2-22-2019

IoT and Digitization Will Reconnect System Engineering and Science

John Blyler
JB Systems

IoT and Digitization Will Reconnect System Engineering and Science

By John Blyler, Founder, JB Systems

PSU SySc Seminar, Feb 22, 2019

John Blyler's Background

- BS Engineering Physics, MS EE (digital and RF)
- Engineering/Management Background
  - 20 years – Systems of Systems, HW-SW Dev.
  - DOD – DOE – Semi - Commercial
- Teaching Background
  - Develop/taught courses for industry and university
  - Affiliate Professor – PSU, Systems Engineering
  - Lecturer – UC Irvine, SE and IOT Certificates
- Media Background
  - VP, Editor-in-Chief: Systems, Semi, Electronics
  - Associate Editor, IEEE publications
  - Accellera-IEEE Standards (IP-XACT)
  - Book author: IEEE, Wiley, Elsevier, SAE
- Contact: jblyler@jbsystech.com
  www.jbsystech.com

Books

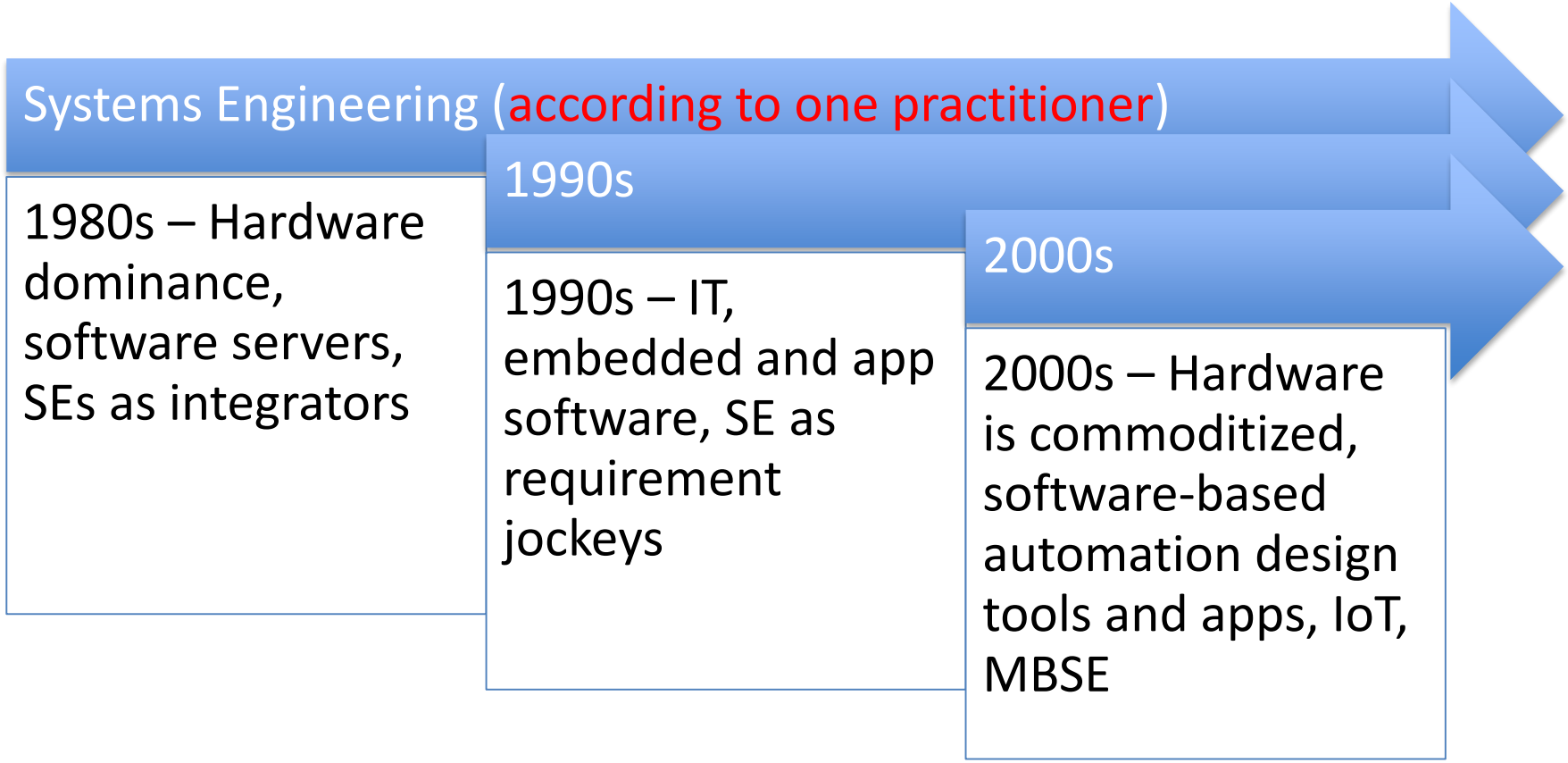
Magazines

Key Points

- Developing an Internet of Things (IoT) system requires managing multidiscipline and multidomain system complexities. To be successful, this will require a tailoring of the traditional system-of-system (SoS) engineering approach.
- System engineers can no longer be **merely** process or requirements experts. They must also have some specific domain knowledge in hardware, software, network/connectivity AND data technologies. This represents a change from traditional systems engineering professionals.
- Model-Based Systems Engineering (MBSE) with domain knowledge will bring the engineering back into systems engineering and enable closer ties to system science.

Evolution of SE Over the Last 40 Years

"**Write what you know**." – Mark Twain



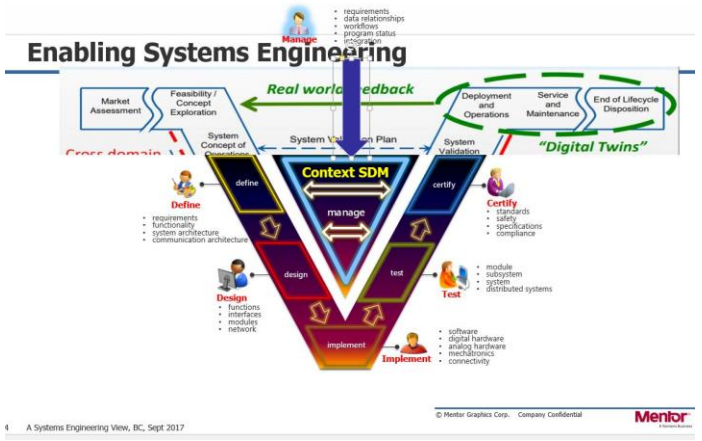
Putting Engineering Back into Systems Engineering

- Has there been a degradation of systems analysis skills to the detriment of the standing of the discipline?
- Has systems engineering become much more about process than outcomes?
- Is it just another engineering discipline rather than the integrating discipline of the parts of the system and the system in its context?

From Oct 2015 NDIA 18th Annual Systems Engineering Conference

Putting Engineering Back into Systems Engineering (Continued)

- The Digital Thread and its connection to the Digital Twin have put the engineering back into systems engineering.
- → MBSE and digital continuity among multidomain architectural-design-testing-manufacturing tools



Relationship to System Science: Two opposite trends, same answer

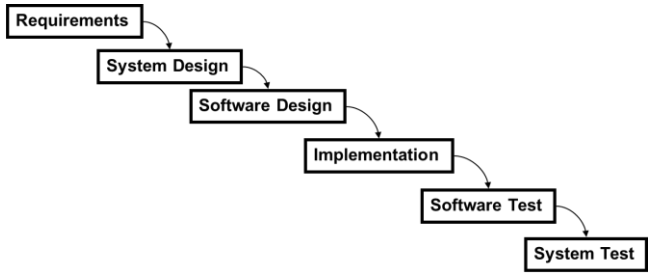
- Trend 1: Divergence of two SEs – for traditional and complex systems
  - Traditional systems engineering: mechanistic, linear, predictable, deterministic, reducible, controllable, static, and existing at one scale
  - Complex systems engineering: include people as well as machines; cross organization boundaries; include multiple disciplines; less predictable, less deterministic, more chaotic, more autonomous, less centrally controlled, and more self organized and adaptive, multiscale
- Trend 2: Blurring of traditional boundaries
  - Natural vs artificial systems
  - Physical vs conceptual systems
  - Science vs engineering disciplines
- Answer to both trends: Integrate while maintaining separation
  - Use systems science as encompassing and integrating foundation of all systems – traditional/complex, natural/artificial, physical/conceptual
  - Use modeling orientation as encompassing and integrating method for all systems, and for integrating and separating science and engineering

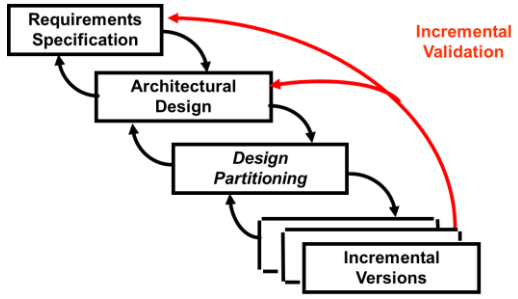
"Ideas on systems science and systems engineering,"Duane Hybertson, MITRE 2006

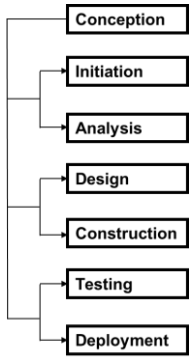
Agenda

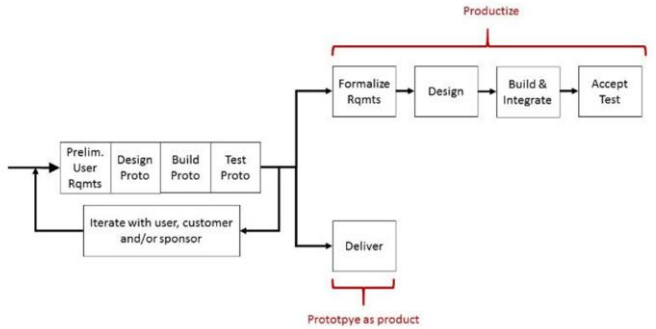
- Lifecycles and Legacies
- Systems Engineering – Key Elements
- IoT Properties
- SE-IoT Motivation and 3 Key Issues
- Summary

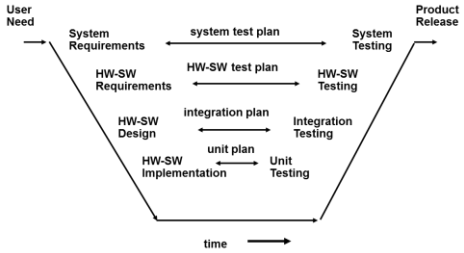
Life Cycle Models (Obligatory Graph)

