Ultra-Large-Scale (ULS) Systems

Kevin Sullivan
University of Virginia

A Simple Question

Given the issues with software engineering today, how can we build systems of the future likely to have billions of lines of code?

SEI Report on ULS Systems

- Report produced for U.S. Government by a group of scholars working with the Carnegie Mellon Software Engineering Institute
- Linda Northrop led the study group
- Ideas have achieved international visibility and are increasingly "in the air"

Report Author Team

- From the SEI: Peter Feiler, John Goodenough, Rick Linger, Tom Longstaff, Rick Kazman, Mark Klein, Linda Northrop & Kurt Wallnau

- Others: Richard P. Gabriel, Sun Microsystems, Inc. (now at IBM Research); Douglas Schmidt, Vanderbilt University; and me, Kevin Sullivan, University of Virginia

Study Group

- Gregory Abowd, Georgia Institute of Technology; Carliss Baldwin, Harvard Business School; Robert Balzer, Teknowledge Corporation; Gregor Kiczales, University of British Columbia; John Lehoczky, Carnegie Mellon University; Ali Mili, New Jersey Institute of Technology; Peter Neumann, SRI International; Mark Pleszkoch, SEI; Mary Shaw, Carnegie Mellon University; Daniel Siewiorek, Carnegie Mellon University; Jack Whalen, Palo Alto Research Center (PARC).

Reviewers

- John Bay, Air Force Research Lab; Brian Barry, Bederra Corporation; Barry Boehm, University of Southern California; Larry Druffel, South Carolina Research Authority (SCRA); Peter Freeman, National Science Foundation; Ron Goldman, Sun Microsystems; Watts S. Humphrey, SEI; Bruce Krogh, Carnegie Mellon University; Jim Linnehan, ASA ALT; Martin Rinard, Massachusetts Institute of Technology; Dennis Smith, SEI; and Guy Steele, Sun Microsystems, Inc.

From BLOC to ULS Systems

- We took BLOC as proxy for complexity in many forms
- Future systems will integrate and orchestrate the actions and evolution of thousands of platforms, decision nodes, sensors, machines, organizations, processes
- And they will adapt continuously to compensate for changes in needs and environments

What’s New?

- We have long lived in a world of ULS systems
- What’s really new is the pervasive *cyber* element
    - Enables systems with radical forms and scale
    - Becomes a dominant concern in system design

Basic Premises

- Today's SE inadequate even for current systems
- Future systems will push SE to untenable point

- Study concludes need for *breakthrough* research, not just incremental extensions of current work

- Software engineering research at a crossroads.

Software Engineering at Crossroads

- SE research achieved a great deal
- But not enough to serve needs of new systems
- And maybe not so much lately.
- Step back and take stock.

ULS Systems Report (2006)

- *Fundamental gaps in our current understanding of software and its development at the scale of ULS systems present profound impediments* to the achievement of mission objectives. These gaps are strategic, not tactical. They are unlikely to be addressed by incremental research in established categories. We require a broad new conception of both the nature of such systems and new ideas for how to develop them.

NSF CISE

- http://cise.nsf.gov (2009): CISE invites researchers to rethink the science and engineering of software - from the basic concepts of design, evolution, and adaptation to advanced systems that seamlessly integrate human and computational capabilities….

Major Themes

- We're facing demands for new kinds of systems
- Software is somehow at heart of phenomenon
- Conventional assumptions, concepts, methods, and tools are somehow *fundamentally* inadequate
- Radically perspectives now needed to succeed

SEI Conclusion

- Need to shift our perspective
  - how we characterize the problems we face
  - new ideas on how to address them
- New perspectives will be arise from work at intersection of normal SE & other disciplines:
  - microeconomics, biology, city planning, anthropology, etc
  - fields concerned with people as well as with coherence in the context of scale and complexity.

NSF “Rethinking Software” 2009

- CISE seeks ground-breaking, transformative research that will produce fundamentally new ways of thinking about how to develop, sustain, and reason about software, both during its design and deployment
- Such research will articulate new research challenges that cannot be addressed with existing software concepts, methods and tools
- CISE will [place] a premium on … proposals that push the frontiers of software research [and] cultivate partnerships between traditional software researchers and those from other areas within and outside of computing

This Talk

- Succeeds if it encourages conversation
- Will leave more questions than answers
- Body of talk
  - Survey of major ideas in SEI report
  - Personal reflection: software & systems engineering
  - What of components in ultra-large-scale systems?

