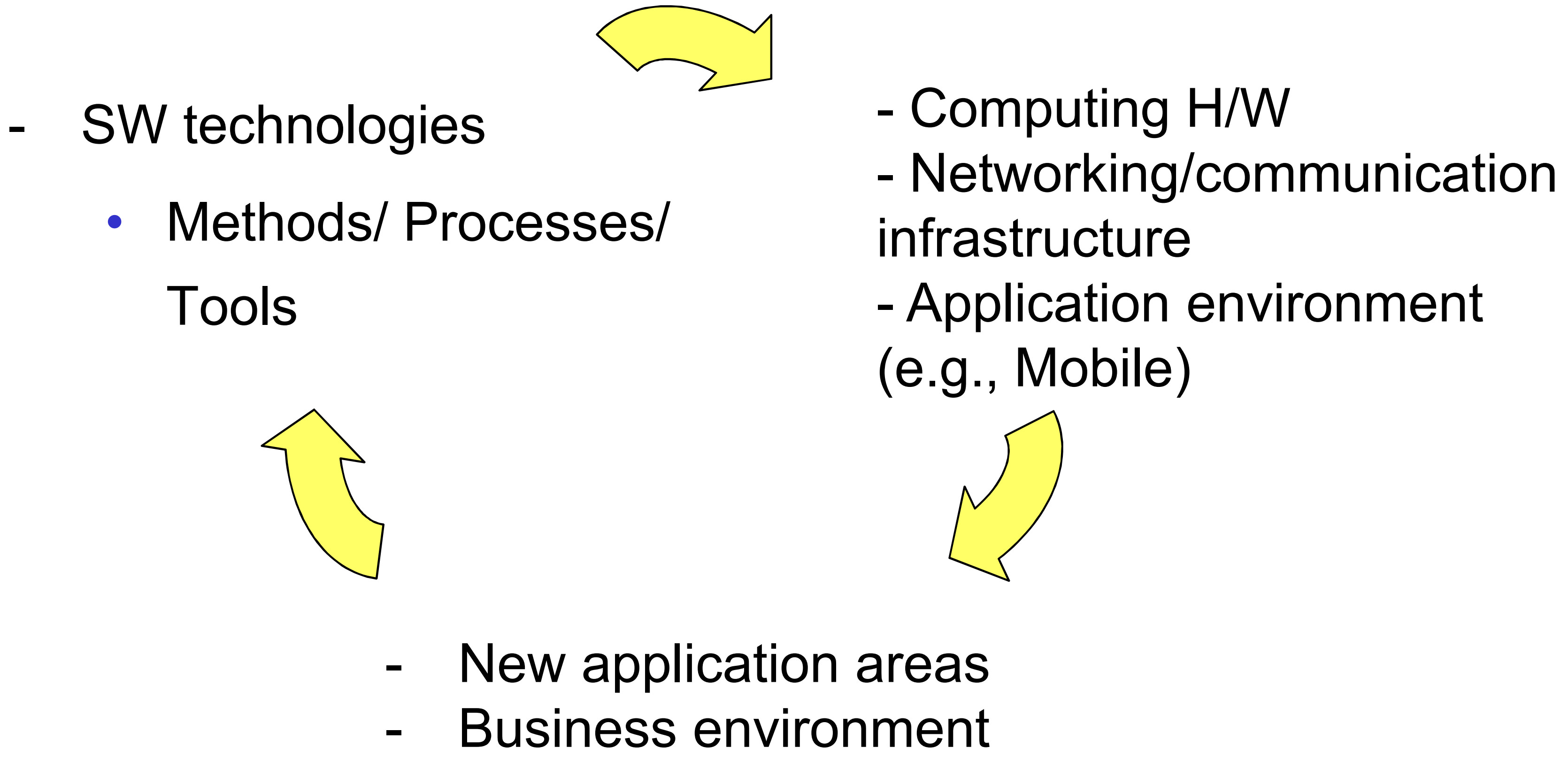
Agenda

- Prologue
- Technology Evolution: Looking Back
- **Driving Forces of Technology Evolution**
- Technology Evolution: Looking Forward
- Epilogue

Driving Forces of Technology Evolution/Development

- **Expansion of application domains**
  - **Engineering, business, u-health, …**
- **Evolution of computing platform/environments**
  - **Batch, time-sharing, distributed, ubiquitous …**
- **Evolution of development environments**
  - **In-house, SI, outsourcing, open source, …**
- **Changes in business environments**
  - **Customer-centric, platform-centric (application platform), …**
- **Development of new technologies**
  - **Compiler, formal verification, machine learning, big data …**

Ecosystem of Technology Evolution



- SW technologies
  - Methods/ Processes/ Tools

- Computing H/W
- Networking/communication infrastructure
- Application environment (e.g., Mobile)

- New application areas
- Business environment

Expansion of Application Domain

- Engineering and scientific applications
- Business information processing applications (security, data integrity)
- Embedded controller applications (real-time, safety critical)
- Mobile communication, Smart home, u-Health, etc. (dependability, security critical)
- Systematic collection of real-time data in public sector for situation awareness – requires an inter-disciplinary approach (e.g., customer complaints, customer expectation)
- Green computing, u-health, u-manufacturing, …

Evolution of Computing Environment

- **Stand alone**
- **Super computer/parallel processing (execution speed)**
- **Networked/distributed environment (security, data integrity)**
- **Internet web, big data (security, scientific/social findings)**
- **Ubiquitous computing (context awareness)**
- **Cloud computing (dependability)**
- **Edge/fog computing (dependability, energy saving)**
- **Green computing (energy saving, energy-aware software design) –** "Harvard University Physicist Alex Wissner-Gross says that a single search on Google generates about 7 grams of CO2 and two searches sums up the energy equivalent to boiling the kettle for a cup of tea."