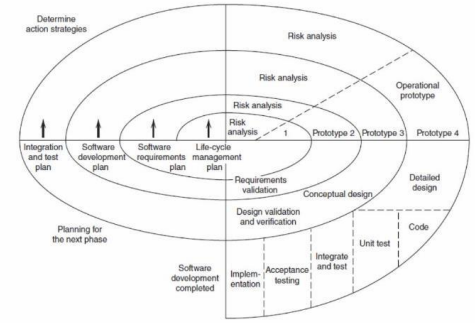


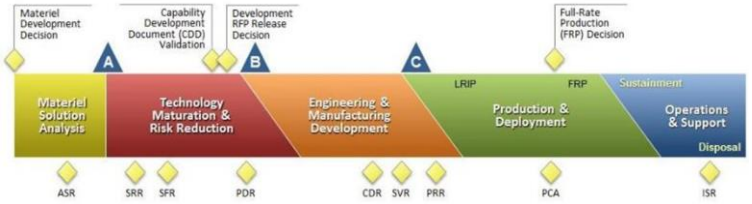




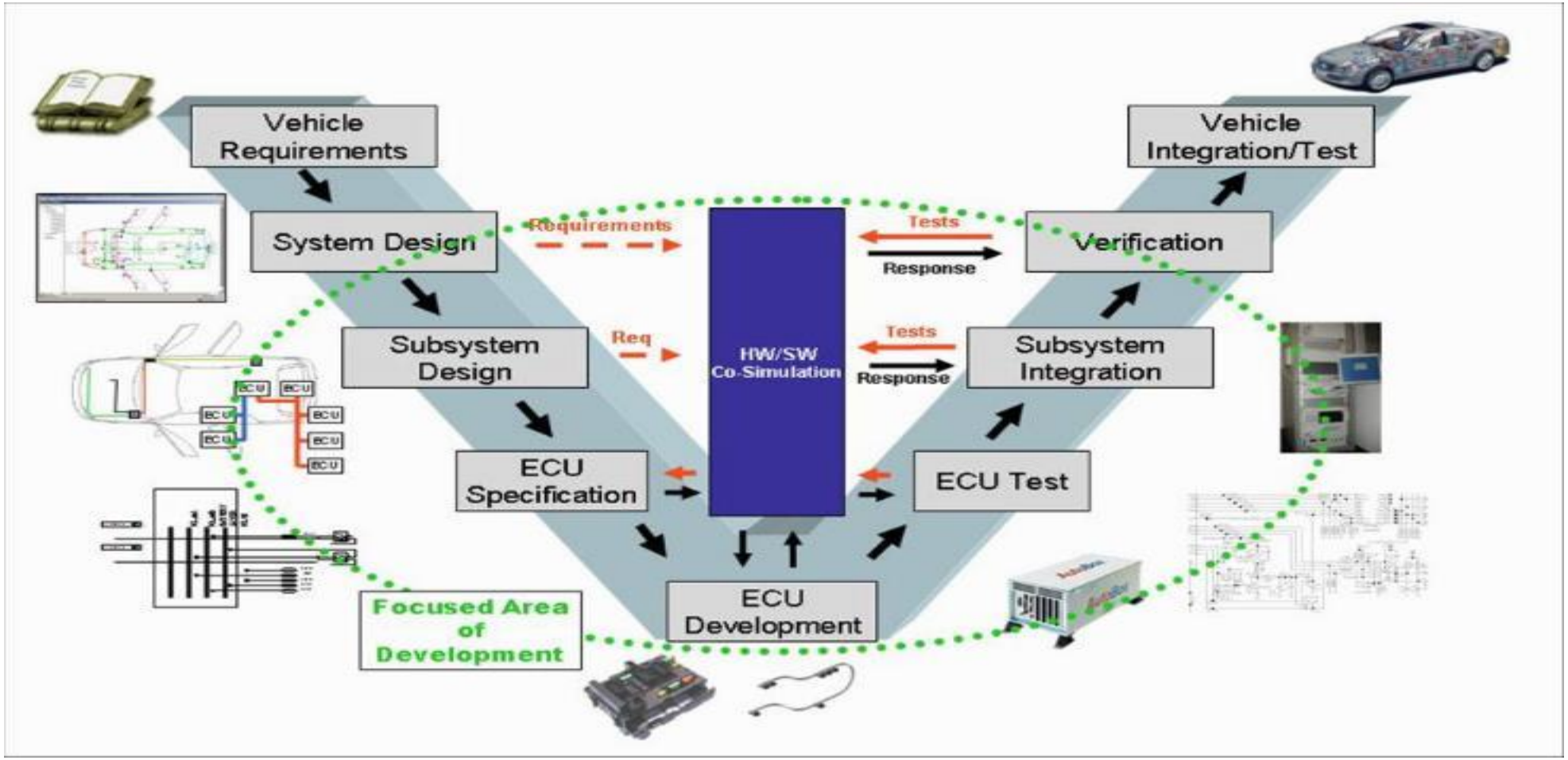






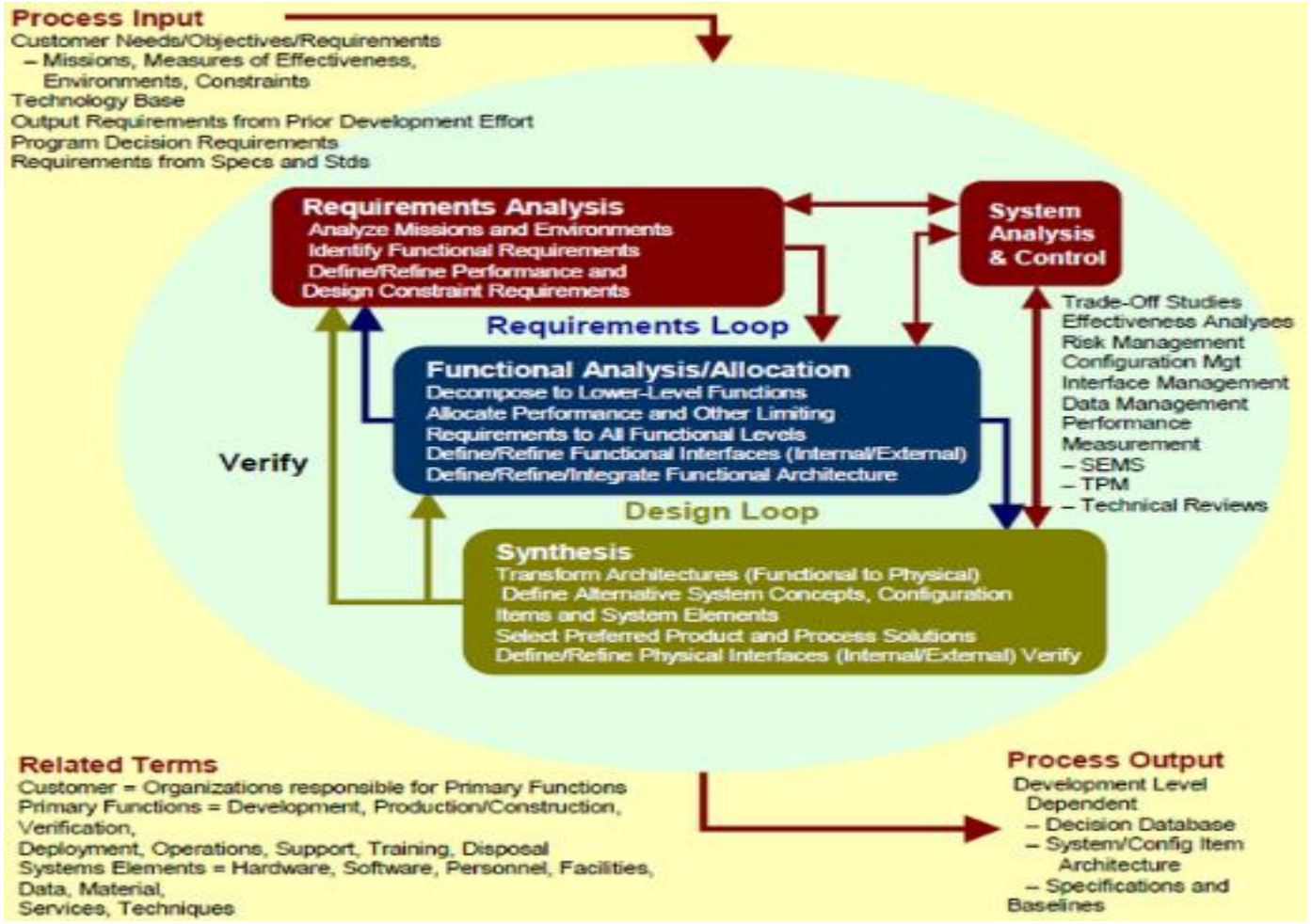


V-Diagram
(Example – Automotive)



Software-Hardware Integration in Automotive Development, SAE Intern., 2014, ISBN 978-0-7680-8052-0