SEI Report is Radical at its Core

- Questions engineering paradigm dating to 1968 NATO report: we aim to be *engineering discipline*, connoting tight, centralized control over design, development, and operation of SW & systems

- Key idea: in the largest scale human–built and natural systems engineering is often <u>not</u> source of effective organization

Examples

- Electrical and water systems are engineered, but cities generally are not—although their forms are regulated by natural and imposed constraints
- Firms are engineered, but the structure of the economy is not—although it is highly regulated
- Ecosystems exhibit high degrees of complexity and organization, but not through engineering

Change in Perspective

- From *direct* satisfaction of *coherent* requirements by top-down, centralized engineering planning & control

– which is how we view software development today –

- To *indirect* satisficing of *conflicting* requirements by the regulation of complex, ***decentralized systems***

Analogies

- Cities vs Buildings
- Socio-Technical Ecosystems
- Economies

Cities vs. Buildings

- Producing a city not a scaled-up version of producing a building
  - Cities not conceived, built, or changed by single organization or group
  - Emerge from regulated actions of individuals acting locally over time
- Regulatory mechanisms include
  - government organizations and policies
  - building codes, zoning laws, city planning
  - economic forces and incentives
  - available infrastructure systems
- ULS systems should be thought of as more like cities than buildings, and should be developed accordingly

Socio-Technical Ecosystems

- ULS systems are more like ecosystems
- Dynamic communities of interdependent and competing organisms (people, organizations, sectors) in complex & changing environments
- Complex, dynamic, evolving, decentralized, hard-to-predict, difficult to monitor, niches, robustness, survivability, adaptability, health, …
- What are the software issues for such *ecosystems?*

Economies

- ULS systems more like economies than firms
- Competition for resources is inherent
- Decentralization of decision-making control
- Regulations, incentives, and mechanisms
- Macro measures of overall performance
- Evolution of frameworks over time

Structure of the SEI Argument

- Distinguishing characteristics of ULS systems
- Major research challenges posed by ULS systems
- Seven proposed research areas for ULS systems

Characteristics of ULS Systems

- Decentralization in fundamental dimensions
- Conflicting, unknown, & diverse requirements
- Continuous evolution and deployment
- Heterogeneous, inconsistent, changing elements
- Deep erosion of the people-system boundary
- Failure normal & frequent, not rare & abnornal

Challenges Posed by ULS Systems

- Design and evolution
- Orchestration and control
- Monitoring and assessment

Design and Evolution

- Example: **economics and industry structure**
- Structure industrial ecosystems and harness their capabilities and motivations to find high-value regions in complex ***problem*** and ***design spaces***
- Align technical architectures with economics and social dynamics of ULS system evolution

Design and Evolution

- Co-existence of conflicting requirements
- Modeling and analysis of social interaction
- Governance mechanisms and processes
- Shared major infrastructure systems & services
- Integration & assurance across major boundaries

Orchestration

- How to maintain reasonable harmony among the components of vast and complex systems, under conflicting goals of self-interested parties

- adaptation to users and contexts
- enabling of user-controlled orchestration
- design & execution of policies, rules & forces
- online continuous updating of system elements

Monitoring & Assessment

- Monitor, assess, and, to extent possible, manage overall state, behavior, health, and well being
- Scale, decentralization, distribution, heterogeneity pose big challenges to monitoring and assessment
- Macro-metrics, like GDP or unemployment rate?
- Address well being of the human, organizational, economic, and business elements of ULS systems because they are essential parts of these systems

Breaking Traditional Assumptions

- The problem to be solved must be understood
- Requirements must be known before construction
- Conflicts must be resolved before construction
- Tradeoffs, once made, are considered stable
- Improvements are made at discrete intervals
- The effects of changes can be predicted well
- Configuration is accurate & tightly controlled
- Components & users are fairly homogeneous

Breaking Traditional Assumptions

- People are just users of the system
- Social interactions not particularly relevant
- Failures are abnormal, undesirable & infrequent
- Defects can be detected and removed
- A prime contractor & integral supply chain is responsible for system development & operation

Research Agenda

- Human Interaction
- Computational Emergence
- Design
- Computational Engineering
- Adaptive System Infrastructure
- Adaptable and Predictable System Quality
- Policy, Acquisition, and Management

Human Interaction

- Devise ways for anthropologists, sociologists, & other social scientists to conduct detailed socio-technical analyses of user interactions in the field, to better understand how to construct and evolve ULS socio-technical ecosystems
- Modeling users and user communities
- Fostering non-competitive social interaction
- Context-aware assistive computing

Computational Emergence

- Devise methods and tools based on economics and ***game theory*** (e.g., ***mechanism design***) to promote globally optimal ULS system behavior despite presence of many self-interested parties
- Explore *metaheuristics* and *digital evolution* to augment cognitive limits of human designers
- See work of Wallnau et al., on SEI ULS site, for work in algorithmic mechanism design

Design

- Design of all levels of ULS systems: e.g., not only of software artifacts but organizations, social networks, economic structures, whole development ecosystems
- Exploit concepts of design rules and evolution by value-seeking, highly decentralized, complex adaptive systems (e.g., work of Baldwin/Clark)
- Assimilation of diverse complex components into architecturally coherent ULS systems