Evolution of Development Environments

- **In house development**
- **System integration (SI)**
- **Development outsourcing**
- **(COTS) Component-based development**
- **Global development (distribute development)**
- **Open source**
- **Product line engineering**
- Embedment of software engineering knowledge in the system
- Run-time verification
- Asset specification technique (e.g., adaptation rules)

Changes in Business Environments

- **Supplier-driven market**
- **Customer-driven market**
- **Short product life-cycle (agility)**
- **Diverse market needs (product line)**
- **Customer specific requirements**
- **On-demand services (SOA- accident-> insurance)**
- **Global engineering**

Development of New Technologies

- Computer hardware
- Communication technologies
- Internet Web
- Middle-ware
- Operating systems
- Artificial intelligence/Machine learning
- Cloud/edge/fog computing
- Bio-inspired SE (seamless integration of new services without human intervention)

Agenda

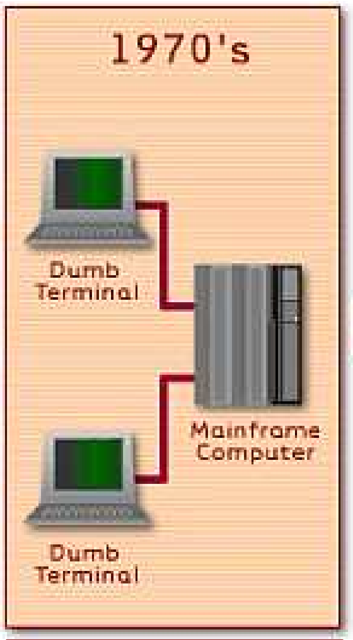
- Prologue
- Technology Evolution: Looking Back
- Driving Forces of Technology Evolution
- **Technology Evolution: Looking Forward**
- Epilogue

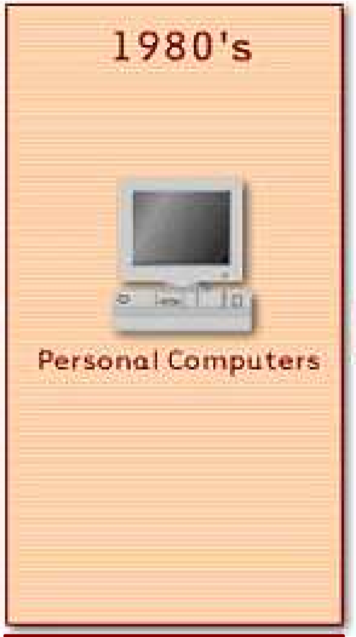
Technology Evolution: Looking Forward

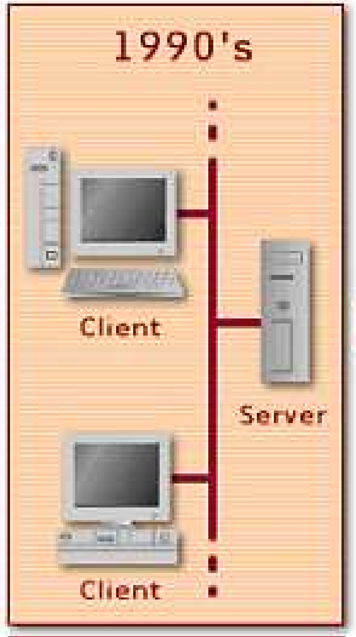
1. Evolution of Computing Technologies
    - Cyber Physical Systems (CPS)
    - Open Systems
    - Ecosystems
    - Systems of systems
2. Evolution of Software Engineering Technologies

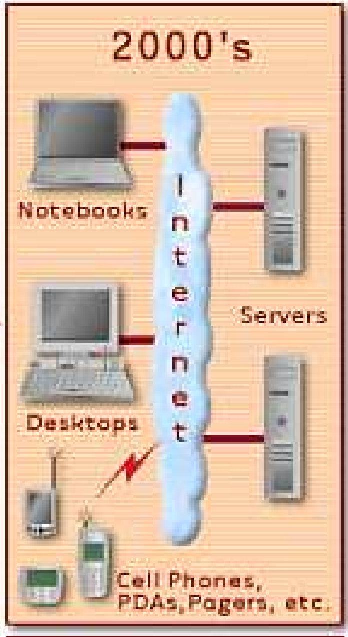
Evolution of Computing Technologies

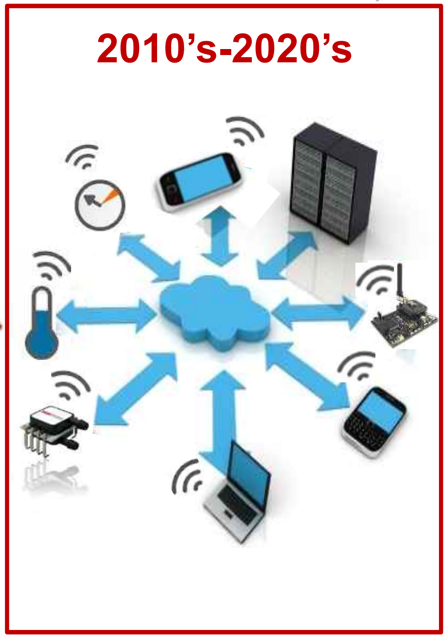
Stand-alone → **Interoperate**











Mainframe Computing	Desktop Computing	Wired Network	Wired + Wireless Network	Open Environment (Ambiguous Boundary)

Performance **Maintainability** **Complexity** **Security** **Self-Adaptiveness** **Feature Interactions**

Images from: 1) "Information Model: The Key to Integration," AutomatedBuildings.com Article, Jan. 2002.;
2) Judy Quinn, "Why switch from Desktop to Cloud Project Management Software?," Project Management Articles, Apr. 8, 2022

Evolution of Computing Technologies (Cont.)

Interoperate

2010's-2020's

Open Environment
(Ambiguous Boundary)

- **Cyber Physical Systems (CPS)**
- **Open Systems**
- **Ecosystems**
- **Systems of Systems (SoS)**

Self-Adaptiveness

Feature Interactions

Interoperate

2010's-2020's

Open Environment
(Ambiguous Boundary)

Cyber Physical Systems (CPS)