Systems Engineering Process Iterative



Middle-Out Engineering – V Diagram

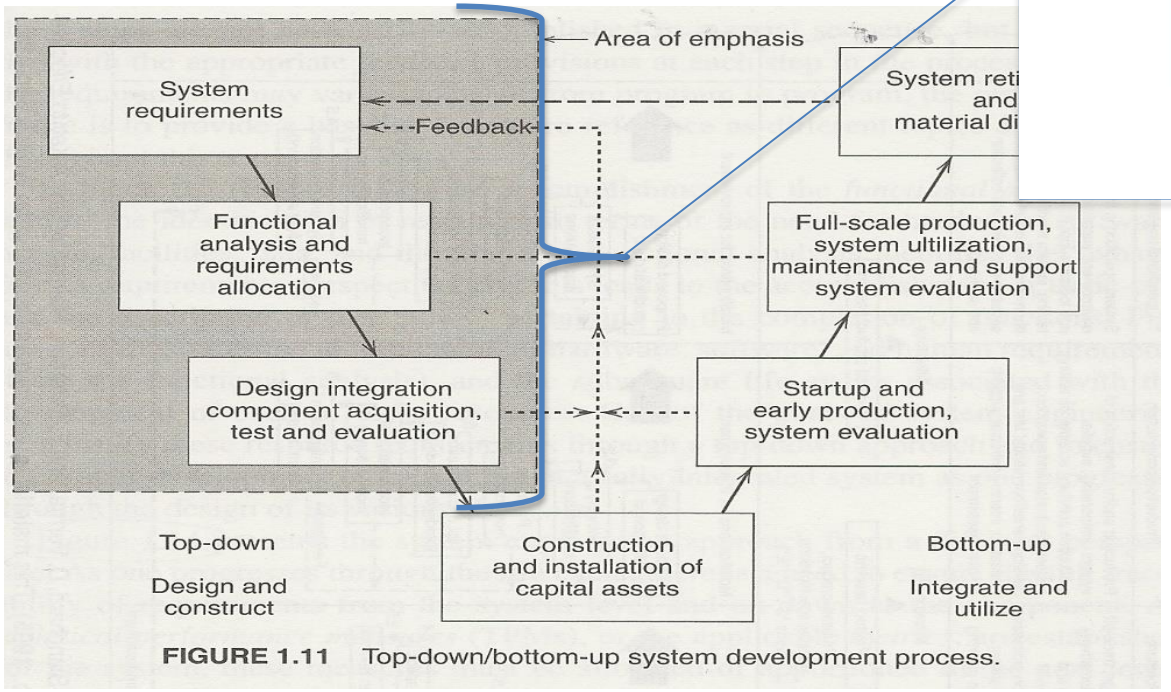
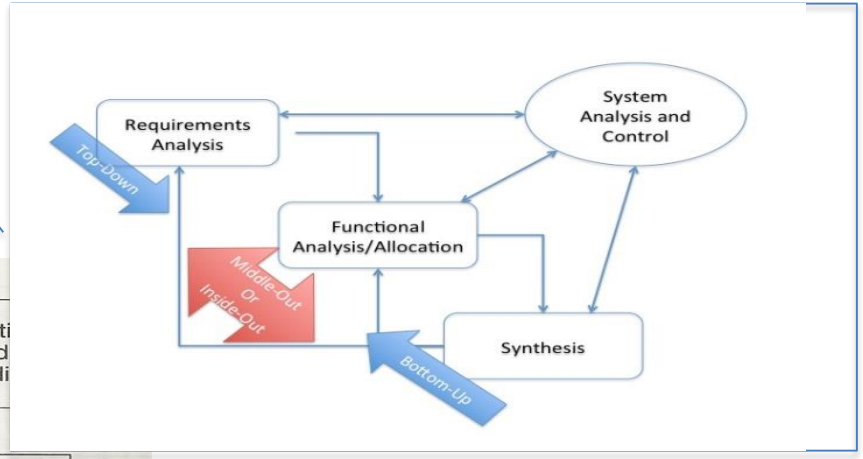
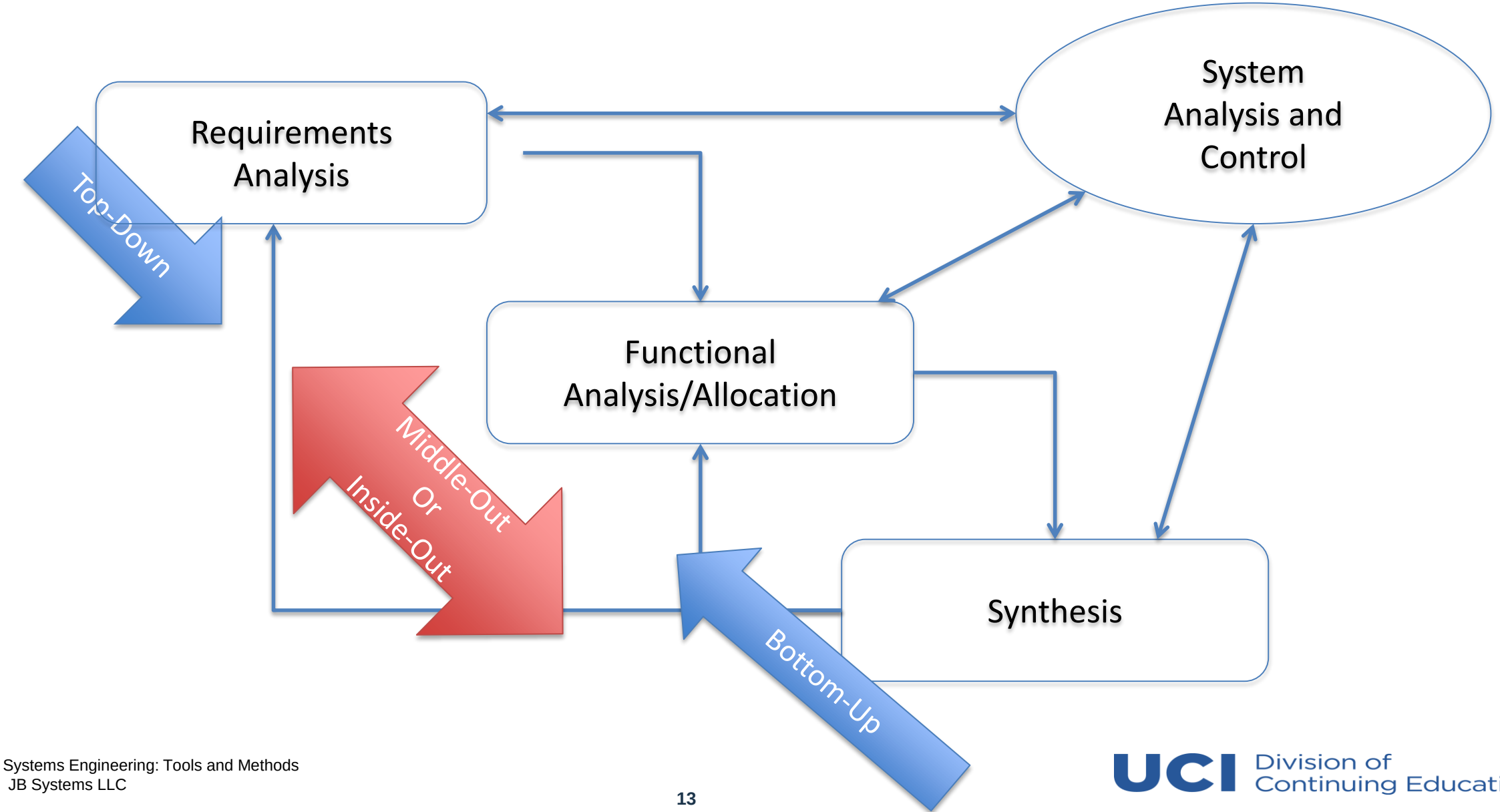


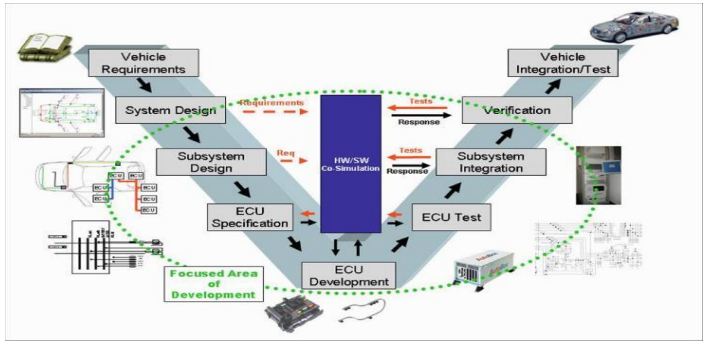
FIGURE 1.11 Top-down/bottom-up system development process.



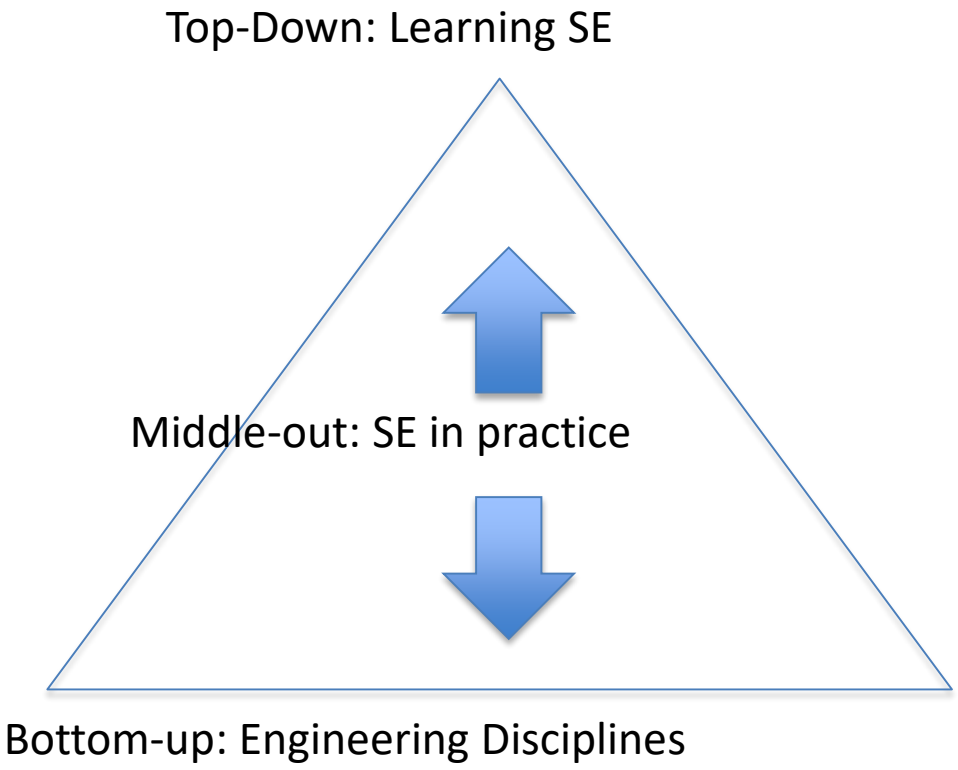
Middle-Out Engineering – Iterative Flow



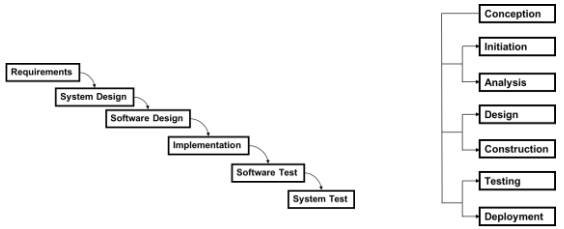
Middle-Out Engineering Overview



- Combination of the two:
  - Bottom – get sense of the system, simple functionalities and scenarios.
  - Top – decomposition that matches top-level physical architecture. Refer to functionalities.