Computational Engineering

- Improve the expressiveness of representations to accommodate semantic diversity of many languages
- Provide automated support for computing the evolving behavior of components & compositions
- Develop methods of assurance and certification to address need for high assurance of quality attributes in ULS systems

Adaptive System Infrastructure

- Development environments and runtime platforms to support decentralized development, analysis, governance, evolution of ULS systems
- Evolutionary development & deployment of ULS systems in deployment environments
- View-based evolution, through key abstractions

Adaptable & Predictable System Quality

- Devise ways to maintain quality in a ULS system in the face of continuous change, failures, and attacks
- Develop approaches to identify, predict, and control *system health* appropriate given the scale of ULS systems
- Security, trust and resiliency at ultra-large-scale

Policy, Acquisition & Management

- Transform government acquisition policies and processes to accommodate rapid and continuous evolution of ULS systems
- Treat suppliers, supply chains & industrial ecosystems as intrinsic and essential components of ULS systems

Capabilities & Mission Impact

- Common operating picture across ULS systems
- Survivability under failure, disaster & major attacks
- Rapid reactive fielding of new capabilities at scale
- Dynamic adaptation to changing environments
- Secure sharing across governments & industry
- Combining right information with local context

➢ ***Unprecedented performance in complex missions***

Mission Domains

- Health Care
- Energy
- Defense
- Transportation
- Finance, etc.

A Personal Reflection

- Group struggled to maintain focus on *software* element of ULS systems, given the pull of deep, interesting, and fundamental *broader systems issues*