Open Systems

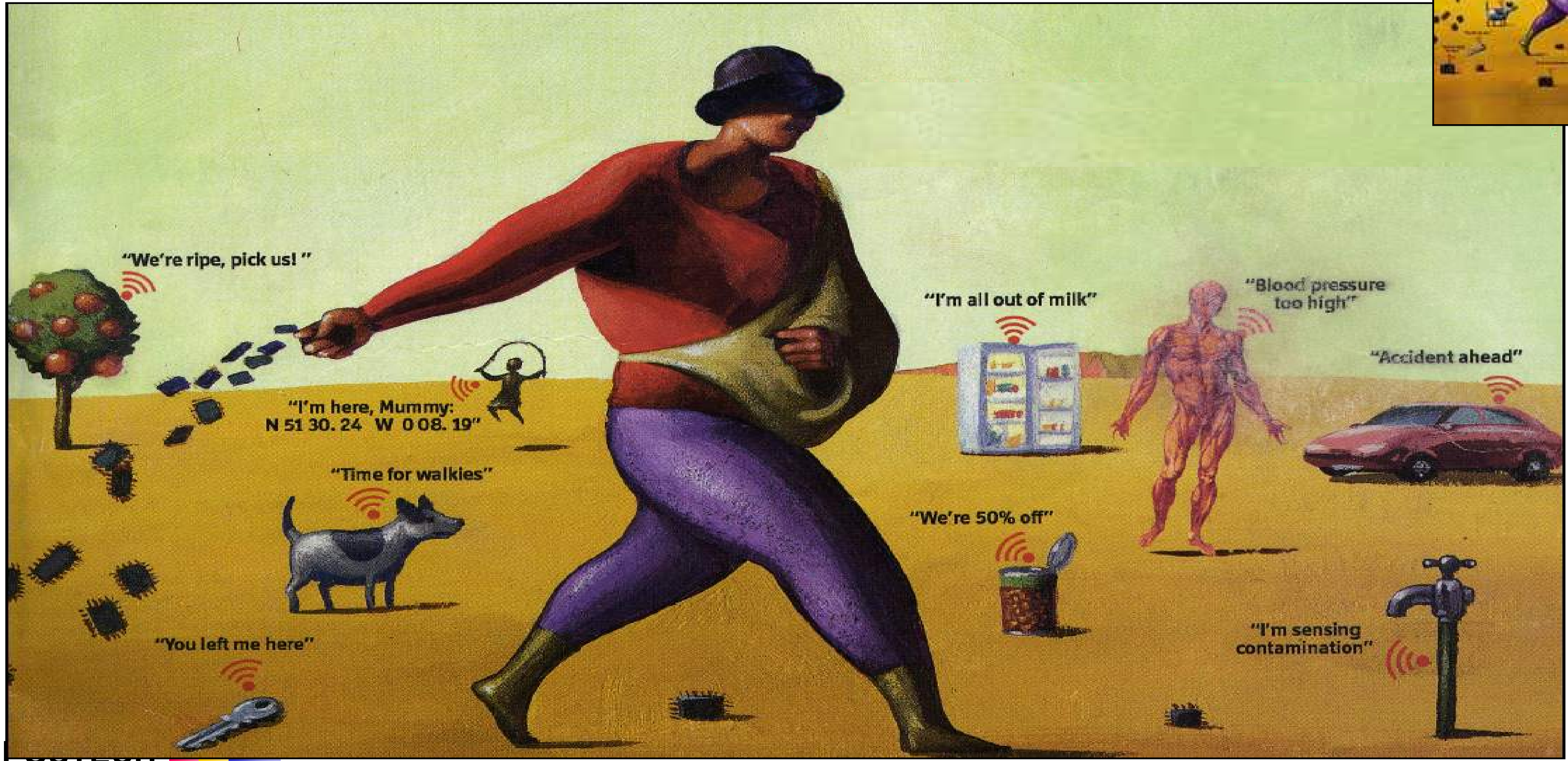
Ecosystems

Systems of Systems (SoS)

Self-Adaptiveness

Feature Interactions

- “When everything connects”
  - The Economist (April 28th, 2007)



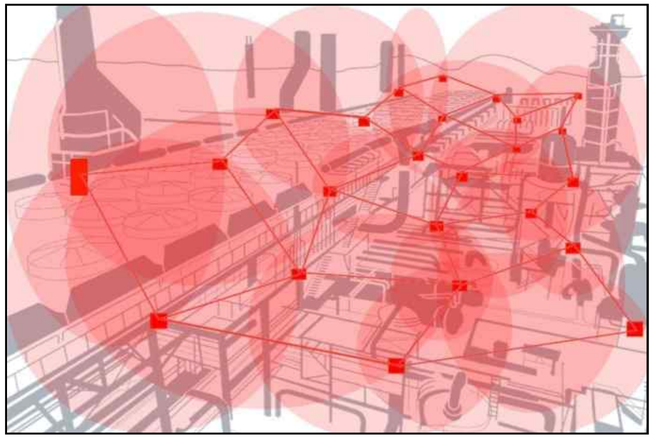
What is CPS? (Cont.)

- CPS (Cyber Physical Systems):
  - A number of sensors, actuators, and systems to monitor, make intelligent decisions, and control physical entities

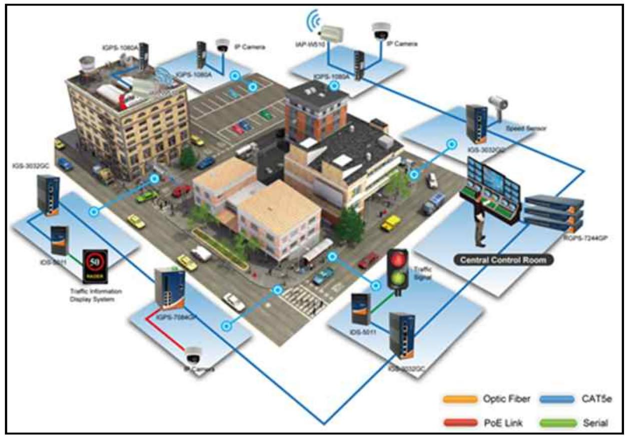
- Potential applications of CPS
  - Social infrastructure monitoring
  - E-city
    - Disaster/accidence monitoring and rescue team dispatching
    - Traffic control
  - Intelligent highways and automobiles
  - Smart buildings and power grids
  - Military systems
  - Medical and healthcare systems
  - …