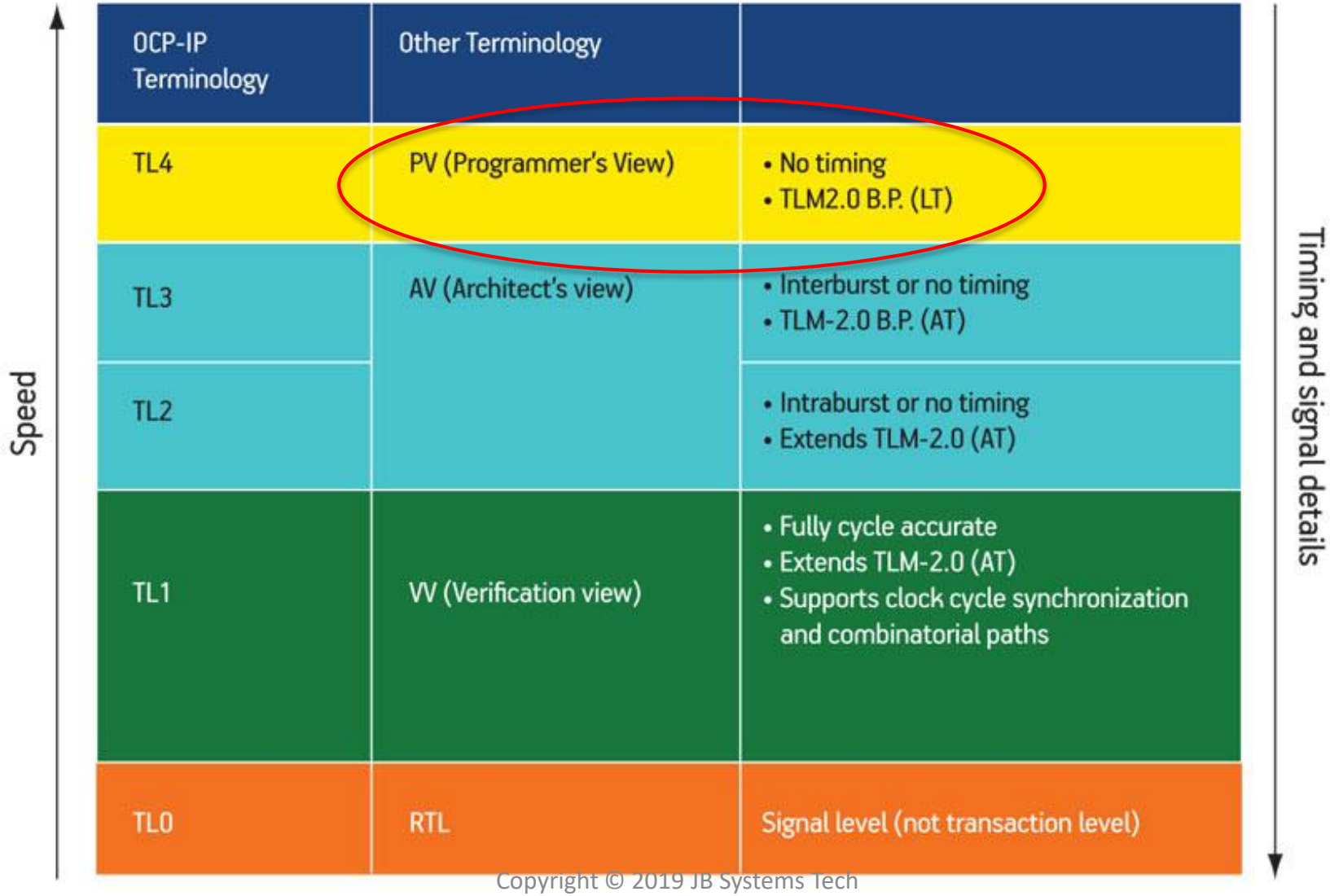
Hardware is Different from Software



- Truly, this is another lecture!
- Suffice to say and especially with the Agile software process:
  - Agile SW and HW Manifesto – Why HW is different:
    1. Lead Time (and timing as shown in the next slide)
    2. Component Cost
    3. Multi-Facetted Work

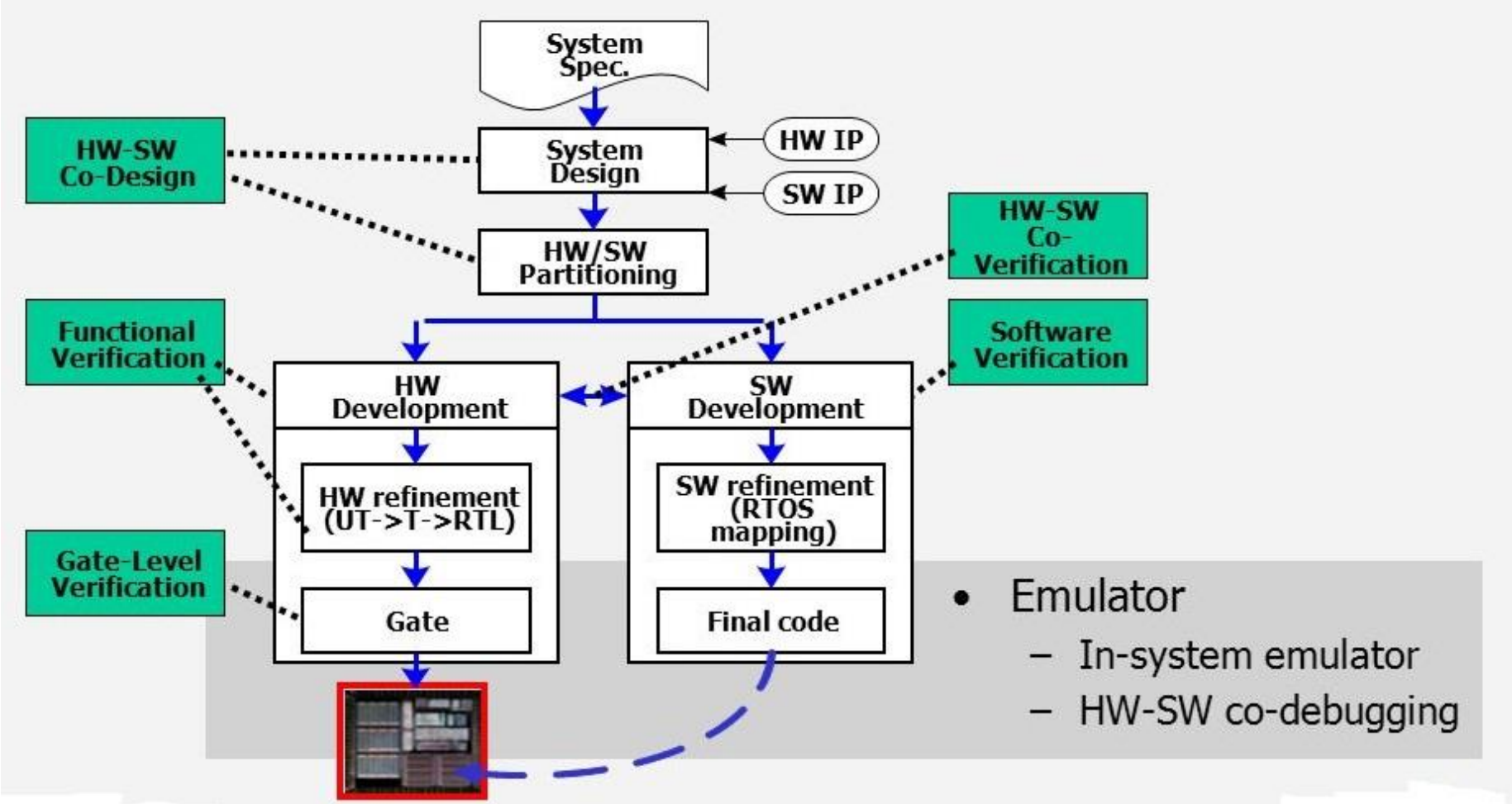
→ If interested, read work on applying Agile Hardware approach to RISC-V (Open Source) MCU: "An Agile Approach to Building RISC-V Microprocessors"

A Word About HW-SW Lifecycles



OCP-IP Terminology	Other Terminology	
TL4	PV (Programmer's View)	• No timing • TLM2.0 B.P. (LT)
TL3	AV (Architect's view)	• Interburst or no timing • TLM-2.0 B.P. (AT)
TL2		• Intraburst or no timing • Extends TLM-2.0 (AT)
TL1	VV (Verification view)	• Fully cycle accurate • Extends TLM-2.0 (AT) • Supports clock cycle synchronization and combinatorial paths
TL0	RTL	Signal level (not transaction level)