- Expertise of group mainly in software and IT, not in systems engineering

- Something going on that we need to understand

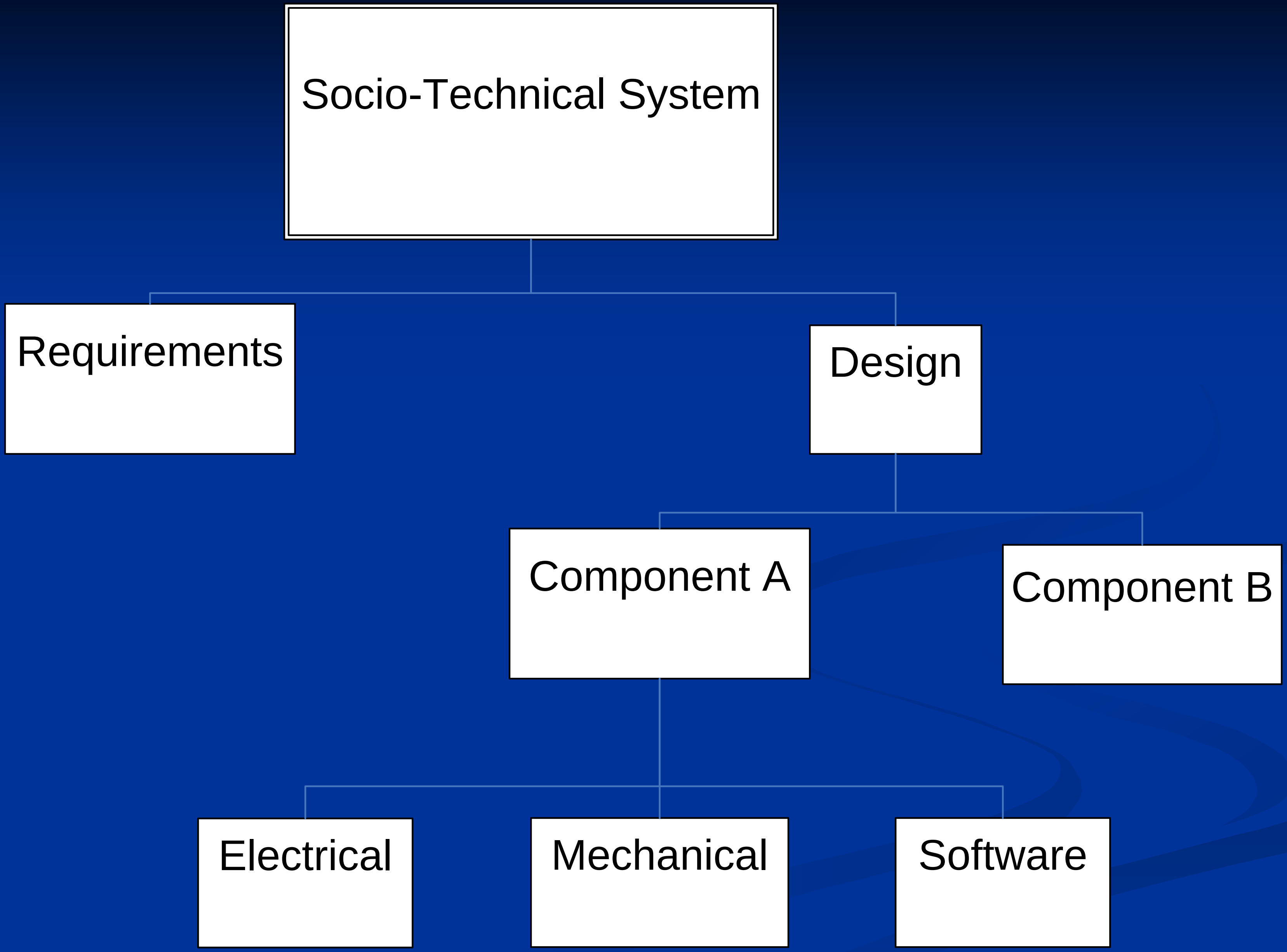
Tension Clear in Words we Used

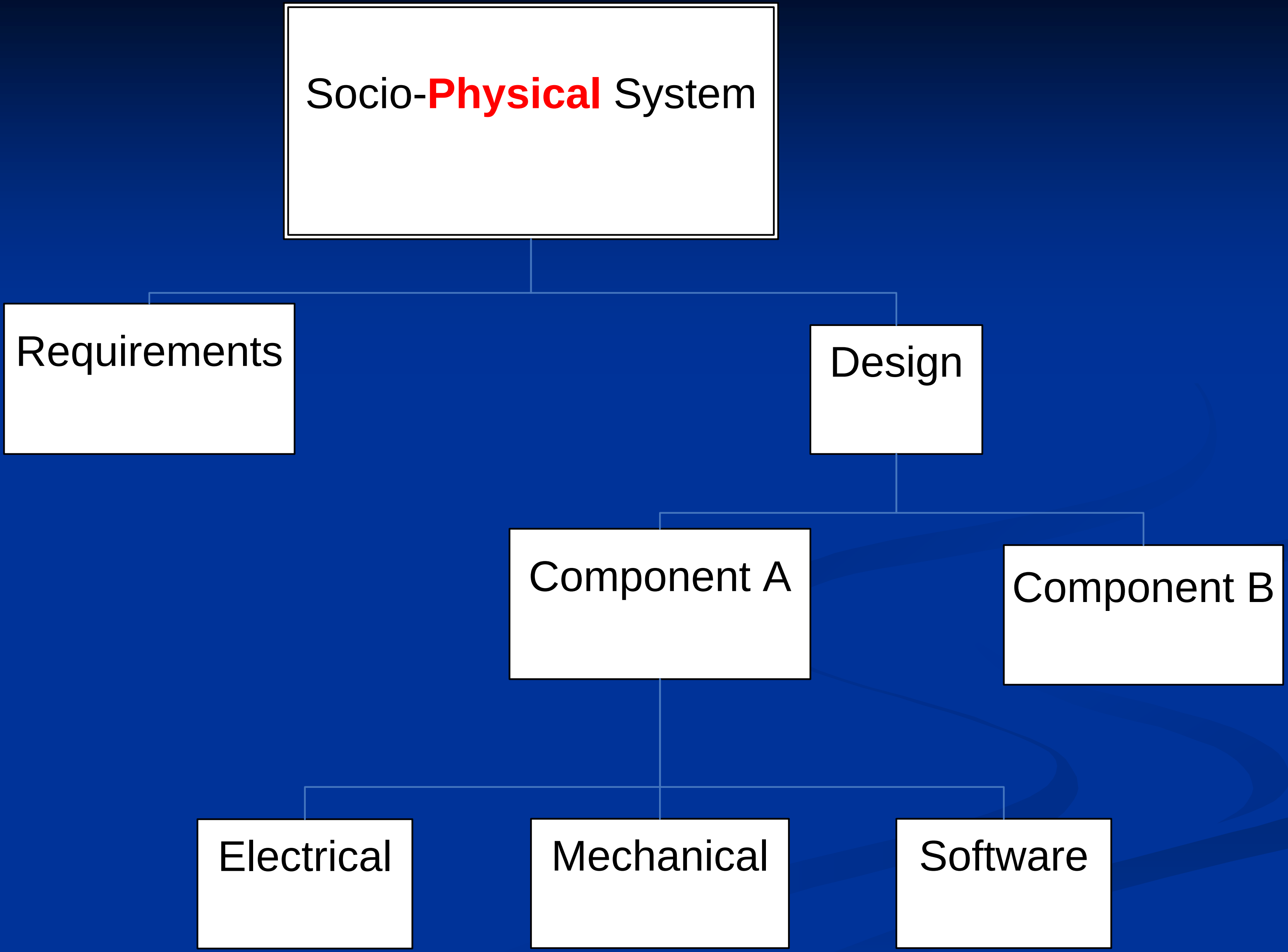
- SEI: “We require a broad new conception of both the <u>nature of such systems</u> and new ideas for how to develop them.”
- NSF: “ … <u>advanced systems</u> that seamlessly integrate human & computational capabilities.”
- Software Engineering vs Systems Engineering