Applications of CPS

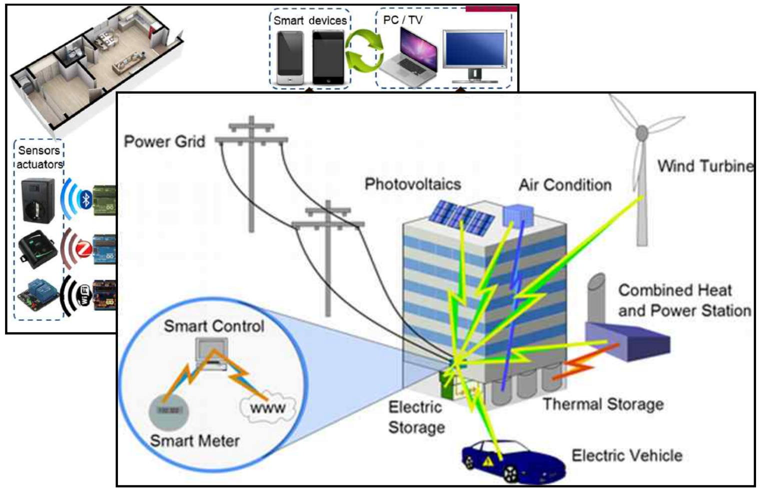
Social Infrastructure



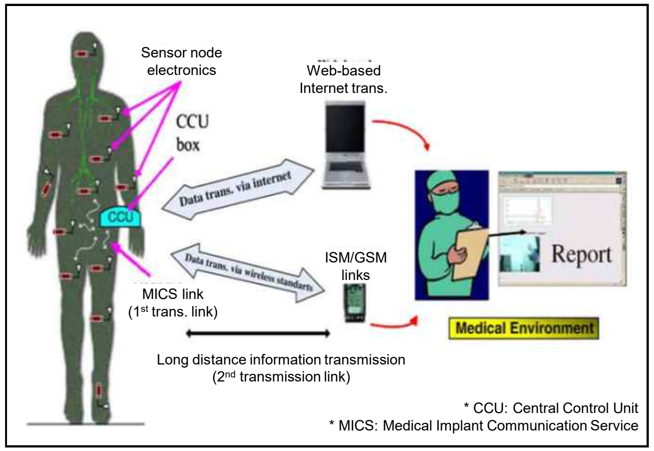
Intelligent Transportation



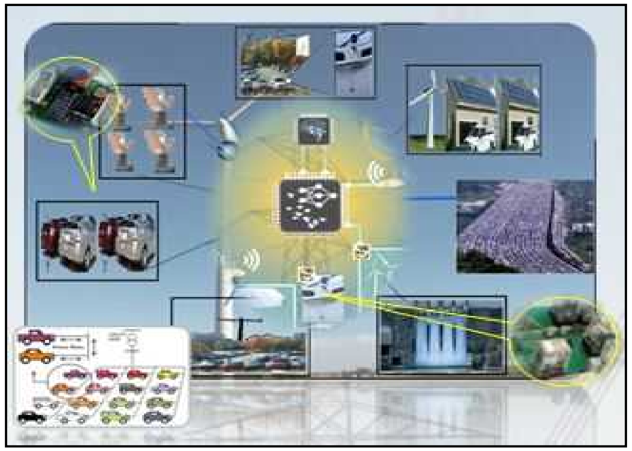
Smart Buildings/Homes



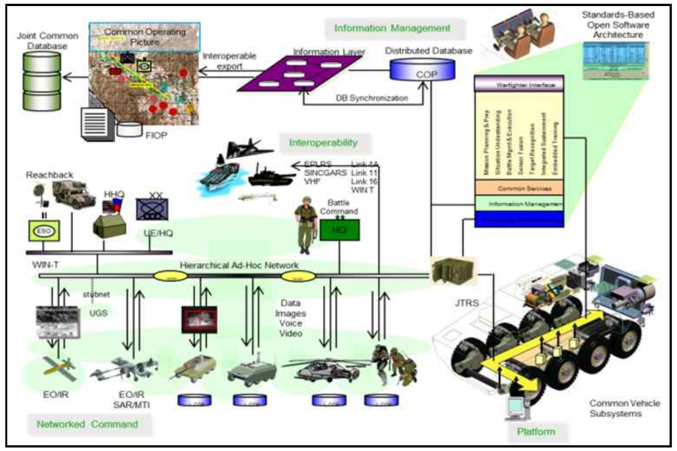
Medical and Healthcare


Sensor node electronics
Web-based Internet trans.
CCU box
Data trans. via internet
CCU
Report
Data trans. via wireless standards
ISM/GSM links
MICS link (1st trans. link)
Medical Environment
Long distance information transmission (2nd transmission link)
* CCU: Central Control Unit
* MICS: Medical Implant Communication Service

Power and Energy



Weapon Systems



Images from: 1) "Wireless Control Networks: A New Approach for Control over Networks," Personal Blog, Miroslav Pajic – Paya, http://www.seas.upenn.edu/~pajic/research.html/.
2) "Intelligent Transportation System," ORing Industrial Networking Corp., http://www.oring-networking.com/doc/view/sn/276/Intelligent%20Transportation%20System/.
3) "SAMBA: Self-Adapting Monitoring for Building Automation" Institute for Dynamic Systems and Control, http://www.idsc.ethz.ch/Research_Guzzella/Building_Technology/SAMBA
4) J. Xiao and R. Boutaba, "The Design and Implementation of an Energy-Smart Home in Korea," *Journal of Computing Science and Engineering*, 7 (3): 204-210, September 2013.
5) M.R. Yuce, et. al.,"Wireless Body Sensor Network Using Medical Implant Band," *Journal of Medical Systems*, 31 (6): 467-474, 2007.
6) "Intelligent Cyber-Physical Power and Energy Systems," RTPIS Lab., CLEMSON University, http://rtpis.org/intelligent-cyber-physical-power-and-energy-systems/
7) J. Sztipanovits, "Cyber Physical Systems (CPS)," ISIS, Vanderbilt University.

Benefits of CPS

- Adding "new" capabilities to physical systems
  - "everything talks and connects"

- Improving quality of life/systems in real world based on continuous monitoring and intelligent decision making
  - **Safety**
    - Bridge health monitoring system
    - Smart home for healthcare
    - Seismic warning system, etc.
  - **Efficiency**
    - Efficient route planning system based on traffic condition awareness
    - Building management system with efficient energy control, etc.

Challenges of CPS

- Danger of unexpected feature interactions
- Real-time response
- Power supply
- Mobility
- Rapid evolution of technologies
- Mixture of heterogeneous technologies
- Fault-tolerance
- Trust, security, and privacy
- Large amount of real-time data being produced ("big data")
  - Real-time identification of important system states

Interoperate

2010's-2020's

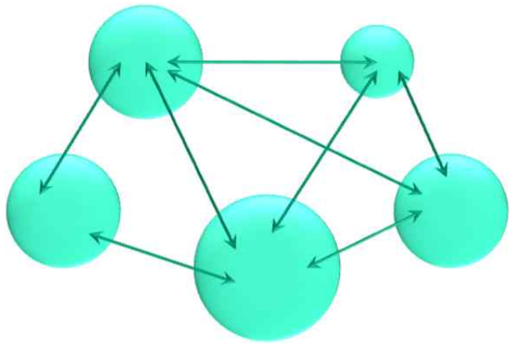
Open Environment
(Ambiguous Boundary)