Typical SoC Design Flow



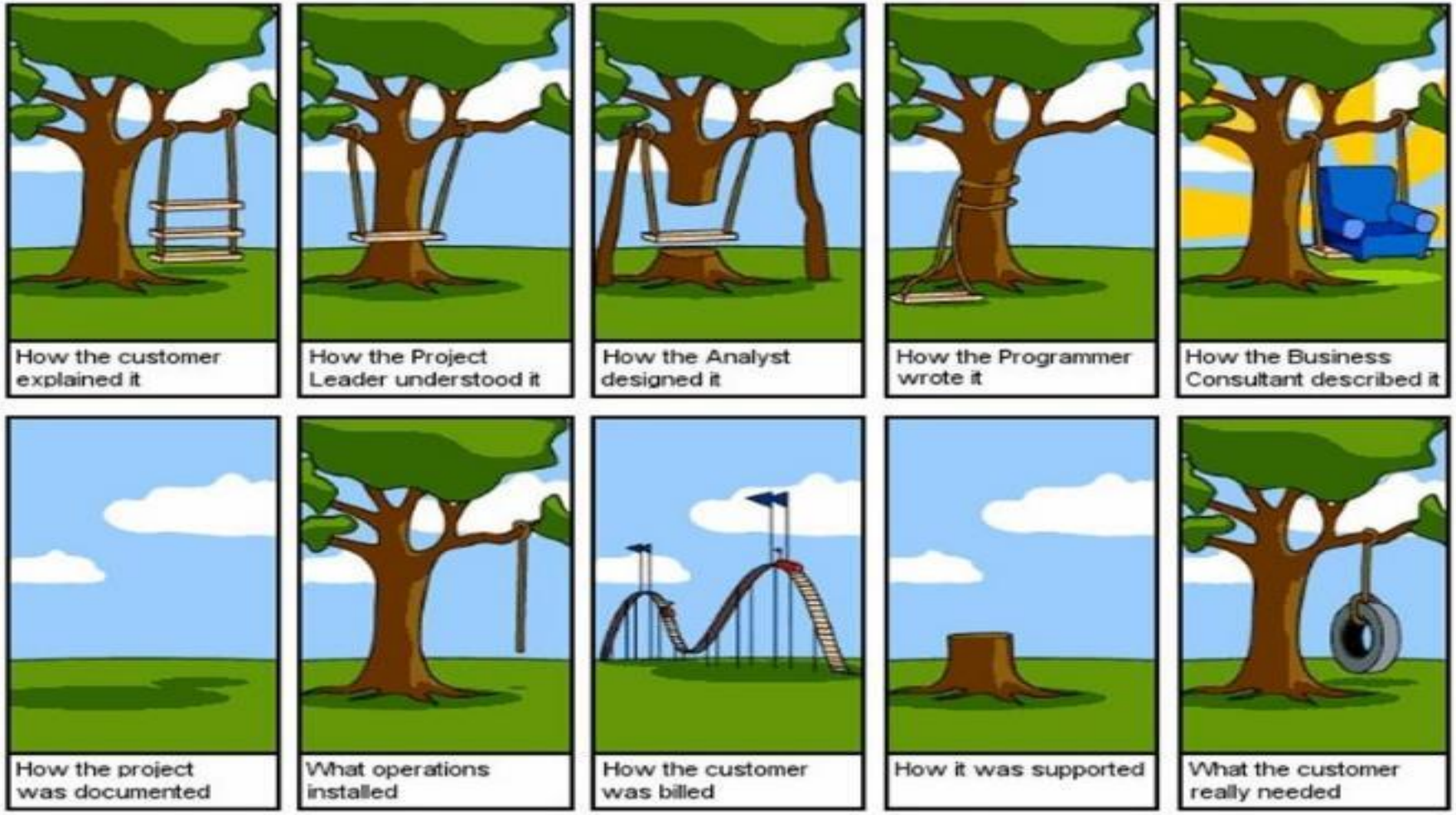
HW – SW Differences

1. Changing when something goes wrong:
   - It is ‘really hard’ for hardware, and somewhere on spectrum from ‘really hard’ to ‘completely trivial’ in software.
2. Adding abstraction to deal with complexity:
   - Software is typically able to ‘absorb’ more of this overhead than hardware.
   - Software it is far easier to only optimize the fast path.
3. Tool set differences:
   - Hardware projects typically stick with really old tools for so long. Why?

Agenda

- Lifecycles and Legacies
- Systems Engineering – Key Elements
- IoT Properties
- SE-IoT Motivation and 3 Key Issues
- Summary

Which System?



Which System Engineer Do You Mean?

- Hardware
- Software (device, application, middleware, etc.)
- Network
- System-of-Systems (SoS)

SE is Good Engineer

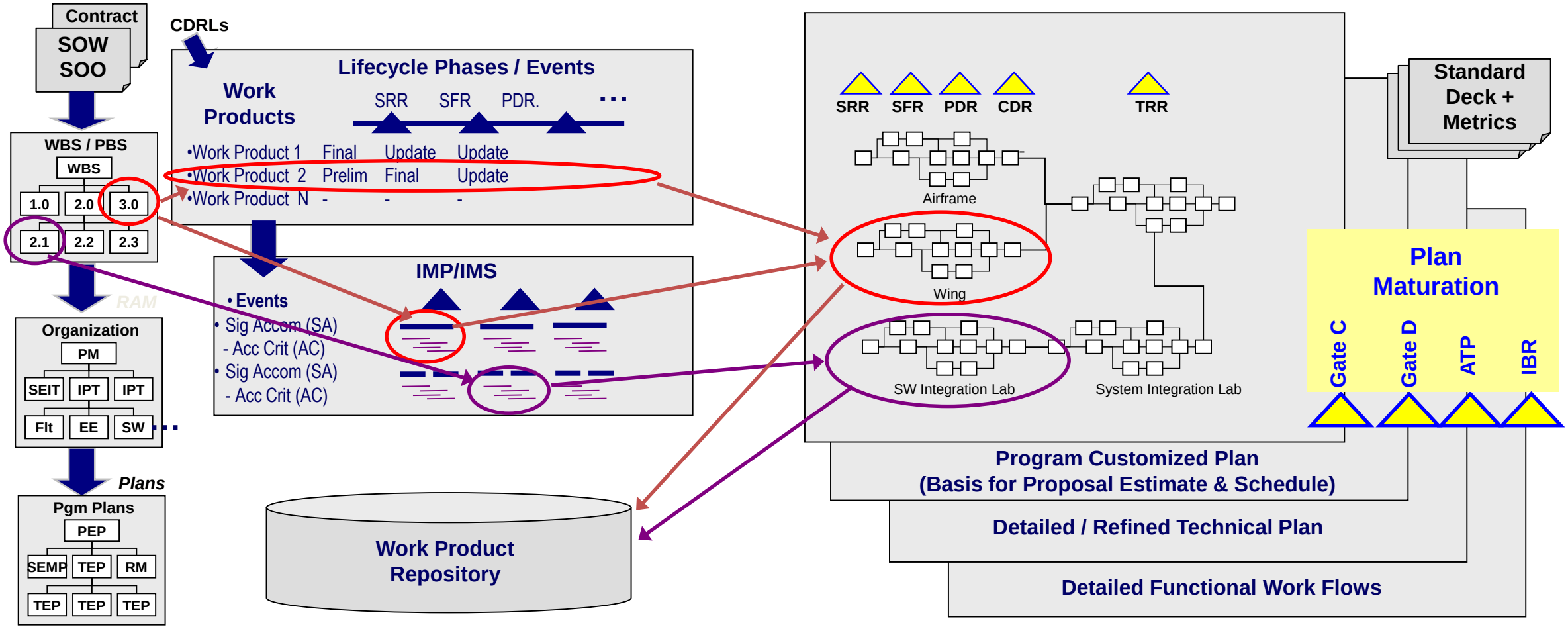
- Systems engineering is good engineering with certain designed areas of interest:
  - Top-Down approach required to view entire system
  - Life-cycle orientation to address all activities and phases
  - Identify the initial system-level requirements to ensure early decision making in design process
  - Interdisciplinary collaborative effort (team environment)
  - Interface management
  - Maintain a multidisciplinary balance
    - Technical product – HW/SW trade-offs
    - Cost and Risk
    - Schedule and budget
  - Define and allocate system requirements
  - Coordinate integration and verification of system

Major Supporting Design Disciplines

Systems Engineering integrates all disciplines and specialties (as shown below):

- Hardware and Software
- Reliability - Resiliency
- Maintainability
- Human factors and safety
- Security engineering
- Manufacturing – Production
- Logistics and supportability
- Disposability
- Environmental

Tech. Planning Requires Tech. View



Make or Buy (System Decisions)

- When to make (contractor's view)
  - Proprietary product
  - Limited or nonexistent market
  - To reduce costs
- When to buy (contractor's view)
  - Workforce limited
  - Lack of skills
  - Outside of product line
  - Off-the-shelf availability

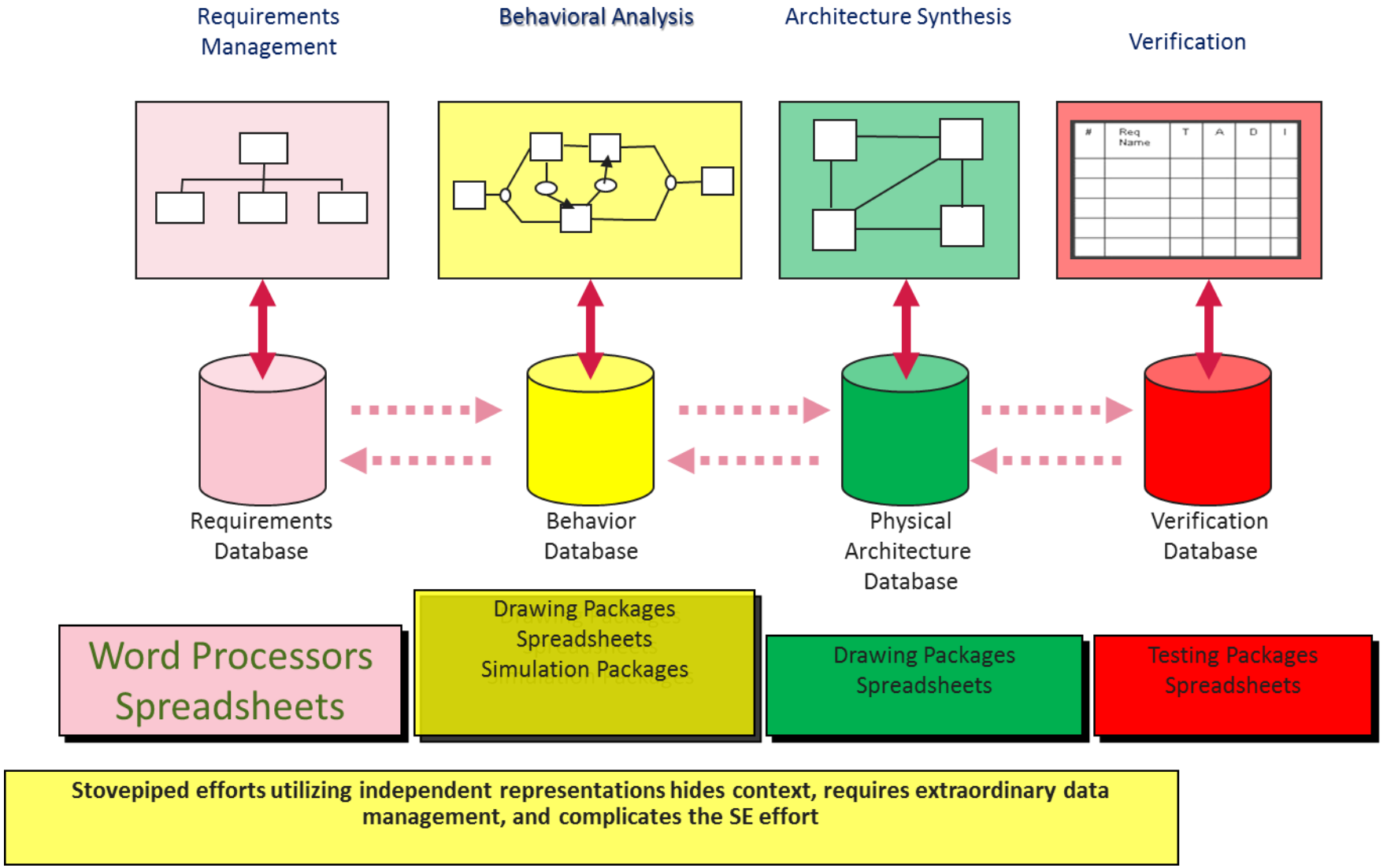
Technology Issues
Forecasting

- Prediction that technical development can occur within specific time period given enough resources
- Normative technology forecasting
  - After desired future goal is chosen, process is developed backward – from future to past – to achieve the goal
- Exploratory technology forecasting
  - Starts at present state of technology and extrapolates into future (assumes some expected rate of technology development).