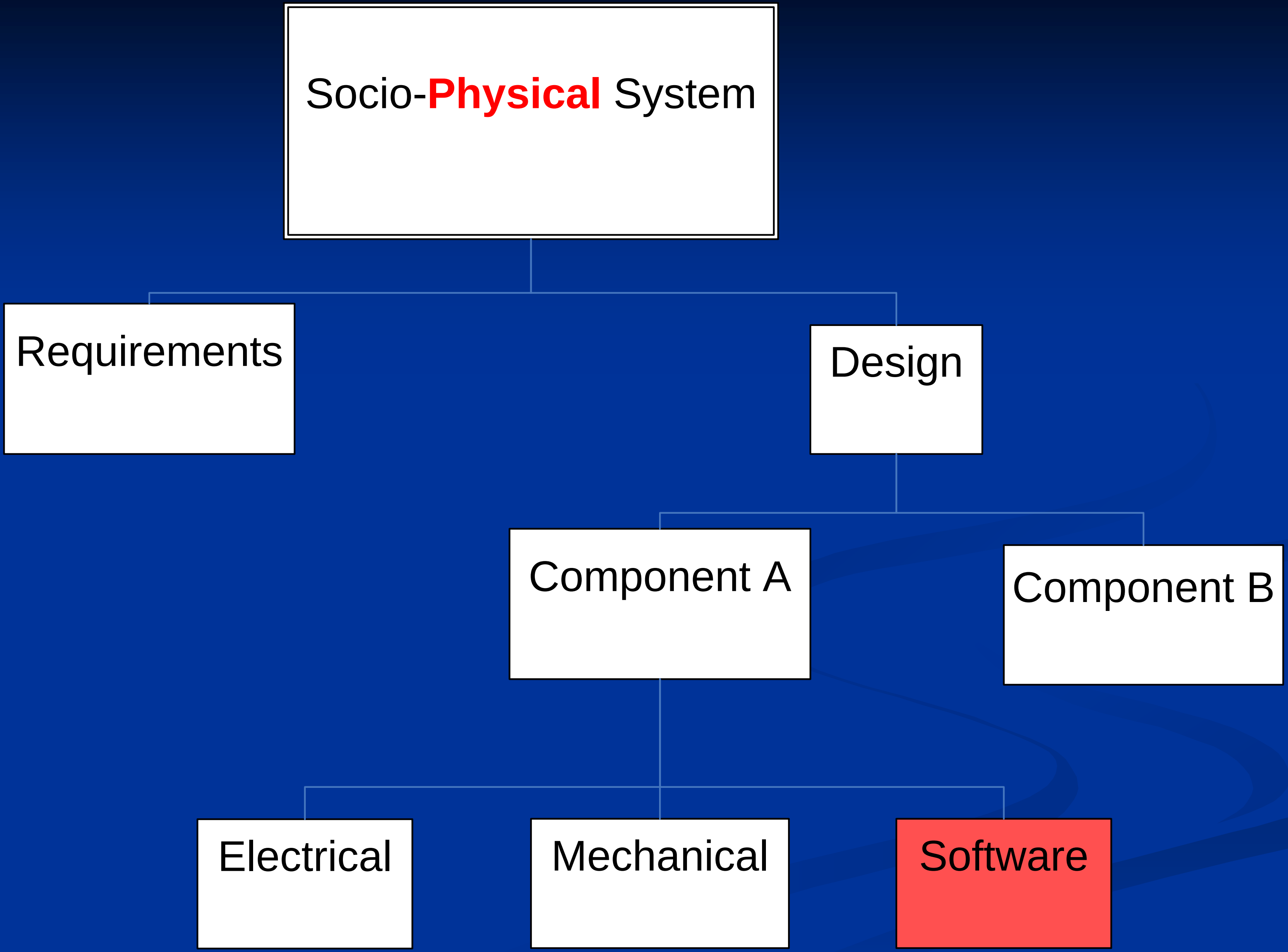
Traditional Systems Engineering View

- Systems engineers
  - Determine *system* requirements & manage tradeoffs
  - Derive and partition technical specifications
  - Allocate specifications to component disciplinary groups
  - Responsible for system integration and assurance

- Software as sub-component of a system
  - Software engineers receive component specifications
  - Responsible for producing implementations to spec
  - And for providing assurances to systems engineers