Self-Adaptiveness

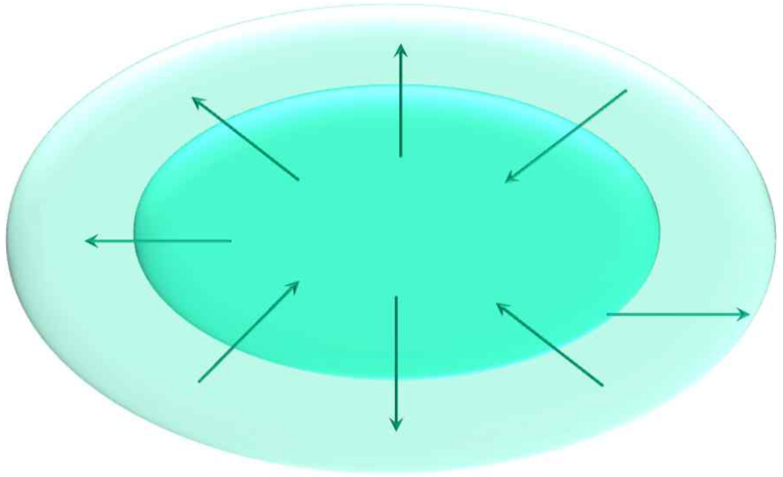
Feature Interactions

- Cyber Physical Systems (CPS)
- **Open Systems**
- Ecosystems
- Systems of Systems (SoS)

Open system – **concepts:**



Interacting/interrelated components



System exists in an environment with which it interacts

- "Open Systems Approach," ACCEL Team Development, http://www.accel-team.com/business_process/

Open system – definition:

– "A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered components to be utilized across a wide range of systems with minimal changes, to interoperate with other components on local and remote systems, and to interact with users in a style that facilitates portability"

- The DoD Open Systems Joint Task Force, 1994.

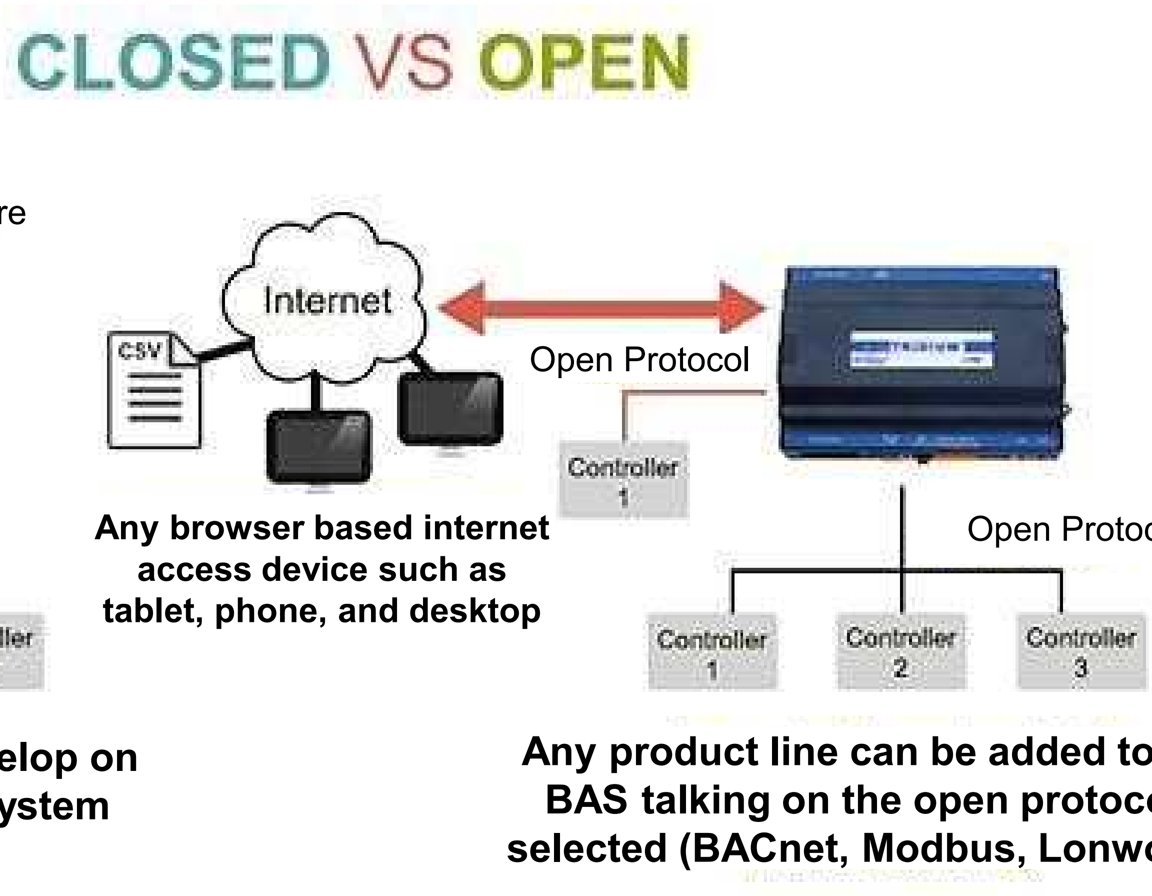
– "A collection of interacting software, hardware, and human components, designed to satisfy stated needs, with the interface specification of components

· fully defined
· available to the public
· maintained according to group consensus, and

in which the implementations of components are conformant to the specification."

- B.C. Meyers and P.A. Oberndorf,
Open Systems: The Promises and the Pitfalls, SEI course, 1996.

Open building automation systems (BAS)


CLOSED VS OPEN
Workstation
Running Proprietary Software
Proprietary Protocol
Controller 1
Controller 2
Controller 3
Only the manufacturer can develop on these devices with a closed system
Internet
CSV
Any browser based internet access device such as tablet, phone, and desktop
Open Protocol
Controller 1
Open Protocol
Controller 1
Controller 2
Controller 3
Any product line can be added to the BAS talking on the open protocols selected (BACnet, Modbus, Lonworks)

- Brian Turner, "What Every Building Owner Should Know - About Open Systems and Smart Buildings," Article, AutomatedBuildings.com, Jul., 2013. http://www.automatedbuildings.com/news/jul13/articles/controlco/130629033606controlco.html

Benefits of Open Systems

- Stability of systems
  - Overall system framework and interface standards
- Interoperability with other systems
- Availability to the public
- Framework for ecosystems
- Variability management

Challenges of Open Systems

- Competing standards and technologies
  - Providers’ love-hate relationship with standards
  - Importance of user “clout”
- Rapid evolution of technologies
- Marketplace availability of compliant products
- Security problems

- P. Emmerich et al., “Benefits, Problems, and Issues in Open Systems Architectures,” *IEEE Transactions on Power Systems*, 9 (1), Feb. 1994.