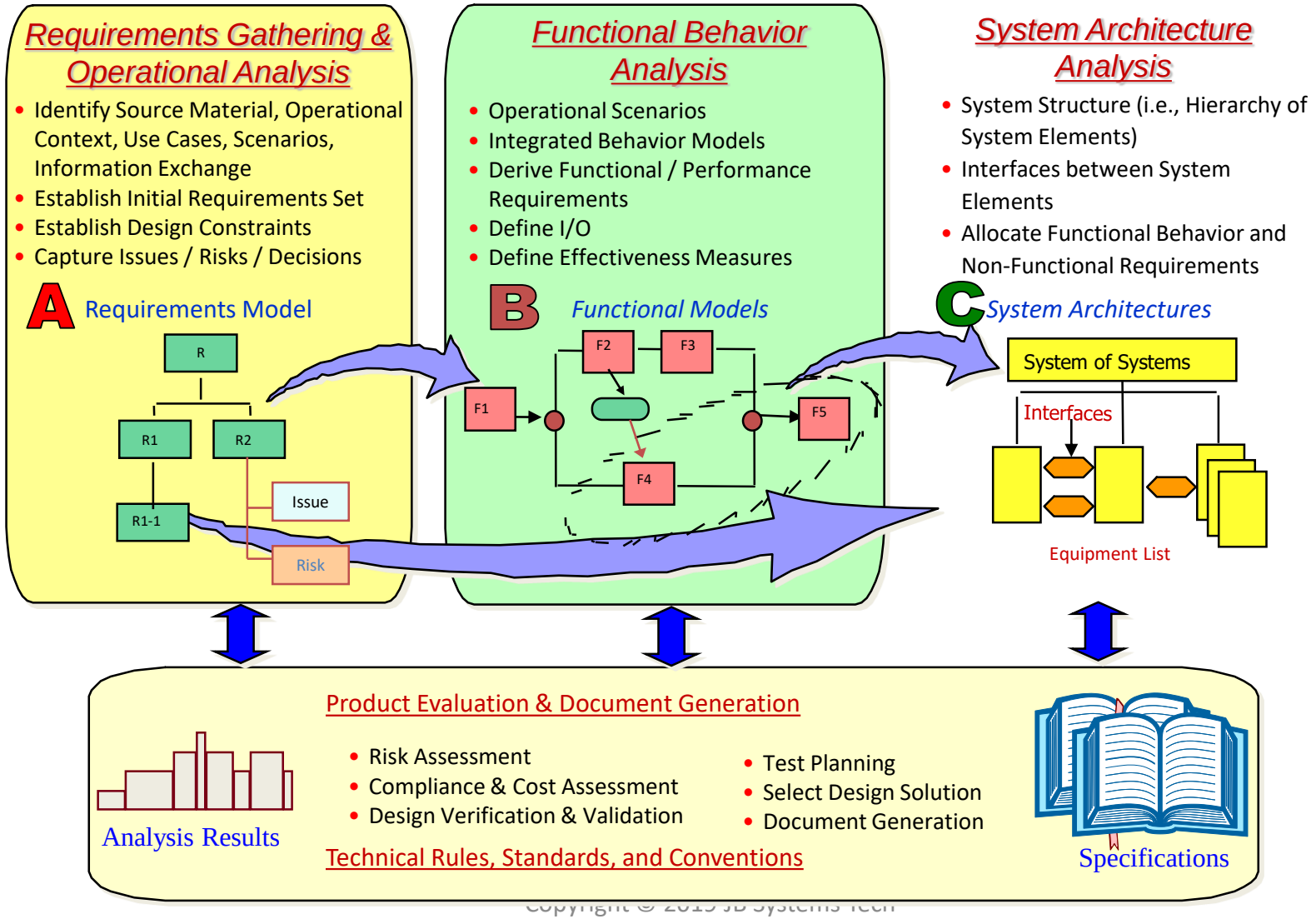
Managing at the Interface

- Essential for HW-SW systems
- Interface-related issues
- Resource margin allocation
- Technical Performance Measurement (TPM) techniques.

Need for MBSE - Silos



Need for MBSE – NASA SRD Development Process



Agenda

- Lifecycles and Legacies
- Systems Engineering – Key Elements
- IoT Properties
- SE-IoT Motivation and 3 Key Issues
- Summary

What Does the IoT Encompass

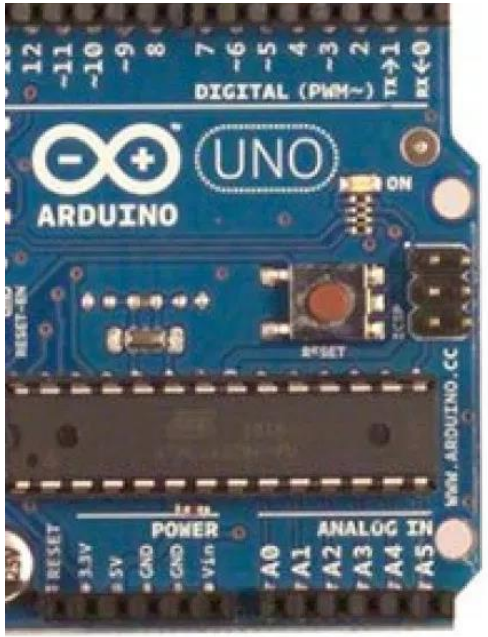
- IoT devices are special-purpose (computers are general purpose)
  - Microcontroller (Audrino) and Microprocessor (Raspberry Pi) w/ RTOS
- Hardware size and power
  - Smaller, lighter, less power (older tech nodes) – Resource constrained
- Yet many IoT devices need significant computation and speed
  - Speed-to-text, audio processing, network communication
- Internet accessible – Wired, wireless (including LiFi), cellular, 4/5 G, edge and cloud computing
- Software – firmware, RTOS, OS, NOS, Middleware, Apps, etc
- Data – Big, Little, AI, machine learning, high-level synthesis downward
- Security – Maybe the most important element
- Business models

Properties of IoT Systems (continued)

- Wireless Internet Access
  - Wireless access (cell phone, Wi-Fi) enables networking with cheap infrastructure
    - Less need to install physical cables
  - Data costs are fairly low
    - This point is arguable, but many can afford it
  - Data bandwidth is high
    - Can stream multiple movies in real-time
- Interface to the Edge (gateway) and the Cloud
  - IoTdevice interfaces can leverage powerful servers and large databases
  - Ex: Siri enables search with verbal questions

Raspberry Pi or Arduino Uno

Assignment for Week 2 in UC-I IoT course
→ Done from a systems tradeoff perspective



VS.

UCI Division of Continuing Education

Courses Certificates International Corporate Education Student Resources About

> Certificates > Engineering > Internet of Things

Internet of Things (IoT): Applications and Opportunities

Agenda

- Lifecycles and Legacies
- Systems Engineering – Key Elements
- IoT Properties
- SE-IoT Motivation and 3 Key Issues
- Summary

Motivation For Systems Engineering

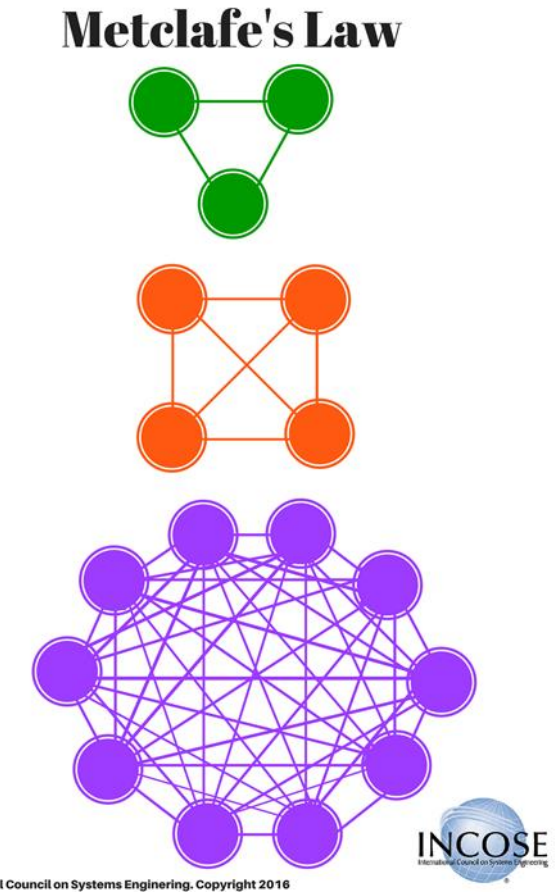
- IOT promises huge revenue potential
  - Amount of devices that connect to the internet will rise from about 13 billion today to 50 billion by 2020
  - By 2025, five sixths of semi growth is going to be the result of AI.
- Moore's Law "slowdown" supports IOT
- Commoditization of hardware
- Aging workforce
- Aging industrial infrastructure
- Others??

Issue #1 IOT vs Embedded

- IoT is the networked interconnection of everyday objects with embedded computers, sensors and actuators
  - IOT = **Connected** embedded devices
  - “Things" (embedded system devices) sense and collect data and send it to the internet. This data can be accessible by other "things" too.
- Connectivity – Data – Security
  - Connectivity means IOT has huge data side
  - Big data industry is expected to grow from US$10.2 billion in 2013 to about US$54.3 billion by 2017.
  - Handling the data is a big deal → Hence topics like machine learning.
  - Dark side of connectivity: Everything fails if security is lacking!

IOT Requires SOS Engineering

- Increasing interconnections between independent components or subsystems leads to more complexity and chance for system failures.
- Metcalfe's Law:
  - Three interconnected components result in a maximum of three interconnections
  - Four interconnected components results in a maximum of six interconnections
  - Ten interconnected components results in a maximum of 45 interconnections
  - Millions or billions of interconnected things on the Internet will results in how many interconnections?
    - Very high rate of complexity



Ease of HW-SW IOT Development

- Falling prices of sensors, low power embedded systems, network and connectivity devices
- All the major system vendors (IBM, ARM, NXP, etc) have demo's to easily put together an IOT applications
  - Smartphone, sensor, low power processor or Raspberry Pi = Done!
- Is the best way to build a complex, reliable system?