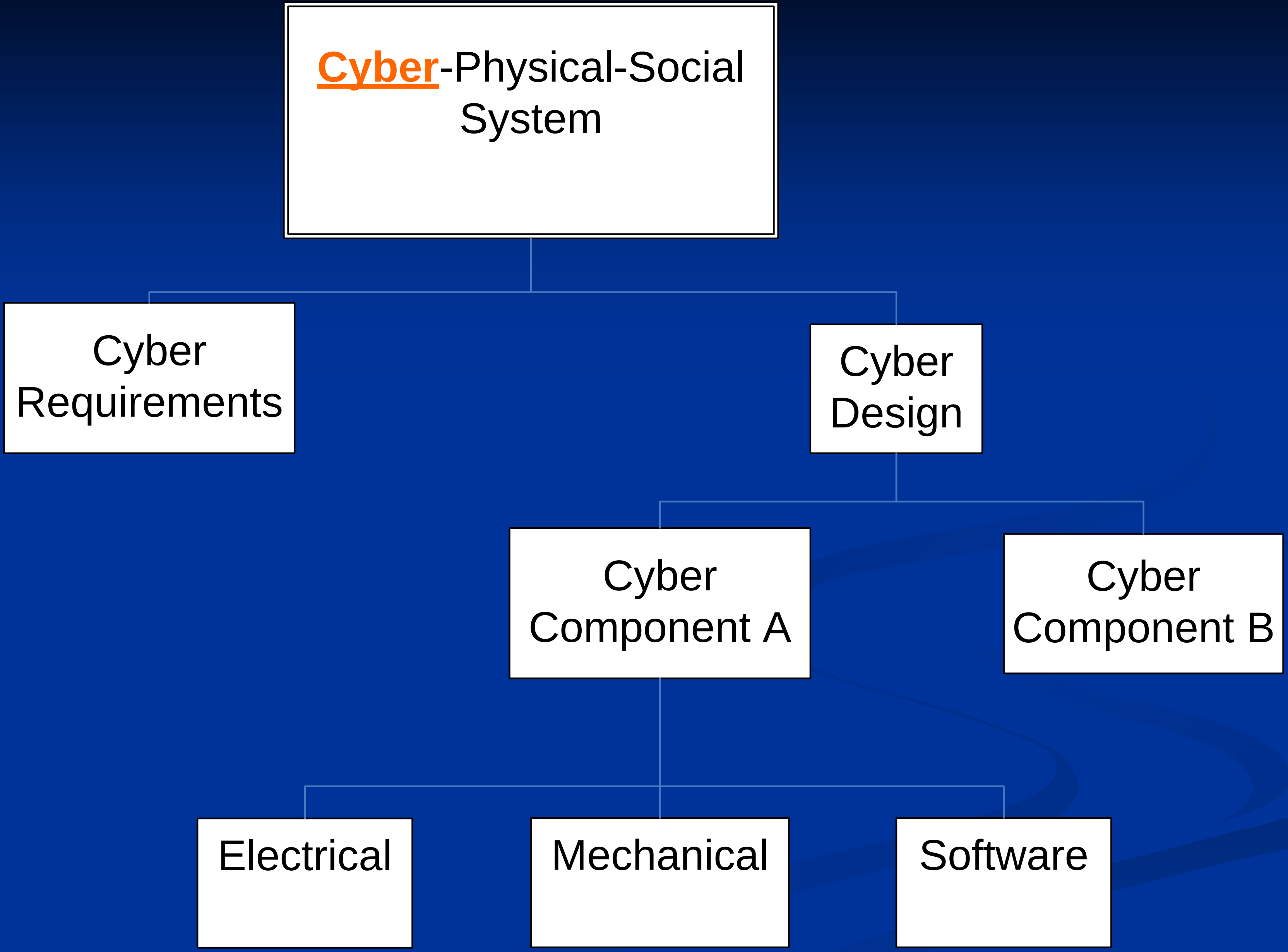

Socio-Technical System
Requirements
Design
Component A
Component B
Electrical
Mechanical
Software