Interoperate

2010's-2020's

Open Environment (Ambiguous Boundary)

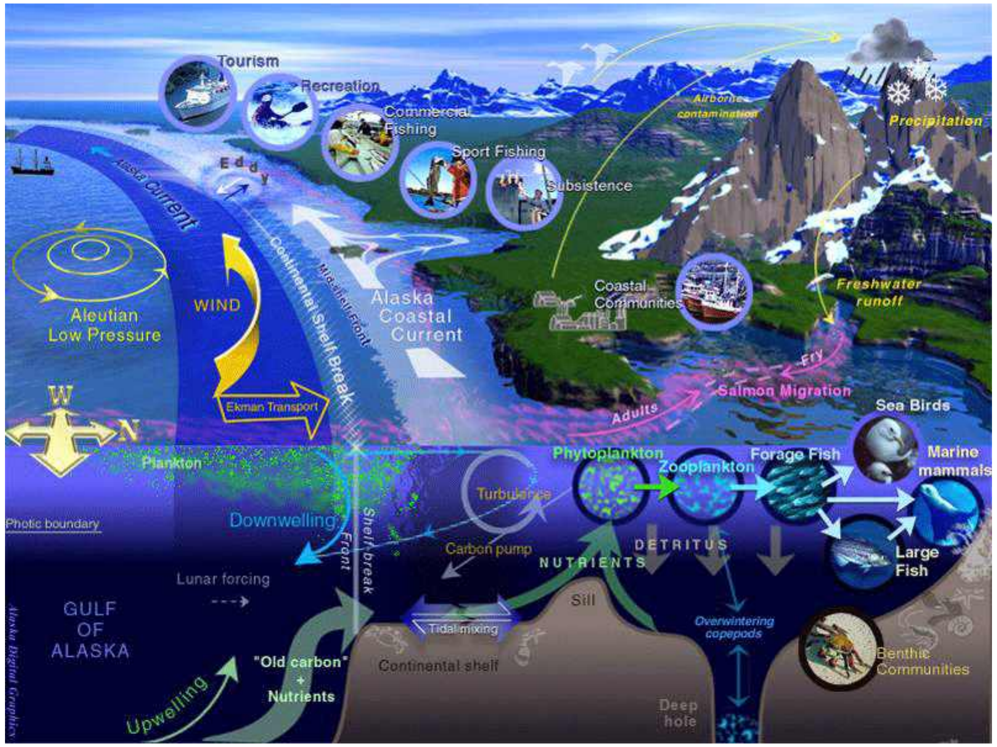
Self-Adaptiveness

Feature Interactions

- Cyber Physical Systems (CPS)
- Open Systems
- **Ecosystems**
- Systems of Systems (SoS)

Ecosystem:

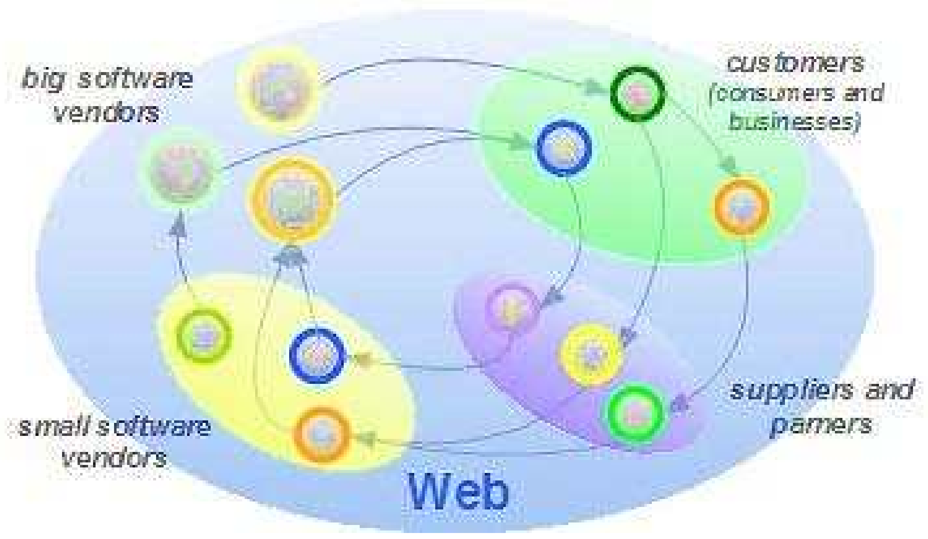
- "A geographically specified system of organisms, including humans, their environment, and the processes that control their dynamics."



- NOAA: Ecosystems Goal Team, http://ecosystems.noaa.gov/what_eco.htm/.

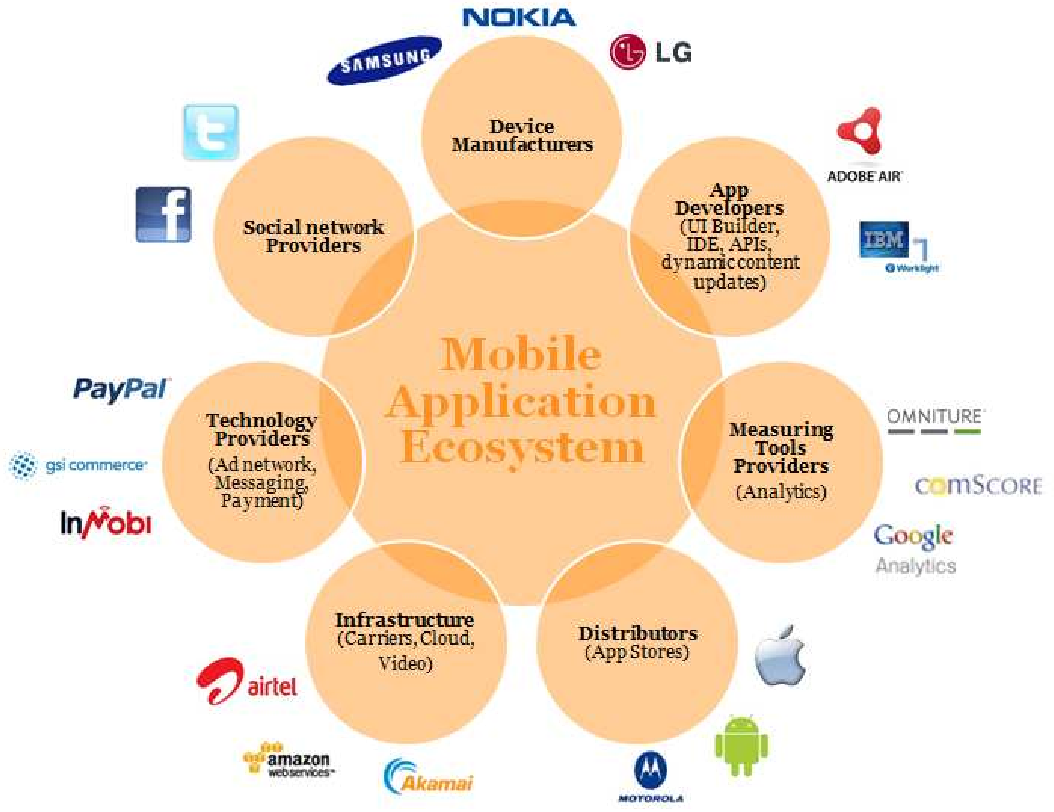
Ecosystem: (Cont.)