System of Systems

- Techniques of systems engineering (SE) have long history of developing complex systems
- But SE must change to meet the new IOT challenge:
  - Short time-to-market
  - Distributed global work force and supply chain
  - Ease of use – no difficult or problem-laden installation and operation
  - Dealing with hardware, software and non-traditional developers

Differences in SE and SW – HW Systems

- 10 Assertions in Paper

Requirements Criticality
Who tests the system?
Apportionment of complexity
Scope of "SE Work"
Interfaces
Engineering Degrees
Addressing "ilities"
Process focus
Object Orientation
Background and Emphasis

6 major differences

Physics
User Domain
Cohesiveness of HW and SW
Requirements Changes
Software Youth
"ilities"

Software Does Not Address “ilities” As Much As Hardware

Software SEs come from CS or math

Hardware SEs come from engineering

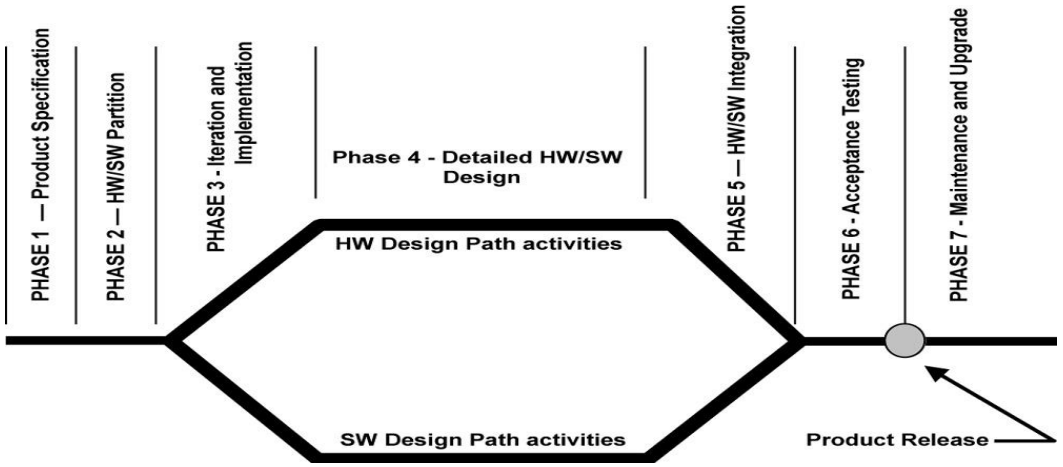
- Some software engineers consider hardware issues “out of scope”
- Little cross training across hardware and software domains

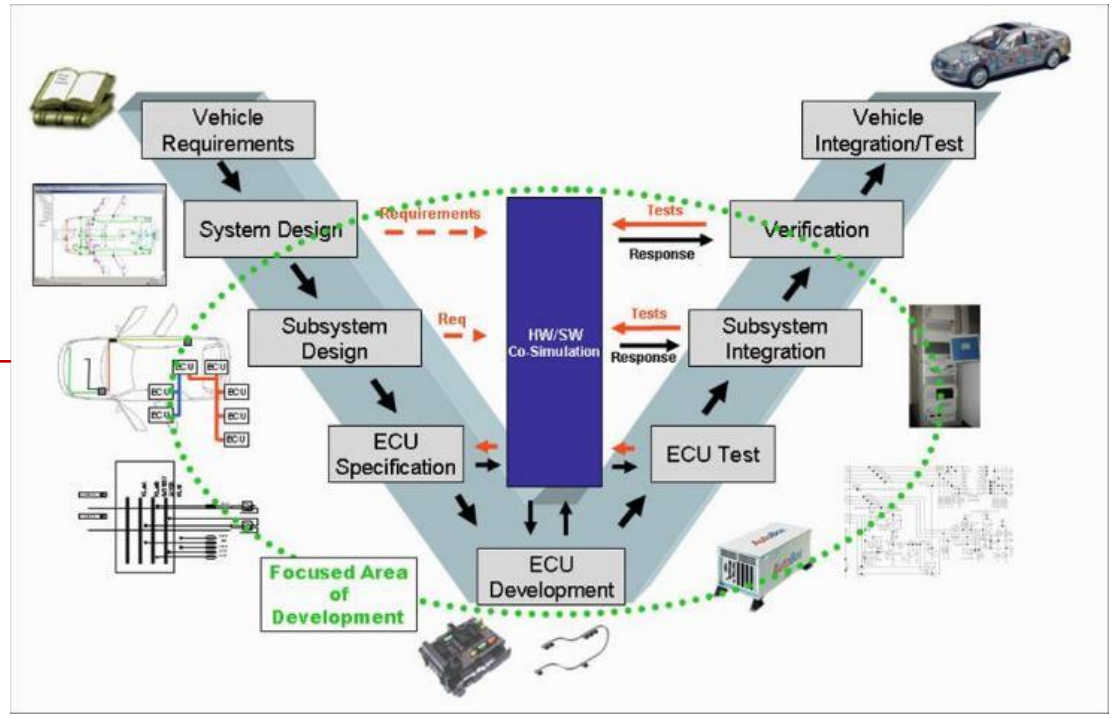
Software's "Ilities"

- **Design qualities*** ("conceptual integrity," maintainability, reusability*)
  Run-time qualities* (availability, interoperability, manageability,* performance, reliability, scalability,* **security**(see separate section))
  System Qualities* (Supportability, Testability),
  User qualities* (usability). Also modifiability.
- Reliability issues are not based on mechanical failures. More based on bugs, or unintended interactions.
- For vulnerabilities, redundancy doesn't work. If a flaw takes out one helicopter, it'll take out the second one too.
- SW architecture is largely driven by non-functional (ilities) = "quality attribute" requirements. Not always easy to fold this into systems engineering's functional decomposition (need something to track, something to target…)

*(Kaniss, Al. What is a software engineer, CrossTalk May/June 2015)

Embedded system Life Cycle





Ideally, development of Hardware and software goes in parallel.

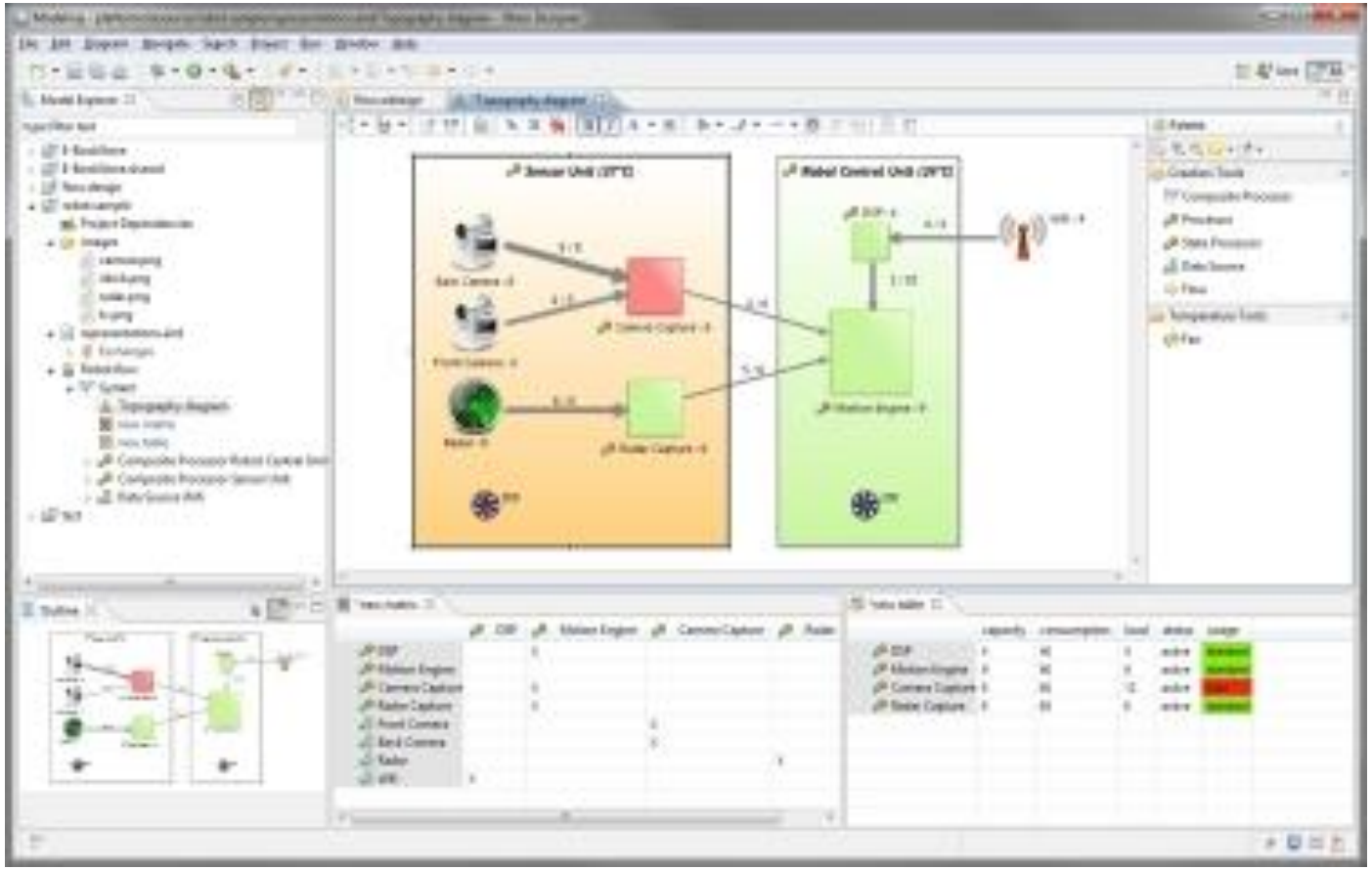
Methodology – HW-SW Integration

Alternative approaches

- Physical prototyping
  - Real pre-production hardware
  - Real-time execution
  - Poor controllability & visibility
  - Limited access – few prototypes
- Emulation
  - X86 based emulation hardware
  - HIL with real sensors/actuation
  - Close-to real-time execution
  - Improved control and visibility
  - Access limited only by hardware cost
- Simulation
  - No special hardware required as each ECU is modeled as a Win32 process
  - HIL with simulated sensors/actuation
  - Full control and visibility
  - Access limited by software cost

4 SL, IESF, June 2012

System Modeling Workbench Multidiscipline



Issue #2 – Engineering Expertise

- Recall: Issue #1 - IOT vs Embedded
- Issue #2 - How will semi and embedded HW-SW fabless Intellectual Property (IP) companies address large number of inexperienced SOC designers needed for IOT market?