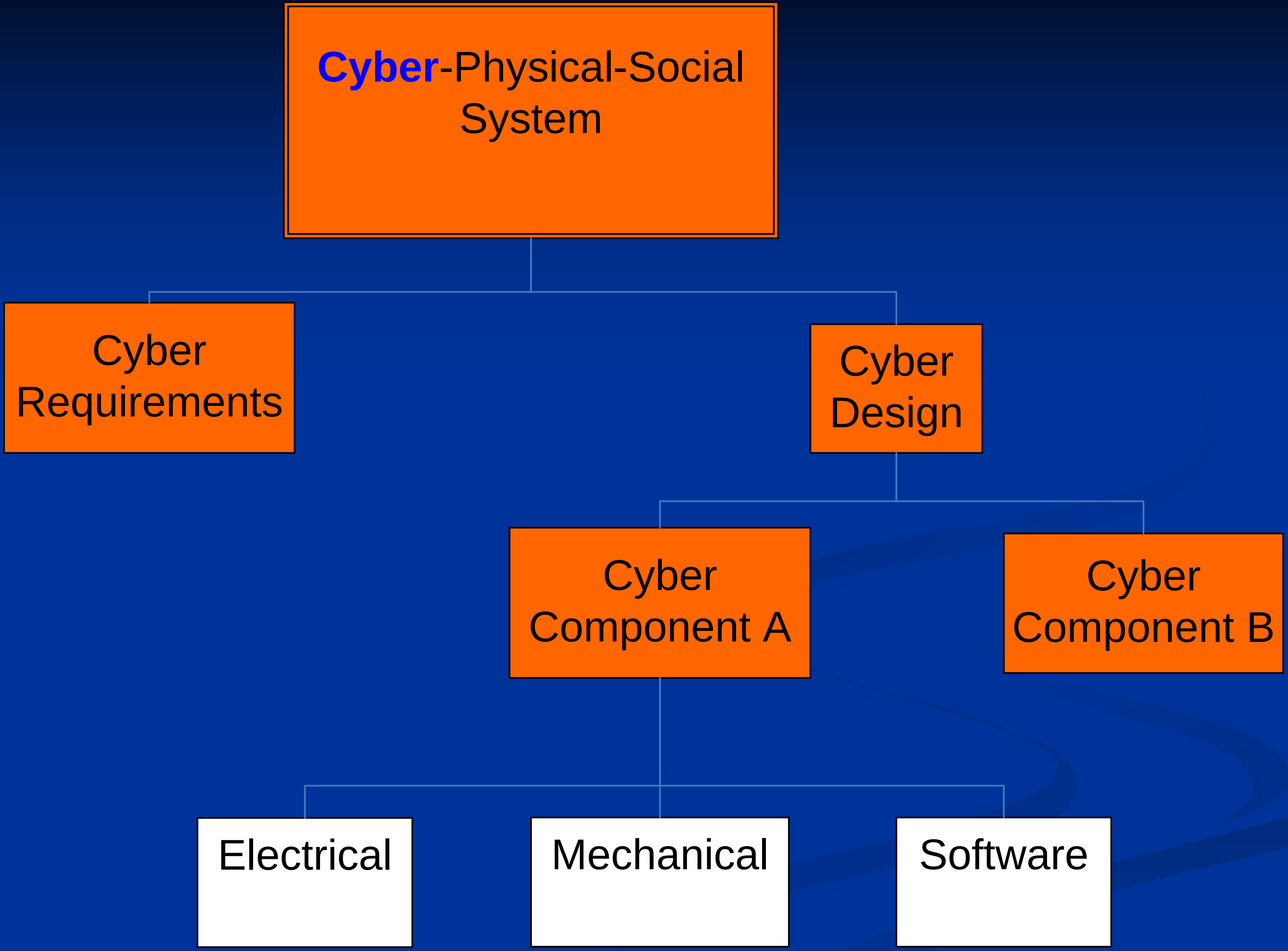

Socio-**Physical** System
Requirements
Design
Component A
Component B
Electrical
Mechanical
Software


Socio-**Physical** System
Requirements
Design
Component A
Component B
Electrical
Mechanical
Software