- "A software ecosystem consists of the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions (Bosch, 2009)."



- J. Bosch, "From Software Product Lines to Software Ecosystems," in: 13th International Software Product Line Conference (SPLC), August 2009.
- D. Hinchcliffe, "Outlook on software in 2006: Healthy, disruptive," ZDNet, April 10, 2006, http://www.zdnet.com/blog/hinchcliffe/outlook-on-software-in-2006-healthy-disruptive/30.

- Mobile ecosystem


Mobile Application Ecosystem
Device Manufacturers
App Developers (UI Builder, IDE, APIs, dynamic content updates)
Measuring Tools Providers (Analytics)
Distributors (App Stores)
Infrastructure (Carriers, Cloud, Video)
Technology Providers (Ad network, Messaging, Payment)
Social network Providers
NOKIA
LG
SAMSUNG
ADOBE AIR
IBM Worklight
OMNITURE
comSCORE
Google Analytics
MOTOROLA
airtel
amazon web services
Akamai
PayPal
gsi commerce
InMobi

- "The Mobile App Ecosystem – Members & their functionality," Telecominfo's Weblog, http://telecominfo.wordpress.com/2012/03/02/the-mobile-app-ecosystem/.

Benefits of Ecosystems

- Cooperation and knowledge sharing among vendors
- Reduced costs in software development/distribution
- Software co-evolution and innovation among vendors
- Increased transparency
  - Involvement of stakeholders and public

- J.V. Joshua et al., "Software Ecosystem: Features, Benefits and Challenges," *International Journal of Advanced Computer Science and Applications*, 4 (8), 2013.

Challenges of Ecosystems

- Establishing adequate actors/relationships
- Mixture of heterogeneous software licenses and technologies
- Heterogeneity of software evolution
- Technical/socio-organizational barriers for communication/coordination
- Insufficient infrastructures and tools
  - for development across organizations involved in ecosystems
- Security problems

- J.V. Joshua et al., "Software Ecosystem: Features, Benefits and Challenges," *International Journal of Advanced Computer Science and Applications*, 4 (8), 2013.

Interoperate

2010's-2020's

Open Environment (Ambiguous Boundary)

Cyber Physical Systems (CPS)

Open Systems

Ecosystems

Systems of Systems (SoS)

Self-Adaptiveness

Feature Interactions

What is SoS?

- System of systems (SoS):
  - "A set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities (DoD, 2004)".

- Examples:
  - Air traffic control
  - Network-centric warfare (NCW)
  - Future combat systems
  - Intelligent transportation systems
  - Medical and healthcare systems
  - U-city
  - …

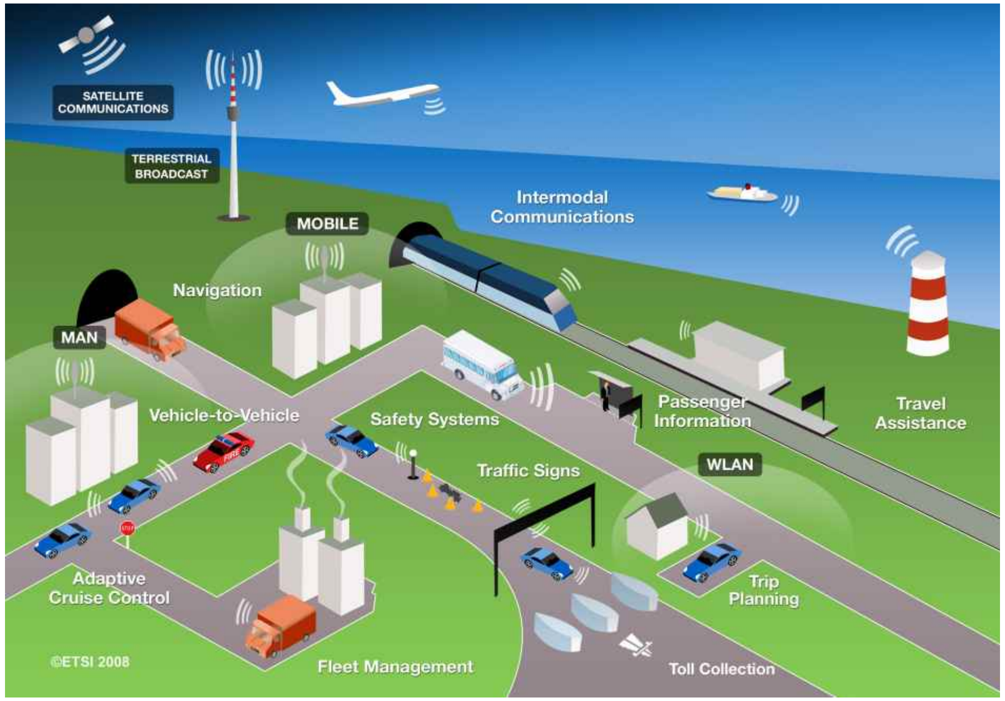
- Department of Defense (DoD), 2004, Defense Acquisition Guidebook Ch. 4 "System of Systems Engineering," Washington, DC: Pentagon, October 14, 2004.
- M. Jamshidi, "Control of Large-Scale Complex Systems – From Hierarchical to Autonomous and now to System of Systems," Electrical and Computer Engineering Department and Autonomous Control Engineering (ACE) Center, University of New Mexico, Albuquerque

What is SoS? (Cont.)

- SoS vs. large systems
  - SoS requires:
    - Operational/managerial independence of the elements
    - Evolutionary development
    - Emergent behaviors
    - Geographic distribution
- SoS vs. CPS (Cyber Physical Systems)
  - SoS requires:
    - Geographic distribution
    - System composition for mega-systems
  - CPS requires:
    - Closing the loop in the "real" world

- M. Jamshidi, "Control of Large-Scale Complex Systems – From Hierarchical to Autonomous and now to System of Systems," Electrical and Computer Engineering Department and Autonomous Control Engineering (ACE) Center, University of New Mexico, Albuquerque
- T. Samad and T. Parisini, "Systems of Systems," The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), 2011.