Internet of Things Drivers

- Predicted **20 Billion** devices
- IoT is driven by **Data** and **Subscription** business models
- IoT devices are **enablers** rather than money makers
- Majority of devices will **NOT be wearables**
- There will be **tens of 1000's** of new IoT businesses
- ***Therefore,***

There will be >> 10,000's new IoT device designers

Thanks to Jim Bruister, CEO, SOC Solutions (from REUSE 2016)

Where will all these new IoT designers come from?

- Systems designers
- Software designers
- FPGA designers (large %)
- PCB Board designers
- Semiconductor designers (small %)
- College graduates

Majority are non-experienced in SOC design

Likely approach New IoT Designer will take?

- Google search SOC, Chip or ASIC
- Trade Magazines
  - EETimes, EDN, Manufacturing Today, *Sports Illustrated*, *Field & Stream*, etc…
- Look for SOC experts, semiconductor consultants
- Look at IP Portals
  - if they know about them
- Contact Design Houses
  - if they can find them

Issue #3– Complexity and Chaos

– Overwhelming number of life-cycle tools, engineering skill sets, life-cycle costs, IP acquisition and industry associations needed to navigate the IOT chip and embedded development process

New IoT Designer Anxiety Disorder

- TMI (information overload)
- Too many IP choices
- Tools sticker shock
- Design Flow complications
- Development Cycle elongation
- NRE fever
- **Where can I get help ?**

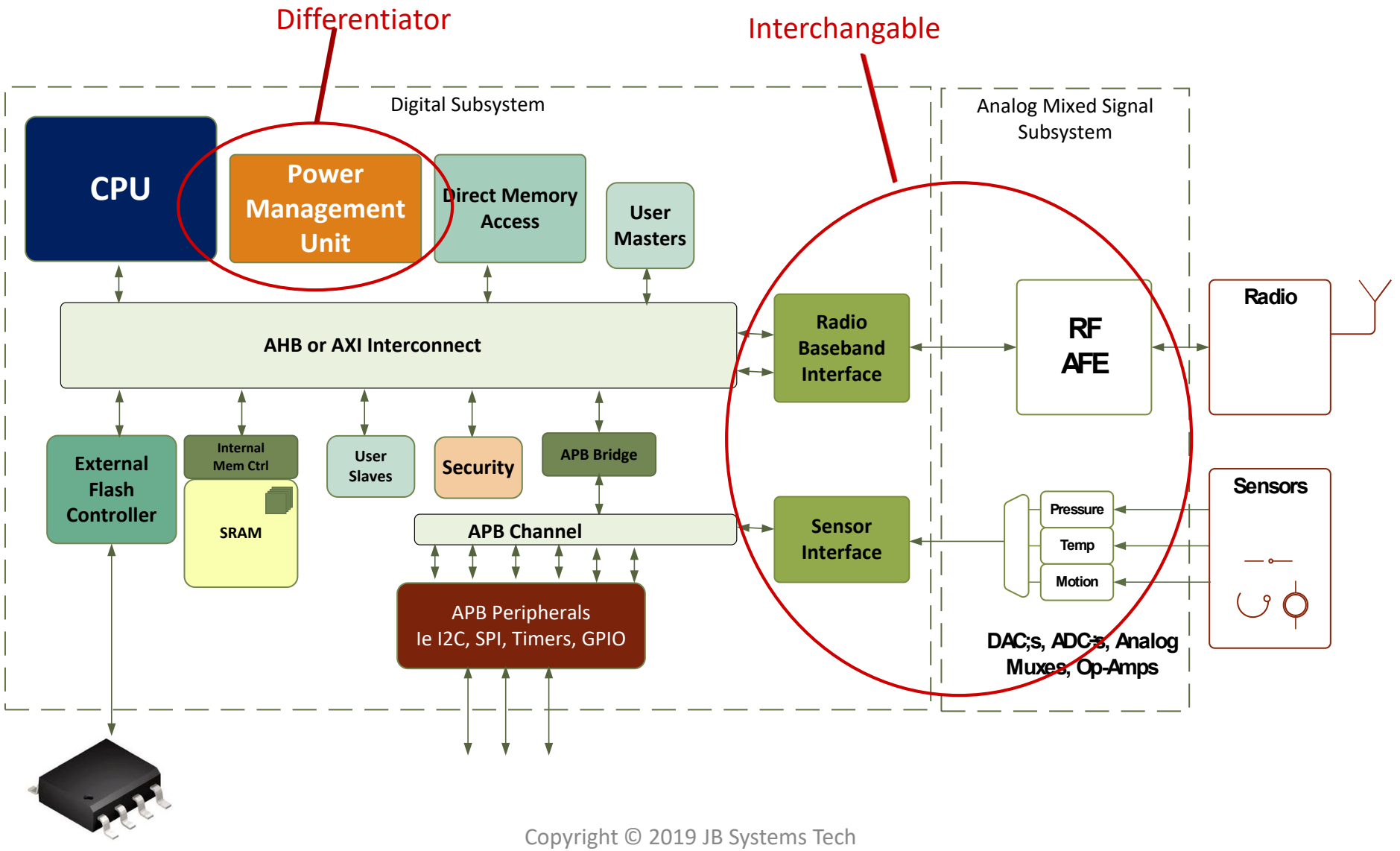
New IDeA Disorder

What's missing ?

From the scope of the New IoT Designer

- Practical education
  - "How to" guides, IoT and electronic (HW-SW-IT) design for Dummies
- Where to find pertinent information
  - Not the overwhelming info from Google searches, etc.
- Enough consultants or industry experts
  - Not enough to go around.
- Easy, Fast, Inexpensive, process from Concept to Product
  - Not enough "One stop shops" with streamlined design processes
- **Easy platforms or reference designs to start from**

Typical IoT SOC Architecture



Agenda

- Lifecycles and Legacies
- Systems Engineering – Key Elements
- IoT Properties
- SE-IoT Motivation and 3 Key Issues
- Summary

Design Abstraction

Raise the bar on design abstraction by providing:

- Modular Architectures
  - Specific to IoT Devices
- Class Libraries for hardware
  - Analog Models
  - Digital Subsystems
- Software Abstractions
  - Standard API's and HAL's

Think Arduino and Raspberry Pi

How do we improve? (continued)

- Systems engineers must employ agile (not necessarily Agile), lean techniques which bring rigor without aerospace-sized overhead.
- Ilities roles especially Security roles
  - Together work reliability, availability, maintainability, etc.
  - Partner regarding security. What SW vulnerabilities might be leverageable by adversaries to compromise systems (both IT and embedded <- newer)

Summary

- Developing an Internet of Things (IoT) system requires managing multidiscipline and multidomain system complexities. To be successful, this will require a tailoring of the traditional system-of-system (SoS) engineering approach.
- System engineers can no longer be **merely** process or requirements experts. They must also have some specific domain knowledge in hardware, software, network/connectivity AND data technologies. This represents a change from traditional systems engineering professionals.
- Model-Based Systems Engineering (MBSE) with domain knowledge will bring the engineering back into systems engineering and enable closer ties to system science.

Certificates and Continuing Education

- Systems Engineering
- Internet of Things
- Others

Required Courses

Title
Foundations of the Systems Engineering Process (2.5 units) EECS X491.81
Systems Requirements Engineering (2.5 units) EECS X491.71
System Design and Integration (2.5 units) EECS X491.94
System Validation and Verification (2.5 units) EECS X491.93

Elective Courses (Minimum 5 units)

Title
Simulation-Based Engineering of Complex Systems (2.5 units) EECS X429.2
Systems Engineering: Tools & Methods (2.5 units) EECS X491.98

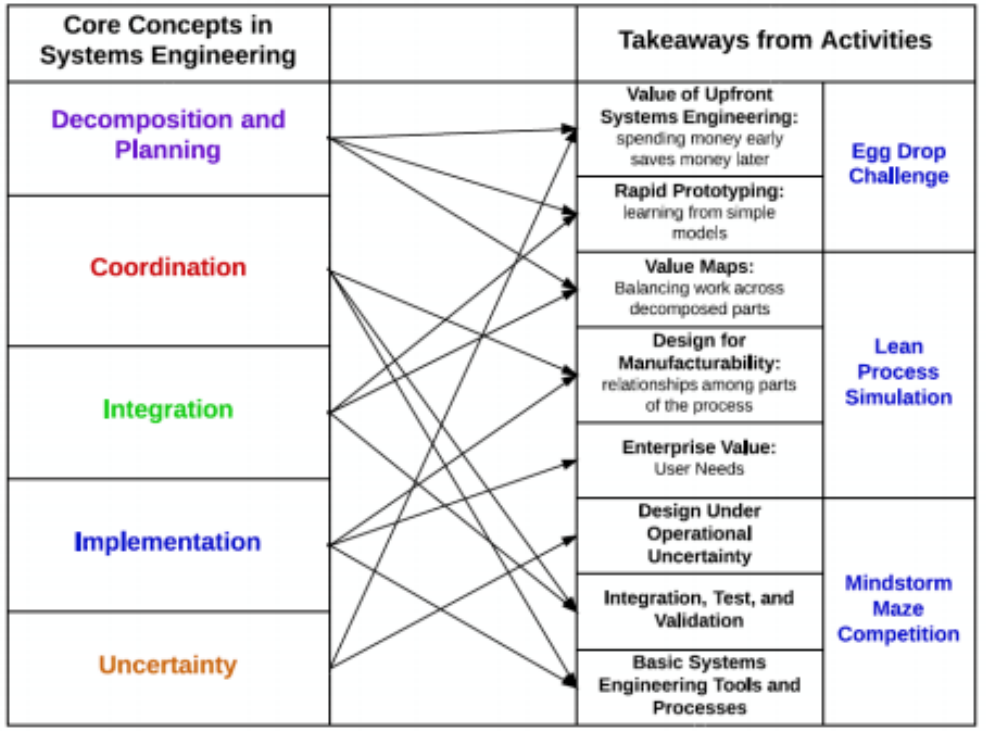
Course Schedule

Required Courses (9 units)

Title
Ambient Computing and the Internet of Things (IoT) (3 units) EECS X480
Designing and Integrating IOT Devices (3 units) I&C SCI X481
Networking and Securing IOT Devices (3 units) I&C SCI X482

Many Ways to Teach Systems Engineering

- Across a typical system or product lifecycle, from beginning to end
- In terms of key concepts or elements such as planning, design, operations, ilities, etc.
- Early NSF studies promoted active learning for online courses.
- Experience shows that case studies, short in-class and especially hands-on exercises have proven invaluable with either of the above two approaches.



Mapping of Systems Engineering Concepts to Hands-On Projects

Thank You!

Backup Slides