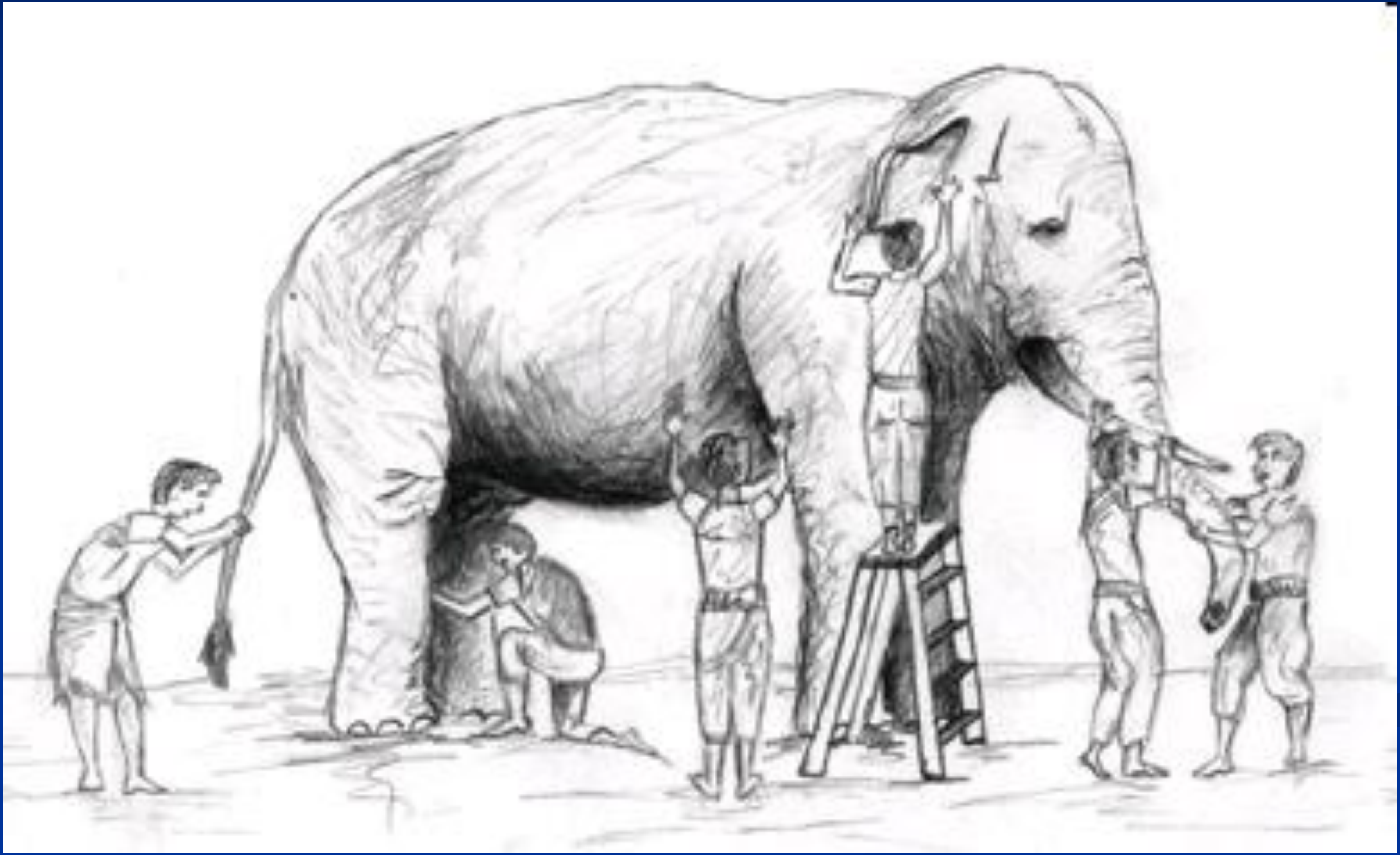
Doesn’t Work Well for ULS Systems

- All manner of function, risk & complexity forced into software components of systems
- “Software is soft” & so can accommodate all manner of late-breaking epiphanies/problems, right?
- Software then gets blamed for system failures, whether in procurement, operation, or evolution


Cyber-Physical-Social System
Cyber Requirements
Cyber Design
Cyber Component A
Cyber Component B
Electrical
Mechanical
Software


Cyber-Physical-Social System
Cyber Requirements
Cyber Design
Cyber Component A
Cyber Component B
Electrical
Mechanical
Software

Seeing Different Parts of Elephant



Problem

- Issues traditionally handled by systems engineering now in domain of experts in software/computation
  - *System*-level requirements
  - Cyber-enabled *system* architectures
  - Economic, social, human factors issues & methods
- Software engineering not set to address these issues
- Traditional systems engineering not well set up to handle complex software and computational issues

Thus Two Distinct, Related Issues

- Transition from conventional to cyber systems, challenging both software & systems engineering
- Transition from conventional to ULS systems, challenging engineering perspectives altogether
- Software/IT driving both transitions
  - Principal *enabler* of new class of ULS systems
  - Dominant technical *concern* at the system level

Where Do We Go From Here?

- We really *do* need to rethink software research
- New synthesis of system & software engineering
- Systems = people + IT + hardware + physical world + economics… *integrated by & performing computations*
- Look beyond traditional engineering for sources of evolving structure, function and quality
- Find news ways to support conception, realization, operation, sustainment & evolution of ULS Cyber-Physical-Social Systems

Conversation: Implications for Components?

INCOSE Definition

- Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem… Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.

- http://www.incose.org

NSF Workshop

- For example, the scale and distributed nature of the systems now being envisioned suggests the need for significant changes in traditional views of ideal software development. While tight, centralized managerial and engineering control of development based on unrestricted access to artifacts and processes will arguably continue to be vital at the component level, for instance, the software that runs large systems increasingly will be produced by, and will operate within, distributed socio-technical ecosystems, not all of whose participants have naturally shared interests. The cost and performance of the resulting systems will depend not only on traditional controls, but on the organization, regulation, analysis, and evolution of networks of several kinds: Software components, sometimes delivered as services, connected into *architectures* that cross organizational boundaries, interacting over *communication networks;* technical decisions connected by networks of constraints and objectives; development activities connected into networks of *tasks and processes; arguments* about design properties of components, and bodies of supporting evidence, connected into dependability cases; people connected in *social networks*; organizations connected in *economic, contract, trust,* and *transaction networks*. To the extent that the cost and quality of software, and thus systems, depends on the structure and performance of diverse networks, then finding effective methods for analyzing, organizing, regulating, and evolving them becomes a central concern in software engineering.

Maier's Systems-of-Systems

- Operational independence of elements
- Managerial independence of elements
- Evolutionary development
- Emergent behavior
- Geographic distribution

- His *virtual systems of systems* closest to ULS systems:

  Virtual systems lack a central management authority. Indeed, they lack a centrally agreed upon purpose ... Large scale behavior emerges, and may be desirable, but the super-system must rely upon relatively invisible mechanisms to maintain it.