- Intelligent transportation systems (ITS)
  - Advanced traveler information systems (ATIS) and advanced traffic management systems (ATMS)



M. Mosley, “The future of Intelligent Transport Systems (ITS)” Engaged IT for the CIO, Nov. 29, 2011, http://mubbisherahmed.wordpress.com/2011/11/29/the-future-of-intelligent-transport-systems-its/

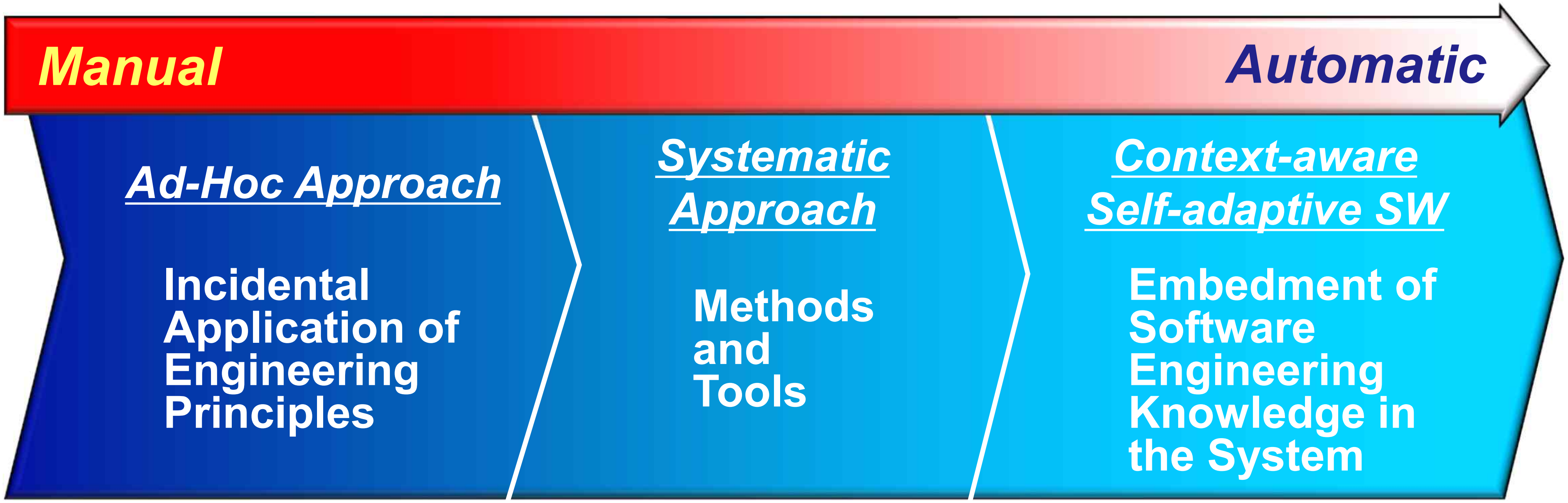
Benefits of SoS

- Meeting needs with a mix of constituent-systems
- Operational/managerial independence of constituent-systems
- Extensibility of systems

Challenges of SoS

- System of systems being configured dynamically
- Uncertain environment
  - Decision making under uncertainty
- Limited central control
- Danger of unexpected feature interactions
- Coordinating development of new capabilities across constituent-systems
- Heterogeneity of software evolution
- Rapid evolution of technologies
- Mixture of heterogeneous technologies
- Trust, security, and privacy

Technical Trends


Manual
Automatic
Ad-Hoc Approach
Incidental Application of Engineering Principles
Systematic Approach
Methods and Tools
Context-aware Self-adaptive SW
Embedment of Software Engineering Knowledge in the System

- Modularization
- Information hiding
- Encapsulation

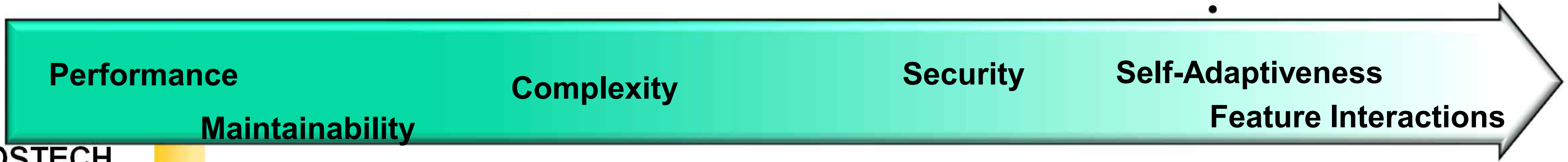
⋮

- Commonality and variability analysis
- Parameterization
- Template framework

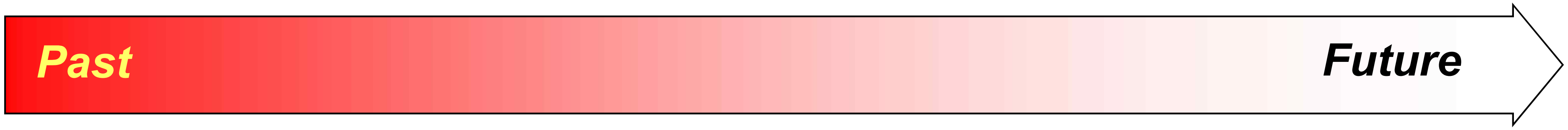
⋮

- Monitor and dynamic reconfigurator
- Dynamic binding
- Architecture reconfiguration
- Self-healing
- Runtime Verification

⋮



SE Technology Trend



Requirements Specific Engineering

-A single system

Asset-based Engineering

-A family of systems
-Integration rules
-Asset verification
-Asset specification
-Adaptation rule specification

Context-aware Self-adaptive Systems Engineering

-A dynamically reconfigured system

-Run-time verification

-Application of ML techniques

- **MACHINE LEARNING APPLICATIONS IN SOFTWARE ENGINEERING**
- edited by **Du Zhang** *(California State University, USA)* **& Jeffrey J P Tsai** *(University of Illinois, Chicago, USA)*

ML Applications in Prediction and Estimation
ML Applications in Property and Model Discovery
ML Applications in Transformation
ML Applications in Generation and Synthesis
ML Applications in Reuse
ML Applications in Requirement Acquisition
ML Applications in Management of Development Knowledge

Epilogue

- Boundary between software engineering and systems engineering becomes fuzzy
- More and more functions are implemented as hardware
- Software engineering should be done in the context of systems engineering (“e-type” systems)
- Awareness of contextual changes and self-adaptation is becoming important; application of ML techniques might be needed
- V&V is becoming harder and more complex
  - Detection/prevention of unintended side-effects

The End